

INTEGRATION OF DATA BASE
MANAGEMENT SYSTEM WITH
OPTIMIZATION

By

ASHOK KUMAR RATHI

Bachelor of Commerce

The University of Rajasthan

Jaipur, India.

1983

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
Master of Business Administration
May, 1987

Name: Ashok Kumar Rathi

Date of Degree: May, 1987

Institution: Oklahoma State University

Location: Stillwater, Oklahoma

Title of Study: INTEGRATION OF DATA BASE MANAGEMENT WITH OPTIMIZATION

Pages in Study: 60

Candidate for Degree of Master of
Business Administration

Major Field: Business Administration

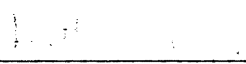
Purpose of the Study: Much time is spent managing the data base and modeling an optimization problem. This report proposes the use of data base management system with optimization. This integrated system offers many advantages compared with the conventional ways of modeling the optimization problem. The integrated system makes use of the existing data base, generates the relevant problem, solves it and produces the necessary reports for the user. An example presented in this study illustrates the entire process of such an integrated system.

Findings and Conclusions: The DBMS based optimization can save the modeler a lot of time and effort. Also, this can improve the overall reliability and accuracy of the model. The modeler consequently has more time to concentrate on the model. Data entry/updates are not duplicated. Updates to the optimization problem can be made simply by updating the data rather than the problem itself. The user can generate a wide variety of reports on the problem solution, which are more understandable and easier to interpret than the problem solution itself. The conclusion is that the DBMS-based optimization is recommended over ad hoc data and ad hoc report procedures.


ADVISER'S APPROVAL _____

INTEGRATION OF DATA BASE
MANAGEMENT SYSTEM WITH
OPTIMIZATION


Report Approved:



Report Advisor



Director of Graduate Studies



Head, Department of Management

ACKNOWLEDGEMENTS

I wish to acknowledge here Dr. Scott Turner and Dr. Ramesh Sharda for their support, encouragement and advice on this report. Their patience and guidance is sincerely appreciated. I also thank Office of Business and Economic Research staff for providing all the facilities needed for this report. My special thanks go to my sister for her continued moral support.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION	1
II. LITERATURE REVIEW	4
Data Base Management	4
Matrix Generators	5
Modeling Languages	6
User's Interaction, Reporting and Display Capabilities	7
Summary	8
III. ADVANTAGES OF INTEGRATING DBMS WITH OPTIMIZATION	9
Use of Existing Data Base	9
Data Management and File Management	10
Application Specific Problem Generation	11
Output Processing	12
Generating Intelligent Reports	13
IV. COMPONENTS OF THE INTEGRATED SYSTEM	14
Data Structure and File Structure	15
Data Management and File Management	16
Matrix Generation	17
Matrix Structure & Variable Names	17
Maintenance of Matrix Generators	18
Matrix Display for the User	18
A Modeling Language	19
Report Writing	19
A Sketch of DBMS-based LP System	21
Summary	23
V. AN EXAMPLE ON JOB-SHOP PROBLEM	24
Problem Definition	24
Problem Analysis	28
Brief Overview of the Integrated System in This Example	28
Detailed Description of the Example	30
Input Preparation	30
Variables Generation	31
Matrix Generation	33

Chapter	Page
Optimization	35
Solution Output Processing	38
Report Writing	40
VI. SUMMARY & CONCLUSIONS	45
A SELECTED BIBLIOGRAPHY	47
APPENDIX	49

LIST OF FIGURES

Figure	Page
1. Integration of DBMS and Optimization Program	22
2. Employee Records (ZEMPLYEE.DBF)	25
3. Structure of ZEMPLYEE.DBF	25
4. Pending Job Orders (ZJOBORDR.DBF)	25
5. Structure of ZJOBORDR.DBF	26
6. Job Categories and Hours to Complete them in Different Shops (ZHRSHIST.DBF)	26
7. Structure of ZHRSHIST.DBF	26
8. Cost Records (ZCOSTREC.DBF)	27
9. Structure of ZCOSTREC.DBF	27
10. Flow Chart of Job Shop Problem Illustrating Integration of dBASE II with LINDO	29
11. Input Prepared from the Data Base (ZINPUT.DBF)	32
12. Structure of ZINPUT.DBF	32
13. LP Variables in a Dictionary Form (ZLPVAR.DBF)	34
14. Structure of ZLPVAR.DBF	34
15. LP Problem in Data Base Format (ZLPMATRX.DBF)	36
16. Structure of ZLPMATRX.DBF	37
17. LP Problem in ASCII Format (ZLPMATRX.TXT)	37
18. LP Problem as Restructured by LINDO	39
19. LP Solution Generated by LINDO	39
20. LP Variables and their Solution Values	41

Figure	Page
21. Menu for Report Generation	41
22. Report Format Generation in dBASE II	43
23. Report Options as Internally Stored by dBASE II	43
24. Report As Generated by dBASE II for Option 2	44

CHAPTER I

INTRODUCTION

Until a few years ago, data collection and its analysis consumed a significant amount of both clerical and managerial time which otherwise could have been used for other important functions like planning etc.. However that was the imperative of the time and not all the companies had the luxury of computers which could do the repetitive jobs for them. Not too many computers were available because of their high prices. Moreover, wherever they were available, they were not used to their fullest potential simply because adequate software was not available. But with the advent of minis and micros, the whole scenario underwent tremendous transformation. Now almost all companies are directly or indirectly seeking computers' help in solving day-to-day problems. For example, chemical companies use computers in solving blending problems of which the constraints change fairly frequently due to changing prices of the ingredients or for other reasons. Manufacturing departments are usually faced with the problem of resolving job priorities so that the overall manufacturing requirement can be met at a minimum cost. Today the use of computers has proven helpful in every sphere.

Managers typically face a range of planning problems that vary in size and scope with the demands of the day. With the advent of microcomputers and its widespread availability and applicability, the

outlook at different kinds of problems frequently faced by the managers has changed. Problems are attempted to be solved as precisely and as quickly as possible to grab the elusive opportunities. Software is now available for a wide variety of problems. Managers use different tools and techniques to arrive at the most satisfactory solutions. At the same time, business planning has become more complex, in part due to these advances in computer technology, causing managers to seek the powerful insights that models can provide.

Consider the following scenario. A company XYZ is a manufacturing concern which receives job orders at random. The Operations Research analyst collects the necessary data to determine the optimal number of hours that each shop should spend on each job in a certain period. Then he or she performs all arithmetic calculations, models the problem, enters necessary coefficients and variables in a desirable format. Finally the OR analyst solves it and reports the results back to the production department. After some time, another order is received which also needs to be executed in the same time period. The OR analyst is again called upon to repeat the entire process. Here the OR analyst gets into a messy situation. But, with the type of system in use, there is nothing that can be done to avoid that problem. The answer to such a problem is to use a system that would take care of all the changes made, reformulate the problem using the given model and produce the new results.

Manager's efficiency and effectiveness can be greatly improved if a very efficient database system can be made available. For example, if the manager is faced with a routine problem of setting up the job priorities with an objective of minimizing overall cost, it is

necessary that the data are easily fed into the analytical tool. The database can be built using database software.

This report mainly focuses on how a data base management system can be integrated with optimization techniques. This report consists of six chapters including this chapter. Chapter II embodies the literature review with respect to data base management, matrix generators, modeling languages, report writing etc.. Chapter III deals with the advantages resulting from integrating data base management system (DBMS) with optimization. Chapter IV discusses at length the various components of the integrated system. Chapter V contains an example on the job-shop problem illustrating almost all the components that an ideal integrated system should have. Chapter VI summarizes all these chapters and draws the conclusion.

CHAPTER II

LITERATURE REVIEW

Data Base Management

Bisschop and Meerus [2] observed that the major portion of most real world LP and other models consists of data and that these data must be managed effectively. The relatively limited use of models in our environment is partly due to the fact that a significant portion of total resources in a modeling exercise is spent on the generation, manipulation and reporting of models. Sharda [15] also presented similar views. Krabek, Sjoquist and Sommer [11] questioned the ignorance of the process required to manage the data, formulate and build the model, report and analyze the results.

In brief, a significant portion of time is spent on data management and problem formulation.

Sharda [15] proposed the use of spreadsheet-based optimization. Spreadsheet-based optimization has some limitations as discussed by Sharda, Turner and Rathi [16]. These limitations are given below:

1. The end user is expected to be expert in problem formulation. The spreadsheet does not formulate the problem in a desirable format.
2. The data are already stored on the computer in files or tapes etc. and still the user has to enter the data in a spreadsheet model.
3. If the problem size is very large, the spreadsheet model occupies a very large storage making it difficult to build large models.

4. Most of the existing spreadsheet packages do not allow the user to customize the screen for data entry and updates. Data entry and updates have to be performed using the given screen.
5. Spreadsheet packages usually do not have a modeling language. This precludes all the possibilities of modeling an optimization problem.

Krabek, Sjoquist and Sommer [11] called for a system that is capable of managing large amounts of data structured in a way meaningful to the modeling process. Such a system should have power and flexibility of large scale data managers.

Sharda, Turner and Rathi [16] have proposed the use of DBMS instead of spreadsheet. Some important advantages of integrating DBMS with optimization are discussed in Chapter III.

Matrix Generators

A matrix generator is a computer program that writes out the algorithm's form--the coefficient matrix--of an LP. Fourer [7] stated that such a matrix generator can be symbolic or explicit, general or specific, and concise or redundant to various degrees, but its most important qualities are understandability as a programming language and convenience to an interpreter. DATAFORM, DATAMAT, GAMMA, MaGen and OMNI/PDS, IBM MGRW, APEX-II MRG, and MODELER are some of the existing matrix-generation systems.

Matrix generators typically incorporate loops, assignments, transfers of control, or other executable statements [7]. Matrix generators are application specific. In other words, they are restricted to only one class of problems. However, matrix generation

is a definite advantage over translation by human.

