Asian Institute of Technology
Bangkok Thailand

thesis DATA RETRIEVAL SYSTEM FOR FARM MECHANIZATION CONSEQUENCES PROJECT

FRANCES G. KASALA

BEST AVAILABLE COPY

DATA RETRIEVAL SYSTEM FOR FARM MECHANIZATION CONSEQUENCES PROJECT

by

Frances G. Kasala

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Engineering

Examination Committee:
- Dr. Gopal K. Gupta (Chairman)
- Dr. Krisorn Jittorntrum
- Dr. H.N. Phien

Division of Computer Applications
Asian Institute of Technology
Bangkok, Thailand
August 1980

CONTENTS

ACKNOWLEDGEMENTS iii

ABSTRACT iv

CHAPTER 1. 1
INTRODUCTION 1
1.1 Background. 1
1.2 Statement of Problem. 2
1.3 Scope of Work. 3

CHAPTER 2. 4
LITERATURE REVIEW. 4

CHAPTER 3. 7
THE CONSEQUENCES OF FARM MECHANIZATION PROJECT 7
3.1 Objectives of the Project. 7
3.2 Application of Sampling to Agriculture. 7
3.3 Method of Data Collection. 9
3.4 Frequency of Data Collection. 9
3.5 Forms Used for Data Collection. 10
3.6 Raw Data. 13

CHAPTER 4. 16
ANALYSIS OF DATA 16
4.1 Analysis of Forms. 16
4.2 Analysis of the Codes and Variables Used. 24
4.2.1 Codes. 24
4.2.2 Variables. 25

CHAPTER 5. 26
SYSTEM DESIGN. 26
5.1 The Needs of the Project. 26
5.2 VSAM (Virtual Storage Access Method) 26
5.3 Storage Requirements. 28
5.4 Phases of System Development. 30
5.4.1 Phase 1. 31
5.4.2 Phase 2. 31
5.4.3 Phase 3. 33

CHAPTER 6. 51
CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER WORK 51
6.1 Conclusions. 51
6.2 Limitations & Recommendations for Further Study. .52

APPENDIX A A1

APPENDEX B B1

APPENDIX C C1

APPENDIX D D1

APPENDIX E E1

APPENDIX F F1

APPENDIX G G1

ACKNOWLEDGEMENTS

The author wishes to express her gratitude and profound thanks to the following persons who, in one way or another, have contributed to the preparation and final completion of this thesis work:

to her advisor, Dr. Gopal K. Gupta, and to the members of her thesis committee, Dr. N.H. Phien and Dr. Krisorn Jittorntrum;

to Dr. Dow Mongkolsmai, Mr. Veerasak, and staff of IRRI, Thailand for their cooperation and patience;

to Mr. Burt Duff and Dr. John Wicks for allowing the author to take on this project and for their encouragement;

to the Royal Thai Government for its scholarship without which the study at AIT would not have been possible;

Lastly, the author expresses her deepest appreciation to Mr. Rafiul Ahad for his guidance, valuable suggestions, and constant encouragement; and to the author's family for their moral support and concern.

ABSTRACT

This paper presents the development of a data management and retrieval system for the Farm Mechanization Consequences Project launched by IRRI for three Asian countries.

First, the method of data collection is discussed. Then the forms involved in the SURVEY are analyzed to determine file organization. VSAM Entry Sequenced Data Sets are then created and appropriate alternate index set up. Finally, the retrieval program is discussed.

The 477 variables involved in the SURVEY were given unique names which the user has to specify in order to retrieve values. The values could be grouped by some variable and can be selected based on some condition.

CHAPTER 1

INTRODUCTION

1.1 Background.

In recent years development specialists have been increasingly concerned with the sector of the population in developing countries which includes small scale, subsistence farmers. Part of the reason for this heightened interest stems from the realization that past development strategies have failed to distribute the fruits of economic progress to the majority of the population, many of whom are subsistence farmers. The increasing levels of unemployment and worsening maldistribution of income are grim testimonials to this fact. Recognizing that the roots of widespread poverty are found in the rural, subsistence sector, development planners are initiating small development programs throughout the developing world. These programs designed to enhance the levels of living for small farmers could assist in alleviating these interrelated problems. The key elements of such programs are the stimulation of rural employment both on and off the farm, and the raising of productivity levels.

The International Rice Research Institute (IRRI), with headquarters at Los Banos, Philippines, is a research institute established to serve all tropical rice producing countries. It has been a major source of new rice technology for the tropics since 1962.

The Farm Mechanization Consequences Project launched by International Rice Research Institute is a study begun in early 1978. The countries involved are Indonesia, the Philippines, and Thailand. The primary objective of the study is to determine the impact of mechanization of small rice farms on production, income, and rural employment. Details of the survey are given in the next chapter.

1.2 Statement of Problem.

Mechanization of small farms in developing nations involves four major issues:

(1) if mechanization enhance output and in what way mechanization enhances output
(2) the degree that labor is displaced by machines and the alternative employment opportunities for that labor
(3) the extent of benefits of mechanization on poor farmers
(4) the policies that governments should follow to obtain the desirable benefits of mechanization while minimizing undesirable effects.

To face these issues, there is a need to assess the consequences of farm mechanization. Also to provide engineers and research administrators with more precise specifications of machinery to meet the needs of small rice farmers, especially in countries faced with growing rural labor force and an increasing demand for food. Previous research on mechanization indicates the severe limitations of small surveys and case studies. These most often do not adequately capture the dynamic implications of mechanization in the allocation of resources within farm households and village communities.

Small surveys pose severe limitations on the extent of analysis that can be done. To meet such limitations, this large scale survey was initiated. With the present rate of technological development, changes in economy, and population growth, survey data can be assumed valid only for a short time. Therefore the data must be processed readily and made available to researchers. For such an ambitious survey involving such large amounts of data, this is possible only with computerized data processing and data management.

Thus, a data management system needs to be developed to transform the data into a form ready for processing and analysis.

1.3 Scope of Work.

This thesis study has the objective of designing a system that will facilitate selective retrieval of data elements as demanded by the user. These data elements, then, will be used for the analysis of the impact of farm mechanization.

The main objective of the thesis is to develop a data management system suitable for the data collected by Farm Mechanization Consequences Project. This includes:

(1) The analysis of forms to suppress unused columns
(2) The splitting and/or combination of forms so that homogeneous files (fixed record length and same format) could be set up.
(3) The selection of storage medium and data organization.
(4) The development of query facility to assist users in the retrieval of data.

CHAPTER 2

LITERATURE REVIEW

Since the early 1960's there has been a growing emphasis on the development and diffusion of agricultural technology in the less developed countries.

WHARTON (1968) pointed out three frequent criticisms of agricultural economics research undertaken by Southeast Asian agricultural economists: (a) it is purely descriptive or fact-finding with little analysis, (b) it is designed to prove preconceived values or norms, and (c) it is frequently a blind imitation of the research projects undertaken in the West by Western economists. He further mentioned general deficiencies which can be summarized as follows: (1) Very little research has been undertaken to ascertain the farmers' goals of production, the nature of their decision making processes, and the influences and patterns of diffusion and adoption of practices and enterprises; (2) Very little empirical research has been of a testing nature, i.e., very rarely are the new models of growth tested against hard facts; (3) Research has been least directed at the general values, attitudes and motivations of farm people and how these affect the goals of production, the farmers' decision-making process, receptivity to innovations, new enterprises and techniques, and the general conduct of farming. He concluded with a plea to consider certain areas which he considered were neglected in previous and current researches: (a) interdisciplinary research which focuses upon the motivational and attitudinal variables as they affect the productive and developmental process, and (b) research which provides solid empirical evidence on the issues of underemployment, price responsiveness, and the economic behaviour of subsistence producers.

In as early as 1969 when amount of data was not as surmountable as now, HUNT (1969) mentioned the problem of data handling in statistical studies. He added that 'classification of a considerable amount of agricultural economic material for computerized retrieval is far from straightforward'.

Researchers and users alike realize the role that agricultural statistics play. No student of agriculture today can afford to neglect the contribution that

statistical science is making to agricultural research. A number of books are written on agricultural statistics.

FINNEY (1972) stated that statistical science must be part of the standard methological approach to many problems in agriculture. He also mentioned that development of high-speed computers has made practicable statistical analyses on a scale that could not have been considered twenty years ago, that the electronic computer now enables large bodies of data to be intensely analyzed. Increasing numbers of pre-written programs. However, he says, 'the extent to which a standard computer program can serve an investigation depends upon both the investigator and his problem. Easy availability of the computer brings new dangers. Inadequate understanding of the logic on which an analysis is based can lead to choice of a program that for the particular use has unsatisfactory features or even is wholly inappropriate. Computer programs tend to increase the sophistication of statistical analysis, and greater experience of theory and practice is required to exploit them safely and well.

MORSS, HATCH, MICKELWAIT, and SWEET (1976) made a study on the strategies of small farmer development. Three data collection approaches were mentioned namely traditional survey, in-depth interview, and observation/measurement. For each technique, the usefulness of the data and impact of the data were discussed.

A number of seminars and conferences had been devoted to agricultural development. It is envisioned that this trend will go on considering the rate of technological changes, economic stability, and the need for increased food production, and the fact that small farmers form the majority of agriculture producers.

HONGDON (1974), SCHULTZ (1974), and TINNERMEIER (1974) in their separate papers supported the issue that despite the apparent economic benefits that may be associated with the adoption of new production technologies, cultural attitudes, and institutional considerations may impede adoption and the long term participation by farmers.

VALDES, SCOBIE, and DILLON (1979) edited the proceedings of the International Conference on Economic Analysis in the Design of New Technology for Small Farmers, November 1975. The papers presented centered generally on technology and rural development programs, and how they relate to small farmers.

IRRI, in its annual reports has constantly reported the progress of its research projects, one of which is the Consequences of Farm Mechanization Project. In one of its reports (1976) it stated that aggregate national statistics

will be used to analyze the social returns to rice research, and the policy alternatives for rice self-sufficiency. Farm surveys will provide data for analyzing changes in labor use in rice production, and for studying in depth the rice village economy.

An IRRI workshop report (1978) summarized the involvement of IRRI in the project, the research design and hypothesis to be tested, and a brief overview of the data collection, sampling procedures, conduct of survey, and record keeping.

The Project Proposal (1978) on the Consequences of Small Rice Farm Mechanization on Production, Income, and Rural Employment in Selected Countries of Asia discussed the study objectives and framework of the project.

MORAN and UNSON (1978) prepared the Operations Handbook of Farm Survey and Record Keeping Procedures. The handbook enumerated the data forms used, method of data collection, and coding system used.

CASILLAN (1979), SUMIRAN (1979), and WICKS (1979) in papers presented at the Workshop on Farm Mechanization discussed the purpose of the project, deficiencies of the proposed format, data editing and validating procedures, and computer resources presently used.

NIYOMVIT (1979) discussed the SURVEY activities on the Consequences of Farm Mechanization Project in Suphanburi, Thailand.

WICKS (1980) presented procedures for constructiong two-way tables for this project. This provided valuable insight into the possible information needs of the users for analysis.

CHAPTER 3

THE CONSEQUENCES OF FARM MECHANIZATION PROJECT

3.1 Objectives of the Project.

The Farm Mechanization Project has three specific objectives. The first is to evaluate the effects of farm mechanization on small rice farms in selected locations in Asia. This objective focuses on the impact or potential impact of machinery on employment, production, and income of farmers and landless workers. It involves the analysis at the farm, regional, and cross national levels. The second objective is to develop an improved understanding of the tasks involved in different systems of rice production to provide guidelines for the design of mechanization appropriate for rice producers. The third objective is to improve the capacity of researchers within the region to conduct such studies.

3.2 Application of Sampling to Agriculture.

Most of the sampling enquiries in developing countries have conformed to one basic pattern. Typically the whole national agricultural industry is divided into sections like regions, type of farming areas, and others. This is known as stratification. Within each of these, a sample of units, of a size which can be readily handled, is selected at random. It could involve various stages, starting from first stage units which may be too big for the detailed recording work down to the final stage units which are randomly chosen from the bigger stage units. Such a design is called stratified, multistage, random sample design.

For the Farm Mechanization Consequences Project, the above technique was used. Within the involved country, initial survey was conducted for village selection as to type of farming areas. Within each of these villages, households were randomly selected. From a household census, households were grouped into:

(1) rice farm household,
(2) field labor households (landless), and
(3) non-field labor households

A rice farm household is one whose head cultivates at least 1000 sq. meters of land. A field labor household (landless) is one whose head does not operate a farm but derives major part of his income from working on farms owned by others.

Group (2) forms a separate stratified cell. Group (3) is not in the study, because they are not considered involved in farming.

Group (1) is further stratified based on four levels of mechanization (animal, animal & machine, power tiller, and tractor for land preparation) and two levels of water control (rainfed and irrigated). This stratification is given in Table 3-1.

LEVEL OF WATER CONTROL	LAND PREPARATION							
	ANIMAL		ANIMAL&MACH		POWERTILLER		TRACTOR	
	SURVEY	FRK	SURVEY	FRK	SURVEY	FRK	SURVEY	FRK
Rainfed	30	5	30	7	-	-	30	5
Irrigated	30	3	30	7	30	8	30	7
					30	4	30	6

Table 3-1. Number of Household Classified by Land Preparation.

Notes: There are not enough samples with power tiller in rainfed area so this stratification cell is not considered. Farmers with machine for land preparation usually grow two crops in one year. (irrigated area).

For rice farm households, stratification unit was based on parcels within the sample villages and not exceeding 10 hectares.

Table 3-1 shows 270 household samples for Group (1). Another 50 landless labor samples, Group (3), were added for a total of 320 households for SURVEY. For FRK, the total number of samples in rainfed area is 17 and in irrigated area 35, for a total of 52 farms for FRK. Note that, FRK samples is actually a subset of the SURVEY samples, i.e., the 52 farms belong to some households in the SURVEY samples. This is so because the FRK forms are meant to supplement the data collected from SURVEY.

3.3 Method of Data Collection.

The data gathering of the study consists of three components:

1) SURVEY - series of cross sectional survey conducted:
   * at start and end of each cropping season involving 28 forms (Forms 911A to 945)
   * for every change of farm activity involving 1 form (RU, Resource Utilization)
2) FRK - Farm Record Keeping
   complimentary record keeping monitored daily on selected farms involving 9 forms.
3) DATA IDENTIFICATION - filled in once by each country team involving 9 forms.