Modeling Languages

Matrix generators have a few drawbacks. The major drawback is that they are written in a computer programming language not easily understandable to the user. Hence Fourer [7] proposed that a modeling language should be used in place of a programming language. He pointed out some difficulties with matrix generators--verifiability, modifiability, documentability, lack of independence and simplicity. Fourer also considered a hypothetical modeling language, XML, which is based on modeler's form. XML has easy indexing and arithmetic expressions, and algebraic notations. Modeler can easily write a problem model using XML.

Brown, Northup and Shapiro [4] developed a modeling and optimization system for business planning called LOGS. The Descriptive Modeling Language (DML) is the key feature of LOGS. This allows a large class of LP and MIP models to be described efficiently and parsimoniously. Rowland and Boudwin [14] developed PLATOFORM (PLanning TOol developed in the dataFORM language of MPS II). This is a model management framework for mathematical programming. PLATOFORM, a single generalized support system, is used by Exxon Corporation. The database contains translation dictionaries, data tables, LP matrices and solution cases.

Lucas and Mitra [12] developed a mathematical programming modeling system called CAMPS (Computer Assisted Mathematical Programming modelling System). This is an interactive system designed to aid model formulation, matrix generation and model development. CAMPS is divided

into subsystems such as INPUT, GENERATE, OPTIMISE, REPORT and UTILITIES. The entire system is controlled by a series of menus and screenforms.

Kisko [10] presented a computer readable format called LPL that attempts to be as much like the human syntax as possible. The LPL allows the user to define or specify such necessary model elements as variables, parameters, indices of vectors and matrices, and summations in LPL expressions. Bisschop and Meerus [2] developed a modeling tool called GAMS (General Algebraic Modeling System). The information content of the model representation is such that a machine can check for algebraic correctness and completeness.

User's Interaction, Reporting and Display Capabilities

Rowland and Boudwin [14] observed that the user's interaction with the system must be at the data level only. The user should not be expected to input coefficients directly but rather the user oriented data which the matrix generation module will use to formulate a problem. They proposed complete independence among data management, problem solution, and report writing modules.

Greenberg, Lucas and Mitra [9] stressed that sophisticated users should be permitted to interactively query through the model. The system should be a friendly conversational system.

Sharda [15] stated that the results stored in a spreadsheet can be used for easy generation of formatted, tabular reports. Spreadsheet-based optimization systems may also permit easy graphical display of at least some of the results. Sharda, Turner and Rathi [16] cited some limitations of report generation capabilities using a spreadsheet.

Summary

The literature review yields a few salient points. Data management has so far been an ignored area. The OR analyst has to spend much time formulating and debugging a problem. The data entry is duplicated increasing the probability of error. As a consequence, the analyst is not sure about the accuracy of computations and reliability of the results. A considerable amount of time is spent in understanding, interpreting and translating the results. This leads to poor reporting.

The solution to the above problems lies in integrating DBMS with optimization. Such an integration has been thoroughly explained and illustrated in subsequent chapters. It should be noted that no modeling language has been introduced in this report.

CHAPTER III

ADVANTAGES OF INTEGRATING DBMS WITH OPTIMIZATION

In Chapter II, we discussed some limitations of using a spreadsheet. These limitations become more serious as the data size increases. We proposed integration of DBMS with optimization. Sharda, Turner and Rathi [16] discussed some important advantages of such an integration, which are being elaborated upon below:

Use of Existing Data Bases

The LP problem is formulated using the data currently stored in the company data base. The data may be stored online or on tape etc.. No matter what mode of storage is in use, the fact remains that the data have already been entered and are readily available on the computer. Let us assume that currently no integrated environment exists in the company. In such a situation, the LP modeler shall be required to manually reenter the necessary data for problem formulation. This amounts to waste of time, effort and resources. Rather, the company could be able to use the existing data base if it had a DBMS in operation.

The DBMS based optimization allows the user to extract the relevant data from the existing data base. This avoids duplicate data entry and also ensures against errors that may otherwise occur in reentering the data. Besides simply extracting the data, the DBMS also

allows the user to perform necessary arithmetic computations on them. Totals, subtotals and aggregations can also be done easily. The arithmetic and aggregation capability is very crucial to any DBMS.

Data strewn across multiple files are not directly usable but rather must be subjected to manipulations for problem formulation. The DBMS can accomplish this easily. The integrated system simply requires a front end to communicate with the existing data base and create a new data base file relevant to the problem formulation. The data base package dBASE II for example, has its own data processing language called a command language. Such a facility can substantially reduce the modeling time.

Data Management and File Management

The DBMS also offers considerable facility on data management. While a particular coefficient in an LP problem, for example, may be needed at several places, it can be entered only at one place and be extracted therefrom when needed. This enhances the integrity of the model. Data entry and updates are also greatly facilitated.

An offshoot of such an integrated system is that the task of managing the data can be separated from modeling - a very important attribute of any DBMS. The modeler no longer has the responsibility of the accuracy of coefficients or other numbers since data entry/updates are done by another person. Thus, the modeler can concentrate on the model.

With the DBMS, file management becomes more efficient. Adequate security is provided to the files by requiring the user to enter the password before he can access them. The automatic back-up facility can

also be provided by the DBMS. The most important aspect of the DBMS with respect to file management is the availability of sharing capabilities. The files can be shared by multiple programs or multiple users. With the rapidly advancing network technologies, this feature is very crucial.

Application Specific Problem Generation

As indicated earlier, most of the matrix generators are application specific. An application specific matrix generator can internally generate only one class of problems due to the specific structure associated with each of them. Optimal Manager, MIXIT-2 and LPMMASTER are examples of software where the LP model is built internally by the system. The user does not see the internal formulation unless requested. For example, MIXIT-2 is designed for farmers to determine optimal feed mixes based on the costs of ingredients, nutritional requirements and constitution of the ingredients. A user of the program only answers the questions in terms of the problem at hand, not in terms of LP terminology. The model is generated by the program and solved using an LP program.

Most of the DBMS packages have the command or programming language which allows the user to write a program tailored to the specific application. Usually the programming language can extract data from multiple data base files, manipulate the data and store them back. They also allow exporting to and importing of the files from other packages through ASCII or other data exchange code.

With some DBMS, it is also possible to link other high level languages. Some possible applications may be the generation of an LP

problem, inventory problem, MRP problem etc..

The use of matrix generators vs. modeling languages has been debated. Models built in a strategic environment play a useful and even powerful role in the overall planning process. Fourer [7] advised that the matrix generator approach can be improved upon by use of modeling languages. Translation from the modeler's form to the algorithm form is an inevitable task in linear programming. Traditionally, the task of translation is accomplished through the writing of computer programs known as matrix generators. Fourer [7] suggested leaving the translation work to the computer. Such an approach involves computer-readable modeling language that expresses a linear program in much the same way that a modeler does. It is argued that modeling languages should lead to more reliable application of linear programming at lower overall cost. The examples of modeling languages include XML, GAMS, LMC, LPMODEL, UIMP etc..

Output Processing

After solving the formulated problem, the output can be processed to produce relevant reports. Such output processing usually involves converting the problem solution to a form that can be understood by the report writer. The DBMS usually embodies powerful report writing options relieving the user from the worries of report writing. The use of standard report writing options is usually more efficient than ad hoc report writing.

The results from optimization models can also be stored in the company's database for future use. Baker and Shobrys [1] point out that a data base approach allows multilevel problems to be solved

efficiently. An aggregate corporate problem is solved and the results are saved in a database which is then accessed for formulation of divisional problems.

The reports can also be interfaced with graphics. This will improve the overall presentation and understandability of the reports.

Generating Intelligent Reports

Integrating DBMS with optimization can further be augmented by generating intelligent reports. This advantage does not arise from integrating DBMS with optimization but rather is an extension of report writing options in DBMS. Greenberg, Lucas and Mitra [9] are developing an expert system that would take the report content as an input, generate an intelligent report that would contain a complete analysis of the report and then provide requisite advice with necessary documentation. This can prove very helpful where routine decisions are made on the basis of certain rules. Development of such a system requires a knowledge base for decision making. Knowledge base is a set of definite rules and the decisions based on these rules.

CHAPTER IV

COMPONENTS OF THE INTEGRATED SYSTEM

One of the most important considerations in the design of any integrated system to support optimization applications concerns the data handling aspects of the systems. With efficient computing hardware capable of extremely rapid arithmetic operations, linear programming (LP) problems of several thousand rows can be solved economically. Imagine a scenario of the company which has 12 shops with 1500 workers and the company has 100 jobs to be delivered next month. Its mathematical formulation by the user will require an enormous amount of time and effort. Errors may also occur due to the large size problem. In a situation like this, the logical control of the data files and data therein becomes a very important consideration.

It is difficult to control the large LP application at the model level, with data updates being made directly to the LP matrix. Often matrix coefficients involve quite complex computations from more fundamental data. Any lack of control may lead to ultimate breakdown as the model stage departs from the original data base. Moreover, control at the matrix level requires the user to have a high skill and familiarity with LP fundamentals, distracting him from more fundamental task of problem solving and decision making.

In brief, the problems that may arise from allowing the user to have control at the model level with updates being made directly to the

LP matrix are given below:

1. Matrix coefficients usually involve quite complex computations from the fundamental data.
2. Any change made to the LP matrix is not reflected in the fundamental data due to the updates being made directly to the LP matrix.
3. The user gets bogged down in problem formulation and modeling rather than in problem solving and decision making.
4. The user is required to have high level of skill and familiarity with LP fundamentals.

The rest of this Chapter discusses a few important considerations in designing an integrated system of DBMS and optimization.

Data Structure and File Structure

Matrix generation from fundamental data is recognized as an essential ingredient of LP support system. The matrix generator is not sufficient to solve the problem since it must get its data from somewhere. If the data is embodied in the program itself, the problem of controlling the data is not alleviated. So, some form of logically structured data base coupled with a data management component becomes an essential feature of any comprehensive LP support system.

The principal objective of separating data from matrix generation is to permit the user to interface with the system at data level and thus, relieving the user from all the worries of problem formulation, direct updates etc.. This also lends accuracy and reliability to the entire integrated system.