Data enumerators and village assistants were trained before actually sent to the field. The topics emphasized during the training are the purpose of each form and recording instructions for each. They are also given light as to the directions of the hypotheses testing to be done later. This later topic is included to guide the enumerators for on-the-spot decisions on special cases they might encounter with the respondents.

The enumerators interview the farmers and household members of the selected households. These enumerators submit the completed questionnaires to the field supervisor who checks for errors and inconsistencies. Completed forms are sent to the data preparation unit where they are coded in SURVEY forms.

The Project was launched by IRRI in early 1978 but actual data collection started in 1979 and will last for three years.

3.4 Frequency of Data Collection.

Because of the relatively large number of SURVEY forms, they are grouped into four:

1) forms to be administered at land preparation stage.
2) forms to be administered at immediate post harvest season.
3) forms to be administered both at land preparation stage and immediate post harvest season
4) RU forms to be administered for every input per activity performed per parcel

For the FRK (Farm Record Keeping) forms, data is collected daily or whenever an activity or transaction is performed. The major purpose of FRK is to collect the detailed data that is difficult to obtain during one visit surveys because of memory bias and farmers' recall problems.

For the Data Idetification Forms, data is collected once by each country team to develop background data on project site.

3.5 Forms Used for Data Collection.

Table 3-2 gives the forms and the rounds in which they are administered. Analysis of these forms and further details are given in Chapter 4.

code	Form	Round1	Round2
900	Data Identification	*	
911	Farm Labour Form A	*	
911099	Farm Labour Form B	*	*
911020	Farm Labour Form C	*	
930	Demographic Information	*	
912	Farm Land	*	
931	Landlord's Contribution to Cultivation Expenses	*	*
932	Cropping Pattern History		*
933	Machine Use/Change in Cropping Pattern		*
913	Crops	*	
916	Draft Animals	*	
917	Productive Animals	*	
918	Farm Buildings	*	
919	Hand Farm Implements & Tools	*	
934	Other Assets: Non-Agricultural Land and Vehicles	*	
935	Home Consumer Durables	*	
920	Financial Liabilities	*	
915	Farm Machinery	*	*
936	Decision-Making re: Machine Purchaces	*	*
937	Machine Use,Income,Maintenance, Accidents with Machine	*	*
938	Machine Breakdown and Repair	*	*
921	Inventory Changes		*
RU	Resource Utilization	every farm activity	
939	Disposal of Product		*
940	Sale/Marketing of Product		*
941	Land Reclamation	*	*
942	Extension Service/Machine Operation		*
	Training/Farmer's Organization		*
943	Work Opportunities/Preference of Landless Field Labourers	*	
944	Actual & Ideal Dates for Crucial Farm Activities	*	*
945	Animal Diseases/Accidents with Animals	*	*

Table 3-2. List of SURVEY Forms Used for Round1 and Round2

There are 9 types of forms used for FRK. These forms are not in the same format as the SURVEY forms, but contains the same information. They were designed to facilitate easy recording by farmers. To enter data collected, information is transferred to the relevant SURVEY forms as shown in Table 3-3.

From FRK Form	To SURVEY Form
Cultivation	Resource Utilization
Production	
A. Total Production	Resource Utilization
B. Disposal of Product	Disposal of Product (939)
C. By-Product	Disposal of Product (939)
Marketing	
Date, quantity, value	Resource Utilization
Remaining data	Marketing of Product (940)
Supplementary Income A (off-farm income)	Resource Utilization
Supplementary Income B (non-farm income)	Farm Labour A (911a)
Supplementary Income C (rents received)	
Animal Care/Maintenance	Draft Animals (916)
Animal Diseases/Accidents	Animal Diseases/Accidents (945)
Machine Use/Maintenance	Machine Breakdown/Repair (938)

Table 3-3. Correspondence of FRK Forms to SURVEY Forms.

The following forms serve as background data and are filled in once by each country team.

1) Form 901 Project name and code
2) Form 902 Data origin, i.e., country, province, no. of villages, no. of observations
3) Form 903 Selective criteria
4) Form 904 Conversion rates and climatic zones
5) Form 905 Monthly rainfall
6) Form 906 Mean monthly temperature
7) Form 907 Author of survey
8) Form 908 Description of soil type
9) Form 910 Write-up cards (project description, revisions, etc.)

3.6 Raw Data.

Data is transferred from the forms to standard 80-column punched cards. Three data items - MANDATORYCODE, HHCODE, and PROJCODE - are always located at the beginning and end of all forms, except RU and 911B. For RU, col 1 to 3 are for enterprise code, col 75 to 77 for household code, and col 78 to 80 for project code. For the 911B col 1 to 6 are for mandatory code and col 75 to 80 are the same as in RU. For the rest of the forms, col 1 to 3 are for mandatory code, and col 75 to 80 follow that of RU.

In Thailand, data is recorded on magnetic tapes. As for the number of records generated, Table 3-4 gives the summary of estimates.

Form No.	Scope/card	Estimated No. of cards/HH
911A	1 member	4
911B	1 household	1
912	1 parcel	3
913	2 crops	3
915	1 machine	1
916	1 draft animal	1
917	1 prod animal	3
918	2 buildings	1
919	5 hand implements	5
920	2 liabilities	1
930	1 household	1
931	2 parcels	2
932	2 yr code/parcel	6
933	1 parcel	3
934	2 pcs land	1
	3 vehicles	
935	6 consumer goods	2
936	3 machine	1
937	1 mach accident	3
938	1 mach breakdown	3
939	1 crop	4
940	2 sales/crop	3
941	1 land	1
942	2 trainings	1
	3 organizations	
943	2 members	1
944	2 parcel	2
945	3 diseases	1
	3 accidents	
RU	5 types I/O per enterprise	132*

Table 3-4. Estimates of 80-Char-Records Generated.

* The following are the assumptions made for this estimate:

1) 2 activities per enterprise per week
2) 5 input/output per activity
3) 3 parcels per household
4) 22 weeks per season
5) 5 input/output per enterprise per card

The estimated number of cards per household sum up to 190. For a sample size of 320 households, it is estimated that 60,800 cards/cropping season is generated. With this estimate, 183,000 cards will be generated in one year.

The estimate given above is for Thailand only. Indonesia and the Philippines will also be generating records close to this estimate.

CHAPTER 4

ANALYSIS OF DATA

4.1 Analysis of Forms.

The forms used for data gathering were studied and analyzed as to their purpose and scope, data items involved, and coding system used.

Several of the forms were found to be interrelated. The forms were then grouped based on possible queries that will be made on the whole data base, and the purpose of each form. The list of this grouping is given in Table 4-1.

It was also noted that several forms have repeated sets of data items. For example, in Form 913 information for two crops are recorded in one form. This means that direct access to those records via alternate index is not possible.

The raw data supplied by the Department of Agriculture, Thailand was compared with their respective forms. It was mentioned in the previous chapter that all records of the raw data are of length 80 characters. Therefore many records have embedded blanks or are padded with zeroes.

In line with these observations, several files were created with the objective of minimizing data storage. For some files, records were split, and for some, records were combined.