One of the first tasks, then, in building a general system for LP

application is to consider how the data is to be stored and manipulated. The concept of a database implies that the data must follow certain rules and disciplines. The question of how flexible or rigid these rules should be made must be answered at the outset. The following factors must be considered to answer this question:

1. How large is the data base? This will determine how much efficiency can be gained out of a specific data structure.
2. What is the nature of optimization? The optimization technique to be used determines the output needed which in turn determines the structure of the data.
3. How many distinct applications need to be supported? The more the applications, the more complex the data structure would be since it has to satisfy the requirements of different applications.

Data Management and File Management

It was shown in Chapter III that there are many advantages of allowing the user to work at only the data level rather than at the modeling level. Data management and file management are two important concepts in this regard. Data should be maintained as a logical structure. Data represents not only numeric values but also control information that determines the structure of the model to be solved.

The question is how the data should be managed so that they can be maintained and updated with ease and can also aid in efficient problem modeling. What factors affect the programming of a data base manager? These questions must be seriously considered before any system design is initiated.

File management capabilities is another concern for the user. The

user is usually very much concerned about the protection and security of his files and data. These concerns become more accentuated in an integrated environment. Automatic back-up of the databases should be provided for all software. The integrated software should also contain a password feature in which the user is required to provide the correct password before he can access any file or data.

The question of providing proper user interface with the data should also be resolved while designing an integrated software system. The data may be needed by one user or by several, in separately or a shared fashion. The user may need to access more files than necessary for proper data management and file management.

Matrix Generation

In Chapter II, it was indicated that matrix generation process should be automated and thereby providing the user more time for problem solving and decision making. Matrix generation involves extracting the needed data from the database, manipulating the data and producing the formulated problem such that it can be solved by the optimization package.

Matrix Structure and Variable Names

The matrix generator needs a consistent structure of the variables names and also standard formulation techniques. Matrix generators are usually application specific. However there may be some modules which can be shared by different matrix generators. Generation of variables names, for example, can be accomplished by a single module and it can be used by different matrix generators. The matrix structure also

depends on the type of optimization software being used to solve the problem. The matrix structure, for example, shall be different for MPSX from that of LINDO.

Generation of LP variable names is very important due to the constraints imposed by the optimization software and the report writers. Some optimization software requires the variable names to be only 8 characters. The significance of the variable names also arises from the fact that the report writers generate the reports through mapping such names onto the descriptive names. Also the variable names must be unique. In brief, the variable names must be generated with the following considerations:

1. Limitations imposed by the optimization package.
2. Easy and efficient mapping onto the descriptive names by the report writers.
3. Use of descriptive names so that the maintenance personnel can understand it in case of need.
4. Uniqueness of the names.

Maintenance of Matrix Generators

The matrix generator should be designed such that it needs minimal maintenance. One way to accomplish this is to keep the data entirely independent of the matrix generator. In this case, the user needs to make updates only to the database rather than to the matrix generator.

Matrix Display for the User

At times, the user wants to see the display of the matrix to verify its accuracy and reliability. The matrix generator should be

able to produce, at the request of the user, the matrix display in a format meaningful and comprehensible to the user. Should the user desire, the matrix generator should be able to replace the internally generated variable names by their descriptive names to make the display easily understandable.

A Modeling Language

A modeling language is a modeler's form of an optimization problem. A modeling language is taken as an input and a corresponding algorithm's form is produced as an output. Modeler's forms have certain common characteristics [7]. Modeler's forms are symbolic, general, concise and easily understandable. In other words, a modeling language represents data by symbols; defines an entire class of LP together; describes an optimization problem nearly as briefly as possible; and presents an optimization problem in a form that is easily read and comprehended by people.

Selection of a particular modeling language depends on the needs of users and the nature of computers. This means that a modeling language should be easy to use and understand, and could be processed and translated at a reasonable cost by the computer.

Report Writing

It was mentioned earlier that the user should devote more time to decision making rather than problem-modeling. The decision-making process is based on the results obtained from the integrated software. Such result generation activity is technically called report writing since it follow from a series of activities and usually done at the

end. Report writing can be interpreted as a means of restructuring the basic solution values produced by the optimization algorithms into a form that relates back to the real world problem being modeled.

Following is a list of factors that should be considered for report writing:

1. Restructuring of the basic solution values tends to vary from one application to another. In other words, report writing is very application-specific.
2. When the problem is modeled, a number of variables are generated internally. When report writing is done, these variables are translated into the actual names so that the user can understand. This phenomenon is also called dictionary preparation.
3. Proper headings, titles and stubs must be generated.
4. Proper units must be appended to the values being reported. The units may be hours, thousands of dollars etc.. Totals and subtotals should be done wherever required.
5. Report writing should be completely menu-driven. The user should be allowed to select an option from the menu for which he wants the report.
6. Sometimes, the user may be interested in maintaining a separate database where a subset of the solution can be placed and be used for other purposes. In other words, the integrated software should be efficient enough to extract the desired subset of a solution and place it wherever the user wants so that the solution values themselves can be used for modeling or other purposes.
7. Multicase reporting is also an important consideration. This involves extraction of a single report solution values from more

than one solution case. This may be needed where multiple periods are considered and solution is generated periodically. Hence the integrated software should be able to access multiple solution cases and generate one desired report.

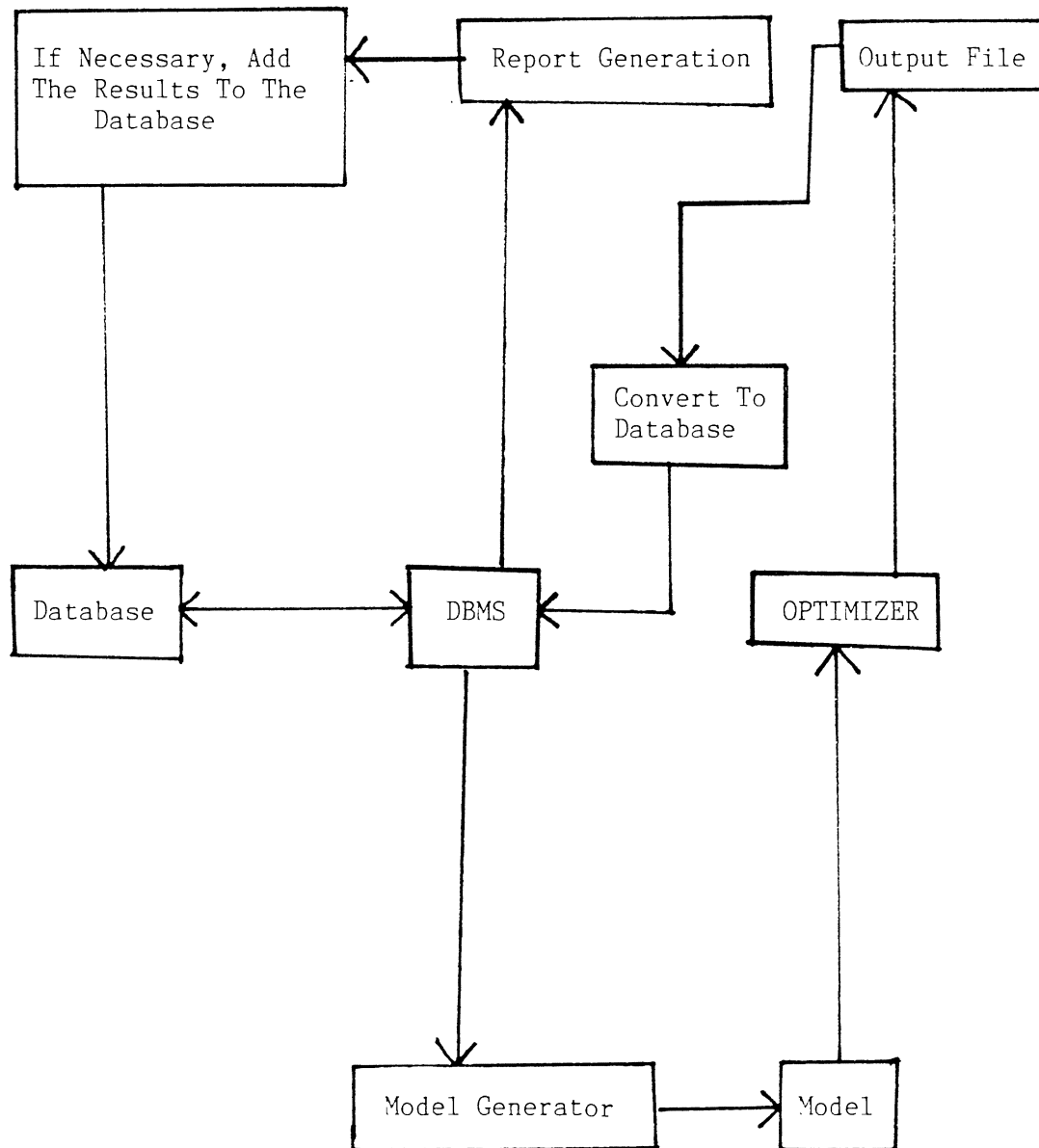
8. The integrated software should provide for printing the reports in a desirable format. The user may want whole or a part of the report to be printed. The user may also want different parts of the report to be printed in different styles and formats.

A Sketch of DBMS-based LP System

Figure 1 displays a schematic of the system showing how DBMS interfaces with an optimizer. All data relevant to model building are maintained by the DBMS in the database.

The model generator accesses the data from the database and formulates the model. The model generator may consist of two components: first, a modeling language; and second, command files. It should be pointed out that the example given in Chapter V does not yet include any modeling language. The command files are used to generate a specific LP problem. The model generator performs all arithmetic and symbolic operations. Arithmetic operations may include computation of coefficients or manipulations of data. The standard structure of the DBMS facilitates the model generator in model formulation.

Such a model is input into an optimizer. The results obtained are placed in a data base file. The format of the file containing results is usually not acceptable to DBMS. In such a situation, it needs to be converted into a proper format. The structure of the results produced by the optimizer is usually not in a convenient form. Hence a proper



Source: Sharda, Turner and Rathi [16]

Figure 1. Integration of DBMS and Optimization Program

interface may be provided.

The resulting data base file is used for report generation. The results, at user's option, can also be added to the database. This allows the company to use the results for other purposes e.g. generating graphs, formulating another problem etc..

Summary

Data structure and file structure should be selected very carefully for better efficiency and memory utilization. Data management and file management resolve the questions pertaining to the degree of sharability and security of data base files, and user interface with the integrated system. A matrix generator should be written with a focus on proper matrix structure and variables generation. Matrix generators are application specific. Hence the requirements of the optimization package decide the matrix structure and variables generation. Matrix generators should need minimal maintenance. A modeling language could be used to produce an algorithm's form of the problem. A modeling language is symbolic, general, concise and easily understandable. However, processing and translation of a modeling language should be done at a reasonable cost. Report writing is crucial to any integrated environment. The generated reports should be meaningful and easy to understand. Proper headings, titles, stubs, units of measurements should be produced. Variable names should be translated into a descriptive form. Reports writers should be endowed with arithmetic and symbolic translation capabilities. A proper interface between the report writer and the user is also important.

CHAPTER V

AN EXAMPLE ON JOB-SHOP PROBLEM

Problem Definition

A manufacturer has n jobs which must be completed this week and each job may be handled in any one or more of the m shops. The manufacturer wants to know how to allocate shop time to different jobs in order to minimize the total cost of completing all jobs. The manufacturer maintains four data base files which are shown in Figures 2, 4, 6, and 8. These data base files have been developed using the data base package dBASE II which also contains a data processing language. Figures 3, 5, 7, and 9 contain the structures of these four data base files respectively.

Figure 2 displays ZEMPLYEE.DBF which contains employee records as to their total weekly working hours, the hourly wage rate, work status etc.. Figure 4 shows ZJOBORDR.DBF which keeps track of all the job records that need to be completed during the week. It also has the job category, the date on which the job is received, and the expected date of its completion. Figure 6 contains ZHRSHIST.DBF. This file has the historical data on how many hours a job of a certain category would take to complete. This is updated periodically. Figure 8 shows ZCOSTREC.DBF which contains only 2 records. The first record relates to the raw material cost and the second record relates to the overhead expenses during that week. Actually, ZCOSTREC.DBF is prepared using

00001	AUSTIN,KEITH	PERM	SHOP2	31	40
00002	GRACEM,W.	PERM	SHOP1	33	40
00003	BAYLOR,CHRIS	PERM	SHOP3	29	40
00004	HART,BRYAN	PERM	SHOP1	30	25
00005	MOSES,DAVID	TEMP	SHOP3	31	40
00006	NORRIS,CHERRY	PERM	SHOP3	28	20
00007	UTTERBACK,ORLEY	TEMP	SHOP2	30	40
00008	CHURCHILL,V.	PERM	SHOP1	29	30
00009	WILLIAMS,S.	PERM	SHOP2	32	40
00010	HWAN,C.	PERM	SHOP3	33	40
00011	AMOS,TOM	PERM	SHOP2	29	40
00012	COLUMBUS,TIM	PERM	SHOP1	30	40
00013	WILLINGTON,K.	PERM	SHOP1	28	40

Figure 2. Employee Records (ZEMPLYEE.DBF)

```

Structure for file: B:ZEMPLYEE.DBF
Number of records: 00013
Date of last update: 01/01/80
Primary use database
Fld      Name      Type Width  Dec
001      EMPNAME   C      015
002      TYPE      C       004
003      SHOPNAME  C       008
004      HRLYRT    N       005
005      WKHRS     N       002
** Total **                00035

```

Figure 3. Structure of ZEMPLYEE.DBF

00001	JOB1	A	APR 3,1986	APR 12,1986
00002	JOB2	B	APR 4,1986	APR 12,1986
00003	JOB3	C	APR 4,1986	APR 12,1986
00004	JOB4	D	APR 5,1986	APR 12,1986

Figure 4. Pending Job Orders (ZJOBORDR.DBF)

```

Structure for file: B:ZJOBORDR.DBF
Number of records: 00004
Date of last update: 01/08/87
Primary use database
Fld      Name              Type Width  Dec
001      JOBNAME           C      008
002      CATGRY             C      006
003      DATERD              C      012
004      DATECMP            C      012
** Total **                00039

```

Figure 5. Structure of ZJOBORDR.DBF

```

00001  A      32      39      46
00002  B     151     147     155
00003  C      72      61      57
00004  D     118     126     121
00005  E      85      93      86
00006  F     102      94     100

```

Figure 6. Job Categories and Hours to Complete Them in Different Shops (ZHRSHIST.DBF)

```

Structure for file: B:ZHRSHIST.DBF
Number of records: 00006
Date of last update: 01/08/87
Primary use database
Fld      Name              Type Width  Dec
001      CATEGORY           C      001
002      SHOP1              N      005
003      SHOP2              N      005
004      SHOP3              N      005
** Total **                00017

```

Figure 7. Structure of ZHRSHIST.DBF

00001	RAW. COST	43	48	36
00002	OVERHD/DAY	2000	1800	2000

Figure 8. Cost Records (ZCOSTREC.DBF)

```

Structure for file: B:ZCOSTREC.DBF
Number of records: 00002
Date of last update: 01/01/80
Primary use database
Fld      Name          Type Width  Dec
001      HEAD             C      010
002      SHOP1            N       005
003      SHOP2            N       005
004      SHOP3            N       005
** Total **                00026

```

Figure 9. Structure of ZCOSTREC.DBF

other files which are not shown here.

Problem Analysis

Now the problem will be analyzed in the light of data base management and optimization. The key feature to be noted here is that the information needed to formulate an LP problem is not directly available in the above problem definition. Only the data base files are provided and all the computations necessary for problem formulation have been left to the modeler which resides in the integrated package. The data base processing language of dBASE II shall be used for arithmetic computation, data extraction and manipulation, LP problem generation and report writing. LINDO has been used as an optimizer to solve the LP problem. The complete procedure, from data preparation to report writing, has been described in the next section.

Brief Overview of the Integrated System in This Example

Figure 10 shows a flow chart illustrating how dBASE II has been integrated with LINDO. As stated earlier, Data are scattered across the entire data base in different files. The first task, therefore, is to extract relevant data from different files and perform arithmetic calculations on them. This task corresponds to the input preparation stage as shown in the flow chart. Second stage is variables generation. The variables are in symbolic form and used by LINDO in LP problem formulation. A translation dictionary of such variables is maintained. Matrix generation stage comes next. The LP problem is formulated in a format acceptable to LINDO. In optimization stage, the formulated LP problem is fed into an optimizer LINDO. Optimal values

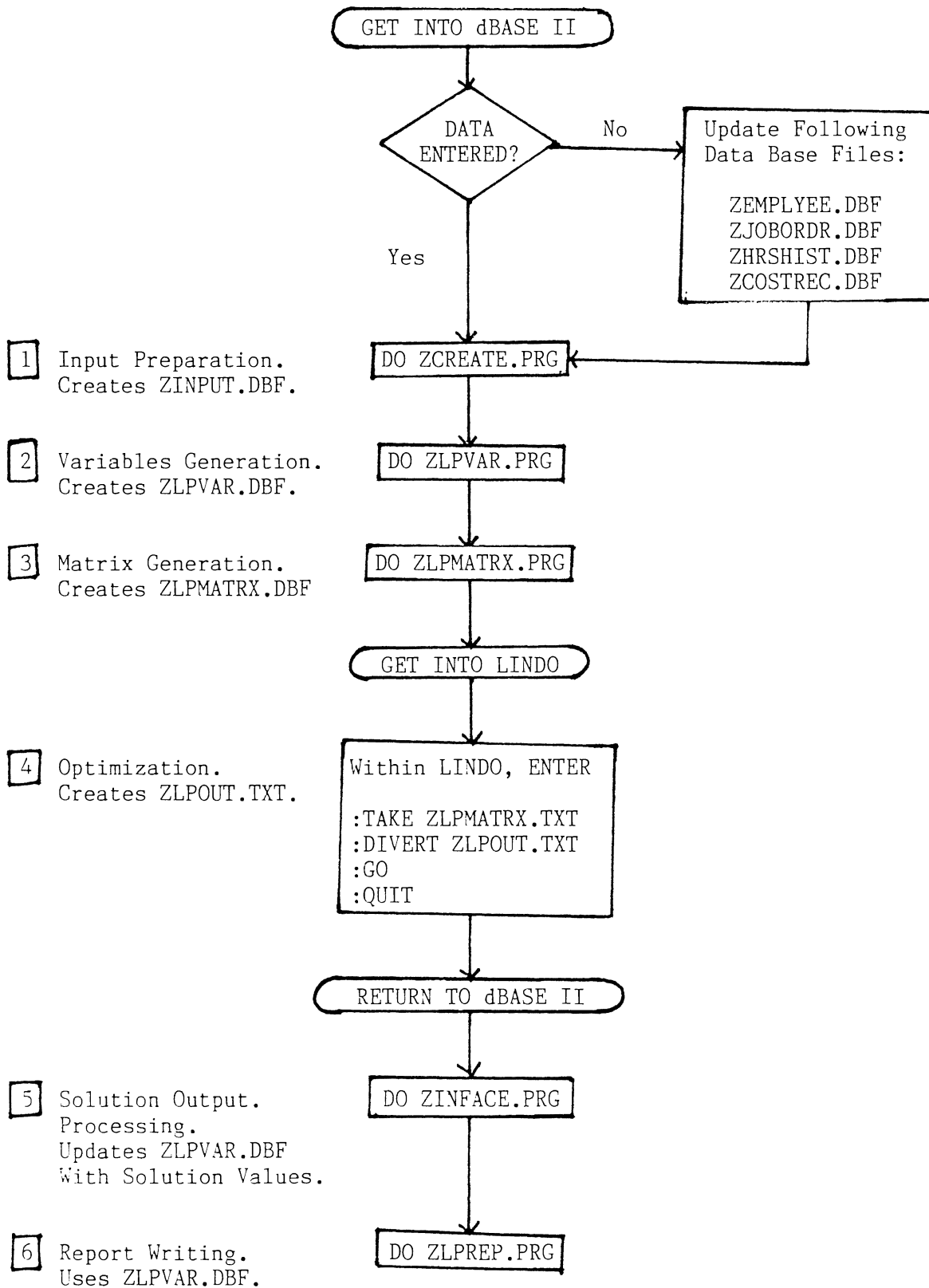


Figure 10. Flow Chart of Job Shop Problem Illustrating Integration of dBASE II with LINDO

are generated and transferred to dBASE II. Then the solution values are processed and stored in a data base file. Finally, report generation takes place. Report writing options allow the user to generate a wide variety of reports. Report writing is completely menu driven.