Data Identification:

Form 901	Project Name and Code	
Form 902	Data Origin (country,province,villages,etc)	
Form 903	Selective Criteria (optional)	
Form 904	Conversion Rates/Climatic Zones	
Form 905	Monthly Rainfall	
Form 906	Mean Monthly Temperature	
Form 907	Author of Survey	
Form 908	Description of Soil Type	
Form 910	Write-up Card (description of project)	

Personal Data of HH-members:

Form 911	
Form 911099	
Form 911020	++
Form 943	*
Form 930	
Form 921	++

Land Parcel Data:

Form 912	
Form 913	*
Form 921	++
Form 931	*
Form 932	
Form 933	
Form 941	
Form 944	

Crop Production:

Form RU001/398	*
Form 913	*

Draft Livestock:

Form RU501/503	*
Form 916	
Form 921	++
Form 945	**

Productive Livestock:

Form RU510/590	*
Form 917	
Form 921	++

Table 4-1. General Grouping of Forms.

Two & Four Wheel Tractor:	
Form 915	
Form 921	++
Form 936	*
Form 937	
Form 938	
Other Machinery:	
Form 915	
Form 921	++
Form 936	*
Form 937	
Form 938	
Other Assets:	
Form 918	*
Form 919	*
Form 921	++
Form 934	**
Form 935	*
Financial Liability:	
Form 920	*
Disposal of Product:	
Form 939	***
Form 940	*
Extension Services:	
Form 942	**
Machine Training:	
Form 942	**
Organization Membership	
Form 942	**

Table 4-1. (cont)

Notes:

- * Forms are split but contained in one file.
- ** Forms are split, and generate more than one file.
- *** Every two successive records are combined to form only one record. Details are given in section xi.
- ++ Forms not used in Thailand.

The following are the details of the analysis of the involved forms used for the project.

i. Data Identification Forms:
901 to 910
- These forms are used to collect background data of the project. They are filled in only once by each country team for each project site.

ii. Personal Data of HH_members:
911A - This form is used for farmers and HH_members with other work.
One record per HH_member with other work. Record length is reduced to 58.
911B - This form is used for all members not recorded in 911A. There should be no overlap between 911A and 911B,i.e.,each HH_member is recorded in either form only.
One record per household with HH_members not recorded in 911A. Record length is reduced to 20.
911C - This form is not used in Thailand and not considered in this study.
921 - This form is not used in Thailand and not considered in this study.
930 - This form collects demographic information of the household such as count and type of HH_members and migration.
One household per record. Record length is retained at 80.
943 - This form collects data on the work opportunities available to landless field labourers and their work preferences. As such, it is only applicable to landless labour households. Two income earners are recorded in one form, with exact information for each. To facilitate direct access for each income earner, there should be only one income earner per record. Each form, therefore is split into 2 records.
One income earner per record. Record length is reduced to 36.

iii. Land Parcel Data:
912 - This form collects background data on each parcel the farmer operates. Data includes technical information,taxes, and status of parcel.
One parcel per record. Record length is reduced to 67.
913 - This form collects data on the crops planted on every parcel each cropping season. Two parcels are recorded in one form, with

exact information for each. One difficulty encountered however, is the nonsymmetry of the form with respect to recorded structure for each parcel. To facilitate direct access to each crop planted, the form is split to accomodate only one parcel per record.
One parcel per record. Record length is reduced to 37.

921 - This form is not used in Thailand and not considered in this study.

931 - This form collects data on landlord's contribution to cultivation expenses. There are two parcels recorded per form, one rented-in and one rented-out. Tenure status identifies if the parcel is rented-in or rented-out. This is important to distinguish the record since they will be stored in the same file.
One parcel per record. Record length is reduced to 33.

932 - This form collects data on the cropping pattern of every parcel operated by the farmer. For this reason, 2 records are taken for each parcel - one for survey year and one for year before mechanization - to determine any change in cropping pattern. The 2 records are differentiated through the data year code.
One parcel per record. Record length is reduced to 67.

933 - This form collects data on machine use for land preparation and its effects on cropping pattern, water supply, and tenure status of parcel.
One parcel per record. Record length is reduced to 71.

944 - This form collects data on actual and ideal dates of crucial farm activities. Two parcels are recorded in the original form, with exact information for each. Again, the form is found unsymmetrical. The original form is split.
One parcel per record. Record length is reduced to 42.

iv. Crop Production.

RU - Resource Utilization Form. This form collects data on every input per activity per parcel operated by farmer. One data item involved in this form is the enterprise which could be any of three types: crops, draft livestock, or productive livestock. Only one enterprise is contained in one form, but may have 3 types of input/output used in the

activity. If more than 3 types of i/o are involved in one enterprise, data is continued in another form. The original form is split. One type of I/O per activity per enterprise in one record. Record length is reduced to 32.
For this group (Crop Production) of forms, only records with enterprise coded as 001 to 398 are considered.

913 - This form is discussed in Sec. iii.

v. Draft Livestock.

RU - Details are discussed in Sec. iv.
For this group of forms, only records with enterprise coded as 501 to 503 are considered.

916 - This form collects data on draft animals used for farm activities. Each form contains one type of draft animal and 5 subtypes. General data for the type of draft animal is 18 bytes long while data for each subtype is 12 bytes. If original record is to be split, the 18 bytes will have to be repeated for each subtype, which is redundant if there is more than one subtype per type of draft animal.
One type of draft animal per record. Record length is reduced to 78.

921 - This form is not used in Thailand and not considered in this study.

945 - This form collects data on the diseases and accidents involving the draft animal during the season for the purpose of determining its effects on farm activities. Three occurrences each for diseases and accidents are recorded in this form. Since the diseases may be taken independent of accidents, two independent files are generated.
One disease file of record length 23 and with disease occurrence per record. One accident file of record length 16 with one accident occurrence per record.

vi. Productive Livestock.

RU - Details are discussed in Sec. iv.
For this group of forms, only records with enterprise coded as 510 to 590 are considered.

917 - This form collects data on draft animals used for farm activities. Each form contains one type of draft animal and 5 subtypes. General data for the type of draft animal is 18 bytes long while data for each subtype is 12 bytes. If original record is to be split,

the 18 bytes will have to be repeated for each subtype, which is redundant if there is more than one subtype per type of draft animal.
One type of draft animal per record. Record length is reduced to 78.

921 - This form is not used in Thailand and not considered in this study.

vii. Two and Four Wheel Tractors.

viii. Other Machinery.

915 - This form collects information on each type of farm machinery owned by the respondent. As such, this form is only applicable to farmers who own machinery.
One machine per record. Record length is reduced to 76.

921 - This form is not used in Thailand and not considered in this study.

936 - This form collects information on persons involved in machine purchase decisions. Three machines are recorded in one form, with exact set of information for each. The original form may therefore be split.
One machine per record. Record length is 26.

937 - This form collects data on utilization and maintenance of every machine owned by respondent.
One machine per record. Record length is 78.

938 - This form collects data on the breakdown and repairs of machine. Data includes the delay caused on farm activities.
One breakdown per record. Record length is 61.

ix. Other Assets.

918 - This form collects data on buildings owned by respondent. Two buildings are recorded in one format with exact set of information for each, and may therefore be split.
One building per record. Record length is reduced to 33.

919 - This form collects data on farm implements and tools. Each form contains 5 types of implements. Often, less than 5 types are recorded thus wasting so much storage space. The original form is split to contain only one type of implements or tools. Record length is reduced to 22.