All the stages mentioned above are controlled by a single module. User's interaction was made as convenient as possible. A detailed discussion of such an integrated system is given in the next section.

Detailed Description of the Example

Input Preparation

Data are scattered across the entire database in different files. In a real life situation, there will be more files and bigger data size in the data base. The basic idea, however, is to use multiple files such that the desired LP problem can be formulated, solved and the results can be reported.

The first step, therefore, is to gather the relevant data scattered across different files and place them in a separate data base file or files which can be conveniently accessed by other modules. In this example, this task is executed by the command file ZCREATE.PRG. The listing of ZCREATE.PRG is given in the Appendix. ZCREATE.PRG accesses ZEMPLYEE.DBF the contents of which are given in the problem definition. ZCREATE.PRG accesses three fields: Shopname, Hrlyrate and Wkhrs for each employee record to compute total number of hours available in and the total wages of each shop.

ZCREATE.PRG then extracts the raw material cost and the overhead expenses from the file ZCOSTREC.DBF. The raw material cost, overhead

expenses and wages are summed up to ascertain the cost/man hour in each shop.

ZHRSHIST.DBF contains the historical data as to the number of hours that each distinct job category requires to be completed. ZJOBORDR.DBF contains a list of the jobs that need to be completed. Each job has a category which is decided according to the number of hours it will take to complete in different shops. First, ZCREATE.PRG accesses ZJOBORDR.DBF to find the job category for each job. Then it extracts from ZHRSHIST.DBF the number of hours corresponding to that job category. This arrangement allows the user to update the historical information and add more job categories as the need arises. Also, ZJOBORDR.DBF may contain any number of job orders.

In brief, ZCREATE.PRG prepares the necessary data from the above mentioned data files and places such data in the file called ZINPUT.DBF. The contents and the structure of ZINPUT.DBF are listed in Figure 11 & Figure 12 respectively. Records 1 and 2 of Figure 11 are number of hours and cost/man hour respectively. All other records except for the last record show the number of hours that a particular job will take in a particular shop. Second and third columns of the last record are the number of shops and the number of jobs respectively.

Variables Generation

This step involves generation of internal variable names. The variable names are in the form of X_{ij} which denotes number of hours of JOB_j in $SHOP_i$. In this example, the number of digits for i and j were restricted to two. It is assumed that number of shops and jobs will

00001		175	160	140
00002		84	89	80
00003	JOB1	32	39	46
00004	JOB2	151	147	155
00005	JOB3	72	61	57
00006	JOB4	118	126	121
00007		3	4	0

.

Figure 11. Input Prepared from the Data Base (ZINPUT.DBF)

```

Structure for file: B:ZINPUT .DBF
Number of records: 00007
Date of last update: 01/01/80
Primary use database
Fld      Name          Type Width  Dec
001      JOBS             C    008
002      SHOP1            N    006
003      SHOP2            N    006
004      SHOP3            N    006
** Total **                00027

```

Figure 12. Structure of ZINPUT.DBF

not exceed 99. The command file ZLPVAR.PRG is executed to generate the variables. This command file accesses the last record of ZINPUT.DBF to extract number of shops and number of jobs. Total number of variables created equals the number of shops times the number of jobs.

ZLPVAR.PRG also extracts Shopnames, Jobnames and cost/man hour. All these data are stored in a file called ZLPVAR.DBF. The contents and the structure of ZLPVAR.DBF are shown in Figure 13 and Figure 14 respectively. ZLPVAR.DBF serves as a dictionary i.e. it contains definitions of the variables generated internally. First column contains variable names and the next two columns define those variables.

First column of Figure 13 contains the internally created variables. Column 4 of this figure has zeros indicating blanks. This is because dBASE II automatically fills the blanks of the numeric fields with zeros. Column 5 contains the cost/man hour in SHOP_i. The listing of ZLPVAR.PRG is given in APPENDIX.

Matrix Generation

Matrix generation follows variables creation. It was indicated earlier that mostly the matrix generators are application specific. In this case, our application is related to the job-shop LP formulation. In Chapter IV, it was mentioned that the limitations imposed by the optimization package should be considered while developing a matrix generator. The optimization package that this example uses is LINDO. Hence, the matrix generator named ZLPMATRIX.PRG generates the LP problem in a format acceptable to LINDO.

ZLPMATRIX.PRG first generates the objective function using the

00001	X0101	JOB1	SHOP1	0.000	84
00002	X0102	JOB2	SHOP1	0.000	84
00003	X0103	JOB3	SHOP1	0.000	84
00004	X0104	JOB4	SHOP1	0.000	84
00005	X0201	JOB1	SHOP2	0.000	89
00006	X0202	JOB2	SHOP2	0.000	89
00007	X0203	JOB3	SHOP2	0.000	89
00008	X0204	JOB4	SHOP2	0.000	89
00009	X0301	JOB1	SHOP3	0.000	80
00010	X0302	JOB2	SHOP3	0.000	80
00011	X0303	JOB3	SHOP3	0.000	80
00012	X0304	JOB4	SHOP3	0.000	80
00013	OBJECT			0.000	0

Figure 13. LP Variables in a Dictionary Form (ZLPVAR.DBF)

```

Structure for file: B:ZLPVAR .DBF
Number of records: 00013
Date of last update: 01/01/80
Primary use database
Fld      Name           Type Width  Dec
001      XVAR              C    006
002      XJOB              C    008
003      XSHOP            C    008
004      XVALUES          N    010   003
005      XCOST            N    005
** Total **                00038

```

Figure 14. Structure of ZLPVAR.DBF

files ZINPUT.DBF and ZLPVAR.DBF. Internal variables stored in ZLPVAR.DBF are prefixed with the cost/man hour (coefficients) contained in ZINPUT.DBF. Each pair consisting of a coefficient and a variable name occupies one record of the file called ZLPMATRIX.DBF which stores the LP formulation. Constraints are also generated using ZINPUT.DBF and ZLPVAR.DBF. Two sets of constraints are generated by ZLPMATRIX.PRG. One set relates to the total number of hours available in SHOP_i and other set relates to ensuring that the JOB_j is completed. '+', '<=', '=' signs are generated internally by the command file. The entire LP formulation is stored in ZLPMATRIX.DBF. Finally, the ASCII version of ZLPMATRIX.DBF is created and stored in ZLPMATRIX.TXT.

Figure 15 and Figure 16 show the contents and the structure of ZLPMATRIX.DBF respectively. In Figure 15, the zeros in front of the variable names, 'MINIMIZE', 'SUBJECT TO' and 'END' are really blanks rather than zeros. The contents of ZLPMATRIX.TXT are displayed in Figure 17. Figure 17 is actually the ASCII version of ZLPMATRIX.DBF. In this figure, all the blanks are shown as blanks rather than as zeros.

Optimization

At this stage, the control is passed to the optimization package LINDO. Within LINDO, the following sequence of commands is used:

```
: TAKE ZLPMATRIX.TXT
: DIVERT ZLPOUT.TXT
: GO
<Objective function will appear here>
<Respond 'N' to the next question>
```

```

00001      0.000000  MINIMIZE
00002      84.000000  X0101  +
00003      84.000000  X0102  +
00004      84.000000  X0103  +
00005      84.000000  X0104  +
00006      89.000000  X0201  +
00007      89.000000  X0202  +
00008      89.000000  X0203  +
00009      89.000000  X0204  +
00010      80.000000  X0301  +
00011      80.000000  X0302  +
00012      80.000000  X0303  +
00013      80.000000  X0304
00014      0.000000  SUBJECT TO
00015      0.000000  X0101  +
00016      0.000000  X0102  +
00017      0.000000  X0103  +
00018      0.000000  X0104  <=
00019      175.000000
00020      0.000000  X0201  +
00021      0.000000  X0202  +
00022      0.000000  X0203  +
00023      0.000000  X0204  <=
00024      160.000000
00025      0.000000  X0301  +
00026      0.000000  X0302  +
00027      0.000000  X0303  +
00028      0.000000  X0304  <=
00029      140.000000
00030      0.031250  X0101  +
00031      0.025641  X0201  +
00032      0.021739  X0301  = 1
00033      0.006622  X0102  +
00034      0.006802  X0202  +
00035      0.006451  X0302  = 1
00036      0.013888  X0103  +
00037      0.016393  X0203  +
00038      0.017543  X0303  = 1
00039      0.008474  X0104  +
00040      0.007936  X0204  +
00041      0.008264  X0304  = 1
00042      0.000000  END

```

Figure 15. LP Problem in Data Base
Format (ZLPMATR.X.DBF)

```

Structure for file: B:ZLPMATRIX.DBF
Number of records: 00042
Date of last update: 01/01/80
Primary use database
Fld      Name      Type Width  Dec
001     COLL      N      010    006
002     MATVAR     C      010
** Total **           00021

```

Figure 16. Structure of ZLPMATRIX.DBF

```

MINIMIZE
84.000000X0101 +
84.000000X0102 +
84.000000X0103 +
84.000000X0104 +
89.000000X0201 +
89.000000X0202 +
89.000000X0203 +
89.000000X0204 +
80.000000X0301 +
80.000000X0302 +
80.000000X0303 +
80.000000X0304
SUBJECT TO
X0101 +
X0102 +
X0103 +
X0104 <=
175.000000
X0201 +
X0202 +
X0203 +
X0204 <=
160.000000
X0301 +
X0302 +
X0303 +
X0304 <=
140.000000
0.031250X0101 +
0.025641X0201 +
0.021739X0301 = 1
0.006622X0102 +
0.006802X0202 +
0.006451X0302 = 1
0.013888X0103 +
0.016393X0203 +
0.017543X0303 = 1
0.008474X0104 +
0.007936X0204 +
0.008264X0304 = 1
END

```

Figure 17. LP Problem in ASCII
Format (ZLPMATRIX.TXT)

: QUIT

The TAKE command within LINDO allows the user to get the problem in ASCII format and restructures it to the standard LINDO format. The restructured LP problem is shown in Figure 18. The DIVERT command will allow the user to divert the output to a file called ZLPOUT.TXT. The output consists of the optimal values of the objective function and variables, slack and surplus values, reduced costs, dual prices and number of iterations. The GO command solves the LP problem. The solution is contained in Figure 19. The QUIT command lets the control go to the DOS level.

Solution Output Processing

The LP solution contained in Figure 19 was generated by LINDO. It has no significance if the user cannot comprehend and interpret it. Let us assume that the user does not know how to read the LINDO output. Also the user may not remember which variable denotes what and how the numbers are coherently related to each other. To sum up, the task of understanding and interpreting the output straightway is quite tedious and cumbersome and may sometimes lead to erroneous conclusions. To make the user's task easy, the command file ZINFACE.PRG provides a back-end to the solution and processes the output. ZINFACE.PRG really serves as an interface between the LINDO solution and the report writing. It extracts the objective function value and all the optimal values of the variables from the solution contained in ZLPOUT.TXT and then places them in the file ZLPVAR.DBF. As was indicated in the section of variables generation, the second last column of ZLPVAR.DBF was filled with zeros for blanks by dBASE II. These all the blanks are

```

MIN      84 X0101 + 84 X0102 + 84 X0103 + 84 X0104 + 89 X0201 + 89 X0202
+ 89 X0203 + 89 X0204 + 80 X0301 + 80 X0302 + 80 X0303 + 80 X0304
SUBJECT TO
2)      X0101 + X0102 + X0103 + X0104 <= 175
3)      X0201 + X0202 + X0203 + X0204 <= 160
4)      X0301 + X0302 + X0303 + X0304 <= 140
5)      .03125 X0101 + .025641 X0201 + .021739 X0301 = 1
6)      .006622 X0102 + .006802 X0202 + .006451 X0302 = 1
7)      .013888 X0103 + .016393 X0203 + .017543 X0303 = 1
8)      .008474 X0104 + .007936 X0204 + .008264 X0304 = 1
END

```

Figure 18. LP Problem as Restructured by LINDO

```

LP OPTIMUM FOUND AT STEP 9

OBJECTIVE FUNCTION VALUE

1)      29805.9000

VARIABLE      VALUE      REDUCED COST
X0101          32.000000      .000000
X0102        105.932400      .000000
X0103           .000000      19.751880
X0104         37.067630      .000000
X0201           .000000      17.906890
X0202         43.886480      .000000
X0203           .000000      10.041480
X0204           .000000       7.856125
X0301           .000000      24.223310
X0302           .000000       .090225
X0303         57.002800      .000000
X0304         82.997210      .000000

      ROW      SLACK OR SURPLUS      DUAL PRICES
      2)           .000000           2.644814
      3)        116.113500           .000000
      4)           .000000           4.497604
      5)           .000000       -2772.634000
--More--
      6)           .000000       -13084.390000
      7)           .000000       -4816.600000
      8)           .000000       -10224.780000

NO. ITERATIONS= 9

```

Figure 19. LP Solution Generated by LINDO

now replaced by the respective optimal values. The objective function value reflects minimum cost of completing all jobs and the variables' values reflecting optimal number of hours to be spent on each job in each shop. ZINFACE.PRG makes use of ZLPOUT.DBF, a data base version of an ASCII file ZLPOUT.TXT.

Figure 20 shows the file ZLPVAR.DBF. The only difference between Figure 13 and Figure 20 lies in the second last column which now contains solution values instead of zeros.

Report Writing

Finally the control passes to the report writer. With the integrated system in place, the user is directly concerned with: first, entering and updating the data; second, generating and studying the relevant reports for decision making. This second step, therefore, largely depends on the judgement of the user in that the decision making shall be influenced by the type of report opted.

dBASE II contains a report writing option which allows the user to generate customized reports. In this example, the command file ZLPREP.PRG contains the report generator module. Report generation is completely menu-driven. Six different options are available to the user to select. What the user views on the screen while generating a report is shown in Figure 21. The report generator ZLPREP.PRG, in fact, uses 5 different report formats for the first five options. All these report formats are stored in five different report format files ending with .FRM.

ZLPVAR.DBF is accessed and the relevant values are extracted and, if needed, these values are manipulated. Proper units and formats are

00001	X0101	JOB1	SHOP1	32.000	84
00002	X0102	JOB2	SHOP1	105.932	84
00003	X0103	JOB3	SHOP1	0.000	84
00004	X0104	JOB4	SHOP1	37.067	84
00005	X0201	JOB1	SHOP2	0.000	89
00006	X0202	JOB2	SHOP2	43.886	89
00007	X0203	JOB3	SHOP2	0.000	89
00008	X0204	JOB4	SHOP2	0.000	89
00009	X0301	JOB1	SHOP3	0.000	80
00010	X0302	JOB2	SHOP3	0.000	80
00011	X0303	JOB3	SHOP3	57.002	80
00012	X0304	JOB4	SHOP3	82.997	80
00013	OBJECT			29805.900	0

Figure 20. LP Variables and their Solution Values

REPORT GENERATOR FOR JOB-SHOP PROBLEM

```

*****
*   Please SELECT one of the following options :-           *
*                                                           *
* 1 Optimal value of the objective function.                 *
*                                                           *
* 2 Optimal(Minimal) # of hours & cost of all jobs.         *
*                                                           *
* 3 Optimal(Minimal) # of hours and cost of a particular job *
*   in a particular shop.                                    *
*                                                           *
* 4 Optimal(Minimal) # of hours & cost of a particular job  *
*   in all shops.                                           *
*                                                           *
* 5 Optimal(Minimal) # of hours & cost of a particular shop *
*   for all jobs.                                           *
*                                                           *
* 6 Quit to Main Menu.                                       *
*****

```

Enter your option (1/2/3/4/5/6) :

Figure 21. Menu for Report Generation

generated by the respective report format files. Descriptive names of the variables are contained in ZLPVAR.DBF. The last option lets the user quit from the report writer. This example does not provide "PRINT" option. But this can be easily done by routing the output to the printer rather than to the screen. "SET PRINT ON" command can be used to accomplish that. Also, for the sake of simplicity, this example avoids inserting the results back into the company's database for future use.

Figure 22 shows the screen automatically generated by dBASE II on requesting for report generation for option 3. Figure 23 displays the contents, as internally stored by dBASE II for this option. Contents of this figure are really the control elements of report generation i.e. they control output of the report. For example, the first line of that figure contains 'Y' which indicates that the user wants the heading to be placed in the report and the heading is contained in the second line. Options as to totals and subtotals are also interpreted the same way. Such internally stored report options play a very useful role in report generation. Report generation becomes very powerful and flexible with these options.

Figure 24 shows a report generated by dBASE II for option 2. Option 2 is related to minimum no. of hours and cost of all jobs. The values are extracted from ZLPVAR.DBF. All totals, subtotals, titles and subtitles, and units have been generated by the report generator. All options given in Figure 22 are included in this report. Page number and date are inserted by the report generator automatically.

```

ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH
PAGE HEADING? (Y/N) Y
ENTER PAGE HEADING: ** MINIMUM NO. OF HOURS FOR ALL JOBS IN ALL SHOPS **
DOUBLE SPACE REPORT? (Y/N) N
ARE TOTALS REQUIRED? (Y/N) Y
SUBTOTALS IN REPORT? (Y/N) Y
ENTER SUBTOTALS FIELD: XSHOP
SUMMARY REPORT ONLY? (Y/N) N
EJECT PAGE AFTER SUBTOTALS? (Y/N) N
ENTER SUBTOTAL HEADING: In
COL    WIDTH,CONTENTS
001    30,xjob
ENTER HEADING: Jobname      ;=====
002    30,xvalues
ENTER HEADING:      No. of Hours;      =====
ARE TOTALS REQUIRED? (Y/N) y
003

```

Figure 22. Report Format Generation in dBASE II

```

Y
** MINIMUM NO. OF HOURS FOR ALL JOBS IN ALL SHOPS **
N
Y
Y
XSHOP
N
N
In
30,xjob
Jobname      ;=====
30,xvalues
      No. of Hours;      =====
v

```

Figure 23. Report Options as Internally Stored by dBASE II

PAGE NO. 00001
01/01/80

** MINIMUM NO. OF HOURS FOR ALL JOBS IN ALL SHOPS **

<u>Jobname</u> =====	<u>No. of Hours</u> =====
* In SHOP1	
JOB1	32.000
JOB2	105.932
JOB3	0.000
JOB4	37.067
** SUBTOTAL **	174.999
* In SHOP2	
JOB1	0.000
JOB2	43.886
JOB3	0.000
JOB4	0.000
** SUBTOTAL **	43.886
* In SHOP3	
JOB1	0.000
JOB2	0.000
JOB3	57.002
JOB4	82.997
** SUBTOTAL **	139.999
** TOTAL **	358.884

Figure 24. Report As Generated by dBASE II for Option 2

CHAPTER VI

SUMMARY AND CONCLUSIONS

The integration of DBMS with optimization offers substantial advantages. With the DBMS in use, the modeler can spend more time on the design of the model. Most of the models are more or less application specific. With the rapidly growing network technologies, the sharing of data files is becoming easier and more reliable. Multiple users can work on the same data base.