921 - This form is not used in Thailand and not considered in this study.

934 - This from collects data on assets such as non-agricultural land and vehicles. Recorded in each original form are two pieces of land

and three vehicles. Since the land may be taken independently of vehicle, two independent files are generated.
Non-agricultural land file has record length of 23, one piece of land per record.
Vehicle file has record length of 26, one vehicle per record.

935 - This form collects data on other assets such as home-consumer durables. Recorded on each original form are 6 types of home-consumer durables. Often less than 6 types are recorded in one form thus wasting so much storage space. The original form in therefore split into 6 records.
One type of home-consumer durable per record. Record length is reduced to 21.

x. Financial Liability.

920 - This form collects data on financial liabilities incurred by the respondent. Two types of financial liability are recorded in one form and may therefore be split into 2 records.
One type of financial liability per record. Record length is reduced to 34.

xi. Disposal of Product.

939 - This form collects data on the total yield of the farmer and its disposal, for every crop per season. One complete record is contained in 2 cards because it cannot be accomodated in a single card. As such, the original file for this form contains 2-record formats. These two records for each crop are combined to give a single record.
One crop per parcel per record. Record length is 131.

940 - This form collects data on the sale and marketing of products. Two sale transactions are recorded in one form, but involving only one type of crop. If the same crop involves more than 2 sale transactions, a second form is filled in. If another crop is sold, another form is filled in. To facilitate direct access to each sale transaction, the form is split into 2 records.
One sale transaction per crop per record. Record length is reduced to 48.

xii. Other Services.

942 - This form collects data on extension services, machine operation, and farmer's organization. Two machine trainings and three farmers organizations may be recorded per

form. These three services may be treated independently. Thus, 3 files are generated: extension services file with record length of 25, machine training file (one training per record) with record length of 17, farmer's organization file (one organization per record) with record length of 21.

4.2 Analysis of the Codes and Variables Used.

4.2.1 Codes.

Codes used are all numeric ranging from one to three digits. Codes used for the whole project come to a total of about 1000 codes. A list of the codes used are given in Appendix A.

Most of the codes are not significant in the sense that the information they carry is isolated on that particular data item. Codes that are of considerable importance are discussed below.

1. Mandatory Code. This consists of 3 digits for all forms except Labour Form B which is of 6 digits (911099). The mandatory code identifies as to which form a record belongs.

2. Project Code. This consists of 3 digits. The first digit signifies the country (3 for Thailand); the second digit signifies the the project site (1 for Suphanburi, Thailand); and the third digit signifies the cropping season (follows the cropping season codes). For the test data used, the project code is 310.

3. Household Code. This consists of 3 digits and refers to the ID number of the household. This code uniquely identifies a household.

4. Parcel ID. This consists of 3 digits. The first two digits are for main parcel number and the third digit for subparcel number. If a parcel has no subparcel, then the third digit is 0. Parcel number does not uniquely identify a parcel as parcel number follows the counting number of the parcels owned and/or operated by the farmer. That is, for all households, the first parcel is always assigned 01 as the first two digits, the second parcel is assigned 02, and so on.

5. Machine ID. This consists of 1 digit only as all respondents do not go beyond 9 in the count of each type of machine owned. Just like parcel ID, machine ID cannot uniquely identify a piece of machinery. It follows the count of each type of machine owned by the respondent. If a respondent has one 2-wheel tractor its ID is 1, if he has one 4-wheel tractor its ID is again 1. Obviously, this ID must be considered or concatenated with the type of machinery to be meaningful.

4.2.2 Variables.

There are two types of variables in general, one that assigns codes for its values and one where values are derived through measures (e.g., count, amount, value). For the forms involved in SURVEY, each form has an average of 20 data fields. Some data fields are repeated in certain forms.

There are about 450 unique variables in this survey. These are listed separately in 3 tables given in Appendix A.

CHAPTER 5

SYSTEM DESIGN

5.1 The Needs of the Project.

The objective of the Farm Mechanization Project is mentioned in Chapter 1, i.e., to determine the impact of mechanization of small rice farms on production, income and rural employment.

Analysis of the 30 forms involved in the SURVEY disclosed that several data items are repeated in various forms and that the forms themselves need to be related for meaningful retrievals. To interrelate these forms, data fields need to be matched among records in various files.

There must be at least one attribute common to the forms for them to be interrelated. In the crude method, one record from the first file is read, the value of the connecting field is extracted and the other file is searched for record/s which have the value in this field. If the search is made sequentially, the time involved will become directly proportional to the product of the number of records in all forms. Considering the amount of records generated by the survey, the above technique is impractical to use for interrelating the forms, because of the large amount of computer time that will be needed.

One technique to get around this problem is to use the random access method so that when a key from one file is obtained, it could be used as the key to retrieve the other forms. In this technique each record must have a unique primary key, and/or some algorithm must be applied to the key to get a unique storage address. This technique is suitable when the keys are unique and have 1 to 1 correspondence amomg files. These conditions, however, are not met using the SURVEY data. The HHCODE which is found in all forms have more than one occurence in some files.

Another technique is to use the indexed sequential access method. This technique does not need the randomizing algorithm but still needs the unique key for each record.

5.2 VSAM (Virtual Storage Access Method)

VSAM is one access method that can be used to meet the needs as discussed in the previous section.

This access method can be used both for batch and online processing. Since all the data sets are cataloged, control commands or JCL are simplified (minimal parameters used to define data sets).

Another feature which fits the needs of the project is the ability to move data from one operating system to another in a format which is common to both systems.

Of the three types of data sets available with VSAM, ESDS was found to be the most suitable to the needs of the system.

With ESDS, records are loaded to the system without respect to the contents of the records. Their sequence is determined by the order in which they are physically arranged in the data set: their entry sequence. New records are stored at the end of the data set.

This goes well with the nature of data collection. As discussed in Chapter 3, the nature of data collection is such that some forms are collected before land preparation, some forms after harvest, and some forms before land preparation as well as after harvest. As such, data entry cannot have a convenient definite sequence, if data is to be entered as they are made available.

Alternate indexes provide ways to gain access to a related base data set through a path which logically relates a base cluster and each of its alternate indexes.

5.3 Storage Requirements.

VSAM stores record of each type of data set in the same way in a fixed-length area of auxiliary storage called a control interval. VSAM chooses the size of control interval based on the type of direct access storage device used to store the data set, the size of data records, and the smallest amount of virtual storage space the processing program will provide for VSAM I/O buffers. The size of control interval is limited by the requirements that it be a whole number of physical records. Physical record sizes that VSAM uses begin at 512 bytes and increase by power of 2 up to 4096 bytes.

A group of control intervals makes up a control area. The number of control intervals per control area of a data set is fixed by VSAM, with a minimum of two.

A group of control areas allocated for a data set is known as a cluster. A cluster which is used to store user's record is known as a base cluster.

For each base cluster, a number of alternate index clusters could be built. In structure, the alternate index is similar to the base cluster.

The whole VSAM organization is controlled by a master catalog.

The size of control interval for base cluster chosen by VSAM for Model 3344 disk is 2048 bytes and the size of the index cluster is 512 bytes. There are three control intervals per control area. One control area occupies one track. The master catalog occupies 48 tracks.

The space of each cluster is allocated in integral multiple of control areas.Each alternate index also occupies one cluster.