The managerial decision making would be improved with timely reports. Reports produced by such an integrated system are usually more accurate and reliable. Such reports are standardized and can be generated easily and efficiently.

Exxon Corporation is already using a very powerful integrated system for planning called PLATOFORM (PLanning TOol developed in the dataFORM language of MPS II). See Palmer [13] for a complete description of PLATOFORM.

Greenberg, Lucas and Mitra [9] went one step further and proposed to augment the reports with proper analysis and documentation. They came up with an idea of appending to the report generator an expert system (as a back-end) called ANALYZE. The ANALYZE will analyze the output of the report generator, give necessary advice and produce the meaningful documentation. It should, however, be noted that this is not a substitute for the manager but rather an aid.

Future of DBMS-based optimization seems very bright. Substantial efforts are geared towards making the optimization techniques available to people who consider themselves unsophisticated users. This will also increase the depth of use of the current problems. A significant amount of research is underway in the field of modeling languages. In future, matrix generators will probably become obsolete due to their being in algorithm's form which lacks understandability and user friendliness. Display capabilities and user interface are also gaining importance. Expert systems are also being developed to analyze the reports.

A SELECTED BIBLIOGRAPHY

1. Baker, T.E., D.E. Shobryns. "The Integration of Planning, Scheduling and Control." National Petroleum Refiners Association, Computer Conference:CC-85-97.
2. Bisschop, J., A. Meerus. "On the Development of a General Algebraic Modeling System in a Strategic Planning." Mathematical Programming Study, 20:1-29.
3. Boyer, P.A. "Fitting Micros to Manufacturing Control Systems." Infosystems, Sep. 1983:opl4-opl7.
4. Brown, R.W., W.D. Northup, J.F. Shapiro. "LOGS: A Modeling and Optimization System for Business Planning." Computer Assisted Decision Making, 1986:227-241.
5. Chesapeake Decision Sciences, Inc. "The Integration of Planning, Scheduling, and Control in the Process Industry." 1986.
6. Ellison, E.F.D., G. Mitra. "UIMP: User Interface for Mathematical Programming." ACM Transactions on Mathematical Software, 8(3):229-255.
7. Fourer, R. "Modeling Languages versus Matrix Generators for Linear Programming." ACM Transactions on Mathematical Software, 9(2):143-183.
8. Graduate College. "Thesis Writing Manual." Stillwater, Oklahoma, Oklahoma State University, Revised, 1972.
9. Greenberg, H.J., C. Lucas, G. Mitra. "Computer Assisted Modelling and Analysis of Linear Programming Problems: Towards a Unified Framework." Working Paper, Brunel University, U.K..
10. Kisko, T.M. "A Syntax for Linear Program That Computers and People Can Understand." Industrial & Systems Engineering Department, University of Florida, no. 86-13 (1986).
11. Krabek, C.B., R.J. Sjoquist, D.C. Sommer. "The Apex Systems: Past and Future." ACM SIGMAP Bulletin, 29 (April,1980): 3-23.
12. Lucas, C., G. Mitra. "Computer Assisted Mathematical Programming (Modelling) System: CAMPS." Working Paper, Brunel University, U.K..

13. Palmer, K.H. "A Model-Management Framework for Mathematical Programming." An Exxon Monograph, 1984.
14. Rowland, A.J., Boudwin, N.K. "A Data and Model Management System in Exxon." 1986:253-259.
15. Sharda, R. "Optimization Using Spreadsheets on a Microcomputer." Annals of Operation Research, 5:594-612.
16. Sharda, R., S. Turner, A. Rathi. "Management of Mathematical Programming Models Using Data Base Management System." Proceedings on North East Decision Science Institute, Atlanta, April 1987 (Forthcoming).
17. Sweet, F. "What, If Anything Is a Relational Database ?." Datamation, July 15, 1984:118-124.
18. Welke, L. "Advent of the Clustered System." Datamation, May 1, 1985:77-82.

APPENDIX

```

*****
* NAME OF COMMAND FILE: zcreate.prg *
* *
* PURPOSE: THIS COMMAND FILE PREPARES THE INPUT NEEDED *
* FOR MODELING THE LINEAR PROGRAMMING PROBLEM.*
* THIS FILE EXTRACTS RELEVANT INFORMATION *
* FROM FOUR FILES: EMPLOYEES' FILE (zemployee.*
* dbf), COST RECORDS FILE (zcostrec.dbf), *
* HISTORY FILE (zhrshist.dbf) AND JOB ORDERS *
* FILE (zjobordr.dbf). zhrshist.dbf HAS NO. OF*
* HOURS REQUIRED TO COMPLETE ANY SPECIFIC *
* CATEGORY OF JOB IN DIFFERENT SHOPS. *
*****
*
*
SET TALK OFF
USE zemployee
GOTO BOTTOM
*
* INITIALIZING THE VARIABLES AND COUNTERS.
*
STORE # TO totemp
STORE 3 TO totshop
STORE 3 TO tempk
STORE 1 TO i
DO WHILE i <= totshop
  STORE STR(i,1) TO m
  STORE 0 TO shopemp&m
  STORE 0 TO scost&m
  STORE 0 TO shrs&m
  STORE i+1 TO i
ENDDO
*
* COMPUTING TOTAL NO. OF HOURS AVAILABLE IN SHOPi AND
* ITS COST/MAN HOUR. THESE ARE BASED ON WEEKLY RECORDS.
*
*
STORE 1 TO i
DO WHILE i <= totemp
  STORE STR(i,1) TO m
  GOTO i
  STORE 0 TO j
  STORE 'F' TO find
  DO WHILE (j+1 <= totshop .AND. find = 'F')
    STORE j+1 TO j
    STORE STR(j,1) TO n
    IF shopname='SHOP&n'
      STORE 'T' TO find
    ENDIF
  ENDDO
  STORE scost&n+(hrlyrt*wkhrs) TO scost&n
  STORE shrs&n+wkhrs TO shrs&n

```

```

        STORE shopemp&n+1 TO shopemp&n
        STORE i+1 TO i
ENDDO
*
*
* COMPUTING COST/MAN HOUR IN SHOPi INCLUDING WAGES,
* RAW MATERIAL COST AND OVERHEAD ON A WEEKLY BASIS.
*
*
STORE 1 TO i
USE zinput
DELETE ALL
PACK
APPEND BLANK
APPEND BLANK
DO WHILE i <= totshop
    STORE STR(i,1) TO m
    USE zcostrec
    GOTO 1
    STORE shop&m TO rawcost
    STORE (shrs&m*rawcost + scost&m) TO scost&m
    GOTO 2
    STORE scost&m+shop&m TO scost&m
    USE zinput
    GOTO 1
    REPLACE shop&m WITH shrs&m
    GOTO 2
    REPLACE shop&m WITH INT(scost&m/shrs&m)
    STORE i+1 TO i
ENDDO
*
*
* MATCHING THE JOB CATEGORY WITH THE STANDARD
* CATEGORIES CONTAINED IN zhrshist.dbf AND DERIVING
* THE NO. OF HOURS THEREFROM FOR COMPLETING THE JOB
* BASED ON ITS CATEGORY.
*
STORE 1 TO i
USE zjobodr
GOTO BOTTOM
STORE # TO totjob
STORE 2 TO tempk
DO WHILE i <= totjob
    USE zjobodr
    GOTO i
    STORE catgry TO tempcat
    STORE jobname TO jobn
    USE zhrshist
    STORE 1 TO j
*
* FINDING THE JOB CATEGORY.
*

```

```

DO WHILE tempcat<>category
  STORE j+1 TO j
  GOTO j
ENDDO
*
STORE 1 TO k
USE zhrshist
GOTO j
DO WHILE k <= totshop
  STORE STR(k,1) TO m
  STORE shop&m TO temp&m
  STORE k+1 TO k
ENDDO
*
USE zinput
APPEND BLANK
STORE tempk+1 to tempk
GOTO tempk
REPLACE jobs WITH jobn
STORE 1 TO k
DO WHILE k <= totshop
  STORE STR(k,1) TO m
  REPLACE shop&m WITH temp&m
  STORE k+1 TO k
ENDDO
STORE i+1 TO i
ENDDO
USE zinput
APPEND BLANK
GOTO BOTTOM
REPLACE shop1 WITH totshop
REPLACE shop2 WITH totjob
RELEASE ALL
RETURN

```

```

*****
* NAME OF COMMAND FILE: zlpvar.prg *
* *
* PURPOSE : THIS COMMAND FILE CREATES Xij VARIABLES BASED ON *
* NO. OF SHOPS AND NO. OF JOBS. Xij DENOTES NO. OF *
* HOURS TO BE SPENT IN SHOPi ON JOBj. TOTAL NO. OF *
* VARIABLES CREATED EQUALS THE PRODUCT OF NO. OF *
* SHOPS AND NO. OF JOBS. *
* *
* JOB NAMES AND COST/MAN HOUR ARE ALSO EXTRACTED *
* FROM zinput.dbf AND PLACED IN zlpvar.dbf. *
*****
*
*
SET TALK OFF
USE zinput
GOTO BOTTOM
*
* GET NO. OF SHOPS AND NO. OF JOBS FROM zinput.
*
STORE shop1 TO ns
STORE shop2 TO nj
*
* CREATING Xij VARIABLES.Xij DENOTES NO. OF
* HOURS TO BE SPEND IN SHOPi ON JOBj. ALSO
* EXTRACTING FROM zinput.dbf THE JOB NAME, COST/MAN HOUR.
*
USE zlpvar
DELETE ALL
PACK
*
*
STORE 1 TO si
DO WHILE si <= ns
  IF si <= 9
    STORE 'X0'+CHR(si+48) TO xs
  ELSE
    STORE INT(si/10) TO a1
    STORE si-(a1*10) TO a2
    STORE 'X'+CHR(a1+48)+CHR(a2+48) TO xs
  ENDIF
*
STORE STR(si,1) TO m
USE zinput
GOTO 2
STORE shop&m TO cost
STORE 1 TO ji
DO WHILE ji <= nj
  USE zinput
  GOTO 2+ji
  STORE jobs TO jobn
  IF ji <= 9

```

```
        STORE '0'+CHR(ji+48) TO xj
    ELSE
        STORE INT(ji/10) TO a1
        STORE ji-(a1*10) TO a2
        STORE CHR(a1+48)+CHR(a2+48) TO xj
    ENDIF
    STORE xs+xj TO xv
*
*   PLACING Xi j VARIABLES, SHOP NAMES, JOB NAMES &
*   & COST/MAN HOUR IN zlpvar.dbf.
*
    USE zlpvar
    APPEND BLANK
    REPLACE xvar WITH xv
    REPLACE xshop WITH 'SHOP'+ '&m'
    REPLACE xjob WITH jobn
    REPLACE xcost WITH cost
    STORE ji+1 TO ji
ENDDO
    STORE si+1 TO si
ENDDO
*
*   ADDING 'OBJECTIVE FUNCTION' AS THE LAST RECORD IN zlpvar.dbf.
*
GOTO BOTTOM
APPEND BLANK
REPLACE xvar WITH 'OBJECT'
RELEASE ALL
RETURN
```

```

*****
* NAME OF COMMAND FILE: zlpmatrx.prg *
* *
* PURPOSE: THIS COMMAND FILE CREATES A COMPLETE LINEAR *
* PROGRAMMING PROBLEM IN lindo format USING *
* zinput.dbf AND zlpvar.dbf. *
*****
*
*
SET TALK OFF
*
* GET NO. OF SHOPS AND NO. OF JOBS.
*
USE zinput
GOTO BOTTOM
STORE shop1 TO mats
STORE shop2 TO matj
*
USE zlpmatrx
DELETE ALL
PACK
APPEND BLANK
*
* GENERATING OBJECTIVE FUNCTION (MINIMIZATION).
*
REPLACE matvar WITH 'MINIMIZE'
STORE 1 TO mati
STORE 1 TO matx
DO WHILE mati <= mats
  USE zinput
  GOTO 2
  STORE STR(mati,1) TO m
  STORE shop&m TO mcost
  STORE 1 TO matk
  DO WHILE matk <= matj
    USE zlpvar
    GOTO matx
    STORE xvar TO matr
    USE zlpmatrx
    APPEND BLANK
    REPLACE coll WITH mcost*1.00
    REPLACE matvar WITH matr+ ' +'
    STORE matk+1 TO matk
    STORE matx+1 TO matx
  ENDDO
  STORE mati+1 TO mati
ENDDO
REPLACE matvar WITH matr
*
* GENERATING CONSTRAINTS.
*
APPEND BLANK

```

```

REPLACE matvar WITH 'SUBJECT TO'
STORE 1 TO mati
STORE 1 TO matvarn
*
* CONSTRAINTS WITH RESPECT TO TOTAL NO. OF HOURS
* AVAILABLE IN SHOPi
*
DO WHILE mati <= mats
  USE
  USE zinput
  GOTO 1
  STORE STR(mati,1) TO m
  STORE shop&m TO mathrs
  STORE 1 TO matk
  DO WHILE matk <= matj
    USE
    USE zlpvar
    GOTO matvarn
    STORE xvar TO matvar1
    USE
    USE zlpmatrx
    APPEND BLANK
    REPLACE matvar WITH matvar1+' +'
    STORE matk+1 TO matk
    STORE matvarn+1 TO matvarn
  ENDDO
  REPLACE matvar WITH matvar1+' <='
  APPEND BLANK
  REPLACE coll WITH mathrs*1.00
  STORE mati+1 TO mati
ENDDO
*
* CONSTRAINTS WITH RESPECT TO COMPLETION OF JOBS.
*
STORE 2 TO matstart
STORE 1 TO mati
DO WHILE mati <= matj
  STORE 1 TO matk
  STORE matstart+1 TO matstart
  DO WHILE matk <= mats
    USE
    USE zinput
    GOTO matstart
    STORE STR(matk,1) TO m
    STORE shop&m TO matval
    IF matval = 0
      STORE 0 TO matval
    ELSE
      STORE (1.000000/matval) TO matval
    ENDIF
  USE
  USE zlpvar

```



```
    STORE mati+(matk-1)*matj TO mdummy
    GOTO mdummy
    STORE xvar TO matvar1
    USE
    USE zlpmatrx
    APPEND BLANK
    REPLACE coll WITH matval
    REPLACE matvar WITH matvar1+' +'
    STORE matk+1 TO matk
  ENDDO
  REPLACE matvar WITH matvar1+' = 1'
  USE
  STORE mati+1 TO mati
ENDDO
*
* END OF THE LP PROBLEM.
*
USE
USE zlpmatrx
APPEND BLANK
REPLACE matvar WITH 'END'
COPY TO zlpmatrx SDF
RELEASE ALL
SET TALK ON
RETURN
```

```

*****
* NAME OF COMMAND FILE: zinface.prg *
* *
* PURPOSE: THIS COMMAND FILE PROVIDES AN INTERFACE BETWEEN *
* THE LP SOLUTION AND THE REPORT GENERATION. THE *
* RELEVANT INFORMATION IS EXTRACTED FROM THE *
* SOLUTION FILE (zlpout.dbf) AND PLACED IN *
* zlpvar.dbf FOR REPORT GENERATION PURPOSES. *
*****
*
*
SET TALK OFF
USE zlpout
DELETE ALL
PACK
APPEND FROM zlpout SDF
USE zlpvar
GOTO BOTTOM
STORE # - 1 TO totvar
STORE 1 TO count
*
* EXTRACTING THE VALUES FOR Xij.
*
DO WHILE count <= totvar
  USE zlpout
  GOTO count+7
  STORE VAL($(line,11,10)+$(line,22,3))/1000.000 TO nvalue
  USE zlpvar
  GOTO count
  REPLACE xvalues WITH nvalue
  STORE count+1 TO count
ENDDO
*
* EXTRACTING THE OBJECTIVE FUNCTION VALUE.
*
USE zlpout
GOTO 5
STORE @('.',line) TO pos
STORE VAL($(line,pos-9,9)+$(line,pos+1,3))/1000.000 TO nvalue
USE zlpvar
GOTO BOTTOM
REPLACE xvalues WITH nvalue
RELEASE ALL
RETURN

```

```

*****
* NAME OF COMMAND FILE: zlprep.prg *
* *
* PURPOSE: THIS COMMAND FILE GENERATES REPORTS FOR THE USER.*
* THE REPORTS CAN BE GENERATED BY THE USER FOR ANY *
* OF THE OPTIONS GIVEN UNDERNEATH. *
*****
*
*
SET TALK OFF
SET CONFIRM ON
USE zlpvar
STORE 'F' TO cont
DO WHILE !(cont) <> 'N'
    ERASE
    ? " REPORT GENERATOR FOR JOB-SHOP PROBLEM"
    ?
    ? "*****"
    ? "* Please SELECT one of the following options :- *"
    ? "* *"
    ? "* 1 Optimal value of the objective function. *"
    ? "* *"
    ? "* 2 Optimal(Minimal) # of hours & cost of all jobs. *"
    ? "* *"
    ? "* 3 Optimal(Minimal) # of hours and cost of a parti - *"
    ? "* cular job in a particular shop. *"
    ? "* *"
    ? "* 4 Optimal(Minimal) # of hours & cost of a parti - *"
    ? "* cular job in all shops. *"
    ? "* *"
    ? "* 5 Optimal(Minimal) # of hours & cost of a parti - *"
    ? "* cular shop for all jobs. *"
    ? "* *"
    ? "* 6 Quit to Main Menu. *"
    ? "*****"
    ?
    ACCEPT "Enter your option (1/2/3/4/5/6) " TO ropt
    DO CASE
    CASE ropt='1'
        ERASE
        REPO FORM zrepol FOR xvar='OBJECT'
    CASE ropt='2'
        ERASE
        REPO FORM zrepo2 FOR xvar<>'OBJECT'
    CASE ropt='3'
        ERASE
        ACCEPT "Jobname?" TO jobname
        ACCEPT "Shopname?" TO shopname
        REPO FORM zrepo3 FOR xjob=!(jobname);
        .AND. xshop=!(shopname)
    CASE ropt='4'
        ERASE

```

```
        ACCEPT "Jobname?" TO jobname
        REPO FORM zrepo4 FOR xjob=!(jobname)
CASE ropt='5'
    ERASE
    ACCEPT "Shopname?" TO shopname
    REPO FORM zrepo5 FOR xshop=!(shopname)
CASE ropt='6'
    ERASE
    RETURN
OTHERWISE
    ERASE
    ?
    ? "*** ERROR ** Selected option is out of range."
    ?
    ACCEPT "Press <RETURN> to continue" TO dummy
ENDCASE
IF (ropt > '0' .AND. ropt < '6')
    ?
    ACCEPT "Press <RETURN> to continue" TO dummy
ENDIF
ENDDO
RELEASE ALL
RETURN
```

VITA

ASHOK KUMAR RATHI

Candidate for the Degree of
Masters of Business Administration

Report: INTEGRATION OF DATA BASE MANAGEMENT SYSTEM WITH OPTIMIZATION

Major Field: Business Administration

Biographical:

Personal Data: Born in Ajmer, India, December 4, 1962, the son of Chand Karan and Subhudra Rathi. Single.

Education: Graduated from the Secondary Board of Rajasthan, Ajmer, India, in June, 1979; received Bachelor of Commerce (Hons.) Degree from The University of Rajasthan in November, 1983; received a Chartered Accountancy Intermediate Examination certificate in May, 1984.

Professional Experience: Chartered Accountancy Trainee, Price Waterhouse & Co., Calcutta, India, March, 1983 to July, 1984; Graduate Research Assistant, College of Business Administration, August, 1984, to present.