The number of control areas for each cluster is estimated from the number of records and the record length. Roughly, if the number of records is n and the record length is r, then the number of control areas, i.e., tracks needed, is estimated with the following formula:

$$(n \times r)/6144 \quad \text{(CEILING)}$$

where 6144 is obtained as follows:

2048 bytes/CI * 3 CI/CA * 1 CA/track

For the test data comprising of 33 households for one

cropping season, the number of tracks occupied was 167 broken down as follows: 53 tracks for base clusters, 66 tracks for alternate indexes, and 48 tracks for master catalog.

The CLCROP, CLDRAF, CLPROD clusters - the resultant files after splitting of RU forms - occupy 10, 6, and 5 tracks respectively. The CL919 and CL932 clusters occupy 2 tracks each and the remaining 28 clusters occupy 1 track each. Thus base clusters occupy a total of 53 tracks. Each alternate index occupies 2 tracks (1 for index set and 1 for data set) making a total of 66 tracks. The master catalog occupies 48 tracks.

The data occupancies of most of the base clusters are very low because of the few number of test records. Most of the control areas are less than 50% full. These occupancies will increase when real data is used.

The storage requirements for real data are discussed below.

The number of tracks occupied by MASTERCATALOG will remain at 48 tracks because the space requirement of the master catalog depends mainly on the number of clusters. No additional clusters will be created for real data.

The number of tracks occupied by ALTERNATEINDEXes will also remain at 66. This is because each control area of the index has enough space left to accomodate real data alternate keys.

The deciding factor in the storage requirements is the base cluster sizes. Since the files have different record lengths and different numbers of records, it is difficult to compute the exact storage requirement for the base clusters combined.

The average record length of the files is 46 bytes. With the estimated number of records of 183,000 for one year and using the formula given in Section 5.3 for computing the number of tracks, the space requirement for base clusters is roughly (183000 * 46) / 6144 = 137 tracks. Assuming the error of estimation to be 50%, the storage requirement is no more than 205.5 tracks.

Therefore the total storage space requirement for real data for one year is (48 + 66 + 205.5) = 319.5 or approximately 27 cylinders. This is about 4% of the IBM 3340 (70M bytes) disk.

5.4 Phases of System Development.

The phases of development as related to VSAM is depicted in Fig. 5-1.

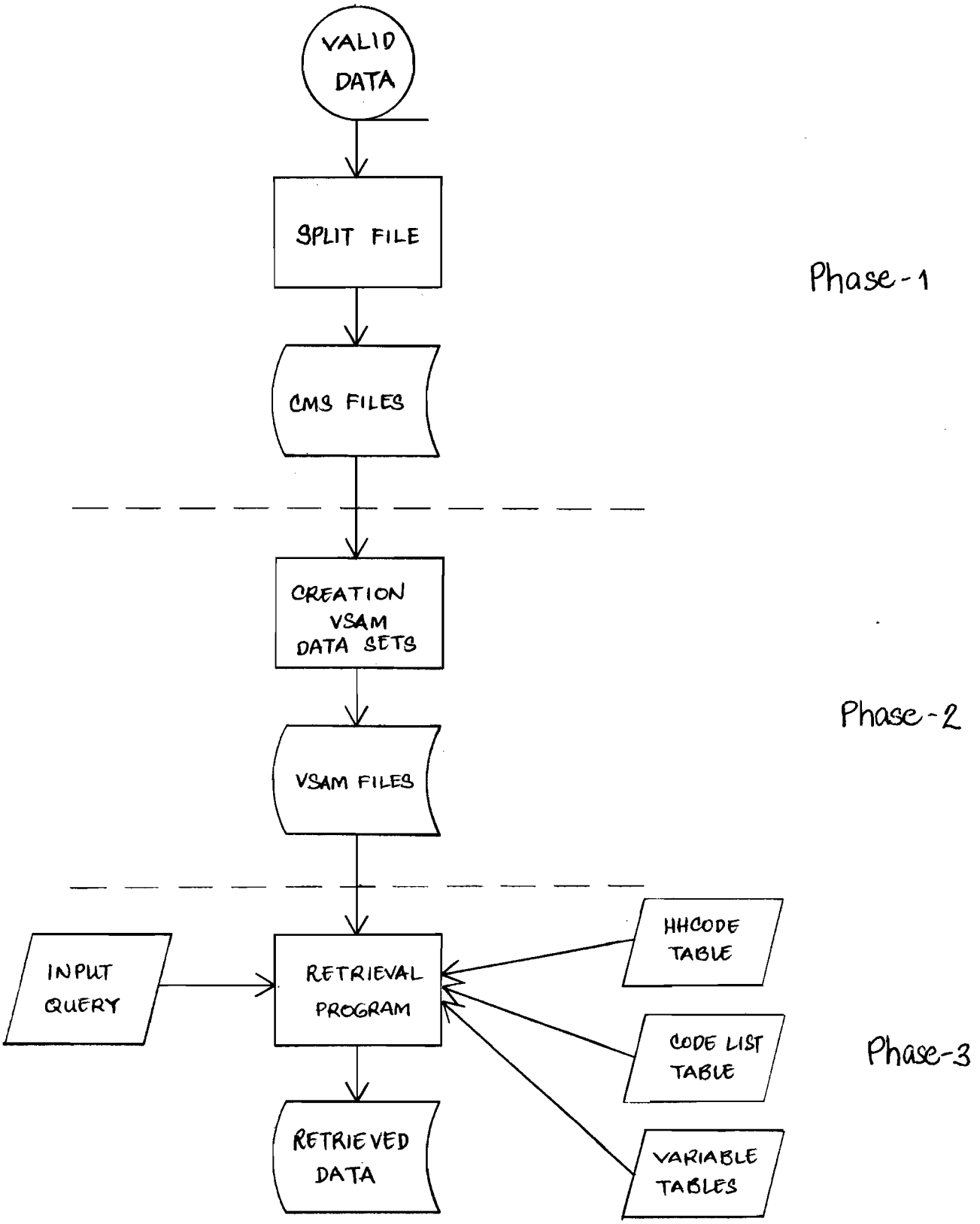

Fig 5-1. Phases of System Development.

5.4.1 Phase 1

For test data, 33 households were selected out of the raw data supplied by the Department of Agriculture. Selection of test data was based on the criteria of equal distribution among the stratified cells outlined in Table 3-1. A program was developed for this selection of test data.

For the creation of CMS files, structure of the input (raw data) and the structure of the files to be generated (with optimized storage space) are declared. Appropriate data items are read into the new structure and the resulting files saved to be later used as input to the VSAM file creation.

Processing for this phase may be done either by on-line or by batch. Due to the large amount of data for the project, tape may be used as input medium. Resultant files may also be stored on tape.

A total of 27 programs were run to process all the involved forms. One sample program together with its appropriate EXEC commands are given in Appendix B. The EXEC commands are used if VM facility is used. For batch processing, there are equivalent JCL cards.

5.4.2 Phase 2

This stage of system development involved the creation of VSAM data sets. This was done in preparation for the final phase which is the development of the retrieval program. VSAM (Virtual Storage Access Method) is the access method found to be suited to the needs of this project under study.

The disk is used to store VSAM catalogs and data sets, and must be formatted as OS disk. For this thesis study, minidisk of 20 cylinders was used (IBM 3340).

A. Disk Initialization.

VM/370 does virtual disk initialization with the OS utility program IBCDASDI. IBCDASDI formats disk volumes for OS, DOS, and VSAM under CMS usage.

The procedures followed in disk initialization are outlined below:

1. Creation of CMS file INITDISK JOB A which contains the control cards discussed previously. The contents of this file is shown in Appendix C.
2. Invoking the IBCDASDI program. The commands for this procedure are shown in Appendix C.
3. Definition of the input device when IBCDASDI program is loaded. In this case, INPUT = 3505,00C
   where 3505 is the virtual device
   00C is the virtual address.

B. Definition of Master Catalog & Clusters.

AMS (Access Method Services) is a service program used to create and maintain OS and VSAM data sets and catalogs. Control statements for AMS are provided in a CMS file and the corresponding data sets are defined using DLBL command. An example is given in Appendix D.

C. Creation of VSAM Data Sets.

After the clusters were defined and data spaces alloted, data sets were created through a set of programs which load the records into the VSAM files. The factors considered in the development of these programs were: record format, data structure and data organization. In this thesis study, ESDS was the data organization used. This is so, because of the nature of input of survey data.

D. Building of Alternate Index and Definition of Path.

An alternate index provides a unique way to gain access to a related base data set, thus there is no need to keep multiple copies of the same information organized in different ways for different applications.

A path logically relates a base cluster and each of its alternate indexes and provides a way to gain access to the base through a specific alternate index.

The selection of alternate index was based mainly on the needs of accessing records in the data sets. Household code is the altenate index chosen for the data sets.

CMS files containing AMS control statements were created and corresponding data sets were defined using DLBL statements. An example is given in Appendix E.

5.4.3 Phase 3

This phase is concerned with the design and implementation of a retrieval routine.

Discussion of Phase-3 is divided into three parts. Part-1 is the design of a query facility. Part-2 discusses data structures used to support the retrieval routine, and Part-3 is the program logic.

PART-1. The Query Facility.

The variables in the forms are divided into three categories. The first category consists of variables whose values are some measures, e.g. WORKMEMBER-AGE, PRODUCT-TOTAL. The variables in this category have unique names and occur in a single file, e.g. WORKMEMBER-AGE occur in form 911A and no where else.

The second category consist of variables whose values are coded. For example SOIL-TEXTURE of a parcel is a coded variable whose value of '1' indicates CLAY, '2' indicates CLAYLOAM, etc. The variables in this category also have unique names and occur in one form only.

The third category has variables, which may be coded, that appear in more than one form, e.g. PARCEL-ID occurs in ten forms.

The variables in the first category will be called ordinary variable (OVAR), those in the second category will be called coded variable (CVAR) and those in the third will be called special variable (SVAR). The formats of the query are shown below.

FORMAT-1:

```
     | COUNT |                        | COUNT |
GET  |  SUM  |   OVAR1 . . . . .      |  SUM  |  OVARn
     |AVERAGE|                        |AVERAGE|

     | BY  SVAR1  SVAR2 . . . . SVARn |

     |       |  OVAR1 cond  value | |
     | WHERE |                    | |
     |       |  CVAR1   =   code  | |
```

where cond in the WHERE clause could be one of the following: < , <= , = , >= , > , ¬=

FORMAT-2:

FORMAT-2 is like FORMAT-1 except that the WHERE clause preceeds the BY clause.

In the above query OVAR1, OVAR2, ...,OVARn can be from different forms. The BY clause is used for two purposes:

(1) to group the OVARs retrieved by values of SVAR, and
(2) to connect OVARs through SVAR if OVARs come from different forms

The BY clause in this query is actually the extention of the default BY clause with household code. In other words if BY clause is not mentioned the OVARs will be grouped and connected by household code.

The WHERE clause is used to select the OVARs retrieved. The selection can be either on the ordinary variables or on the coded variables. If selection is to be done on the ordinary variables then it can be done by selecting OVARs satisfying one of the six conditions with given value.

e.g. WHERE WORKMEMBERAGE > 40

If selection is to be done based on the coded variable then WHERE clause should have the format.

WHERE SOIL-TEXTURE = CLAY

Here CLAY is the constant value whose code is obtained by the system during run time.

In some cases the variables involved in the WHERE clause may not be one of the variables to be retrieved and it may come from a different form. In this case same sort of connection is needed. In FORMAT-1 of the query where the WHERE clause follows the BY clause, the connection is done through HH code. In this case the variable used in the WHERE clause must have one and only one occurence for each HH code.

In FORMAT-2 where BY clause follows the WHERE clause, the variable involved in the WHERE clause is connected through HH code, SVAR1, SVAR2,....., SVARn.

If more than one set of conditions are used then they are considered to be 'AND'.

EXAMPLE-1:

GET AVERAGE WORKMEMBER-AGE AVERAGE YRS-STUDY .

will yield two values for each household as shown below (the actual output does not contain the heading).

HH-CODE	WORKMEMBER-AGE	YRS-STUDY
120	48	02
131	33	04
...	..	..
...	..	..

EXAMPLE-2:

GET PARCEL-ANNUAL-MAINTENANCE-COST BY PARCEL-ID .

will produce the following result:

HH-CODE	PARCEL-ID	ANNUAL-MAINTENANCE-COST
120	10	0
131	20	0
307	40	450
...	.	..
...	.	..

EXAMPLE-3:

```
GET ESTIMATED PARCELVALUE
WHERE LPMECHANIZED = YES BY PARCEL-ID .
```

HH-CODE	PARCEL-ID	ESTIMATED-PARCELVALUE
131	10	35000
163	10	21000
163	20	15000
...	..	.....

All parcels in the result (Example 3) are mechanized.

Actual outputs for the three examples are given in the next pages.

ILL: QUERY DATA U IPLE RELEASE I SYSTEM (VM/370 PUT8702) 02-11-80

ET AVERAGE WORKWEEK-AGE AVERAGE YRS-STUDY .

Best Available Document

AGE | SEX

BEST AVAILABLE COPY

38

FILE: QUERY DATA ID IPUE RELEASE 1 SYSTEM (VM/370 PUT6002) 02-11-80

GET PARCEL-ANNUAL-MAINTENANCE-COST BY PARCEL-ID .

BEST AVAILABLE COPY

```
12000000001000000000000000000000000000000000000000000000000000000000000000000000
15100000001000000000000000000000000000000000000000000000000000000000000000000000
16300000001000000000000000000000000000000000000000000000000000000000000000000000
16300000002000000000000000000000000000000000000000000000000000000000000000000000
22100000001000000000000000000000000000000000000000000000000000000000000000000000
22100000002000000000000000000000000000000000000000000000000000000000000000000000
22100000003000000000000000000000000000000000000000000000000000000000000000000000
30500000001000000000000000000000000000000000000000000000000000000000000000000000
30700000001000045000000000000000000000000000000000000000000000000000000000000000
30700000002000000000000000000000000000000000000000000000000000000000000000000000
30700000003000000000000000000000000000000000000000000000000000000000000000000000
32200000001000000000000000000000000000000000000000000000000000000000000000000000
32200000002000000000000000000000000000000000000000000000000000000000000000000000
32200000003000000000000000000000000000000000000000000000000000000000000000000000
32500000001000000000000000000000000000000000000000000000000000000000000000000000
33400000001000000000000000000000000000000000000000000000000000000000000000000000
33400000002000000000000000000000000000000000000000000000000000000000000000000000
33600000001000000000000000000000000000000000000000000000000000000000000000000000
33600000002000000000000000000000000000000000000000000000000000000000000000000000
33600000003000000000000000000000000000000000000000000000000000000000000000000000
33700000001000000000000000000000000000000000000000000000000000000000000000000000
40300000001000000000000000000000000000000000000000000000000000000000000000000000
40300000002000000000000000000000000000000000000000000000000000000000000000000000
40300000003000000000000000000000000000000000000000000000000000000000000000000000
40300000004000000000000000000000000000000000000000000000000000000000000000000000
40800000001000000000000000000000000000000000000000000000000000000000000000000000
40800000002000000000000000000000000000000000000000000000000000000000000000000000
42300000001000000000000000000000000000000000000000000000000000000000000000000000
42300000002000000000000000000000000000000000000000000000000000000000000000000000
50400000001000000000000000000000000000000000000000000000000000000000000000000000
50700000001000000000000000000000000000000000000000000000000000000000000000000000
50700000002000000000000000000000000000000000000000000000000000000000000000000000
50700000003000000000000000000000000000000000000000000000000000000000000000000000
50800000001000015000000000000000000000000000000000000000000000000000000000000000
50800000002000025000000000000000000000000000000000000000000000000000000000000000
50900000001000000000000000000000000000000000000000000000000000000000000000000000
50900000002000000000000000000000000000000000000000000000000000000000000000000000
51000000001000200000000000000000000000000000000000000000000000000000000000000000
51000000002000000000000000000000000000000000000000000000000000000000000000000000
51000000003000000000000000000000000000000000000000000000000000000000000000000000
51000000004000000000000000000000000000000000000000000000000000000000000000000000
51600000001000000000000000000000000000000000000000000000000000000000000000000000
51800000001000000000000000000000000000000000000000000000000000000000000000000000
52300000001000000000000000000000000000000000000000000000000000000000000000000000
61200000001000000000000000000000000000000000000000000000000000000000000000000000
62500000001000000000000000000000000000000000000000000000000000000000000000000000
62700000001000000000000000000000000000000000000000000000000000000000000000000000
63200000001000000000000000000000000000000000000000000000000000000000000000000000
63200000002000000000000000000000000000000000000000000000000000000000000000000000
64800000001000000000000000000000000000000000000000000000000000000000000000000000
64800000002000000000000000000000000000000000000000000000000000000000000000000000
71000000001000000000000000000000000000000000000000000000000000000000000000000000
71000000002000000000000000000000000000000000000000000000000000000000000000000000
71000000003000000000000000000000000000000000000000000000000000000000000000000000
74800000001000000000000000000000000000000000000000000000000000000000000000000000
```

BEST AVAILABLE COPY

FILE: RESULT DATA D IPUE RELEASE 1 SYSTEM (VM/370 PUT8002) 02-11-89

```
74200100000320 0000000000000000000000000000 0000000000000000000000000000
7541000100010000000000000000000000000000000000000000000000000000000000
```

BEST AVAILABLE COPY

ILE:-COLLY DATA D APPLE RELEASE 1 SYSTEM (VM/370 PUTD 02) 02-11-00

RT ESTIMATE--PARCELVALUE WHERE LPMECHANIZED = YES BY PARCEL-ID .

BEST AVAILABLE COPY

BEST AVAILABLE COPY

PART-2. Data Structures.

The following five tables were set up:
(1) Ordinary Variable Table (VTABLE)
(2) Coded Variable Table (CTABLE)
(3) Special Variable Table (SVTABLE)
(4) Code List (CODES)
(5) Household Code Table (HHTABLE)

Each element of the VTABLE contains:
- ordinary variable name
- filename where the variable is located
- offset (in bytes) of the variable's position in the record
- length (in bytes) of the variable field

Each element of the CTABLE contains:
- coded variable name
- filename where variable is located
- offset (in bytes) of the variable's position in the record
- length (in bytes) of the variable field
- entry number in the code list where the codes for the variable starts
- entry number in the code list where the codes for the variable ends

Each element of the SVTABLE contains:
- special variable name
- filenames where the variable is located (up to 30 file names)
- corresponding offsets of the variable's position
- corresponding lengths of the variable's field

Each element of CODES contains:
- code name
- corresponding code

Each element of HHTABLE contains:
- household code

The following structures were declared in the retrieval program to store the query in decoded form.
(1) Ordinary Variable Attribute Structure (OVAR_ATTR) declared as array of structures

(2) Coded Variable Attribute Structure (CVAR_ATTR) declared as array of structures
(3) Special Variable Attribute Structure (SVAR_ATTR) declared as array of structures
(4) Array to store filenames involved in query (FNAME)
(5) Output Buffer Structure (BUF) declared as array of structures of array

Each structure of the OVAR_ATTR array of structures contains:

- filename where OVAR that is specified in query is located
- offset (in bytes) of OVAR's position in the record
- length (in bytes) of OVAR field
- function to be done on this variable; coded as
  - 0 for simple list or no function
  - 1 for COUNT
  - 2 for SUM
  - 3 for AVERAGE

Each structure of the CVAR_ATTR array of structures contains:

- filename where CVAR specified in the query is located
- offset (in bytes) of CVAR's position in the record
- length (in bytes) of CVAR
- condition coded as: 1 for < , 2 for <= , 3 for = , 4 for >= , 5 for > , 6 for ¬=
- value for this condition. If the WHERE option specifies the OVAR then the value is user supplied in the query, else the value is obtained from CODES list.

Each element of SVAR_ATTR contains:

- name of the special variable

Each element of the array FNAME contains:

- distinct filenames for OVARs in the query

Each structure of the output buffer (BUF) contains:

- household code
- concatenation of special variables
- variable values (up to 10)
- variable count (number of occurences of

this variable)

PART-3. The Retrieval Program.

The framework of the retrieval program is shown in Fig. 5-2. Each box represents one procedure.

The MAIN program calls 4 procedures:

1) SETTAB - to set up tables (enumerated in previous sub-section Part-2)
2) RECOG - to recognize the query and fill VAR-ATTR, BY-ATTR, COND-ATTR, CON-VAL
3) GETFN - to store the unique filenames of the variables involved in query
4) RETRVE - to retrieve all variables in query

RECOG calls 6 procedures:

1) NEXWRD - to get the query input byte by byte till blank ' ' is met indicating end of one word
2) RECBY - to recognize BY clause, i.e., the words following BY
3) RECWH - to recognize WHERE clause
4) VTSR - to search VINDEX for OVAR
5) CTSR - to search CINDEX for CVAR
6) CLSR - to search CLIST for code

RETRVE calls 8 procedures:

1) INITBUF - to initialize input buffer before variables for next HHCODE are retrieved
2) RRHH - to retrieve ordinary variables from given file for given HHCODE through alternate index path
3) GETOV - to retrieve ordinary and special variables from the record passed on by RRHH and to test the condition to see if the record is to be accepted
4) RTBV - to retrieve the special variable values from the given record
5) TEST - to test if the given record satisfies the condition
6) COMP - to test if condition in the WHERE clause is satisfied
7) SPVSR - to search SVINDEX for offset and length
8) RTVR - to accumulate values into BUF as per function specified

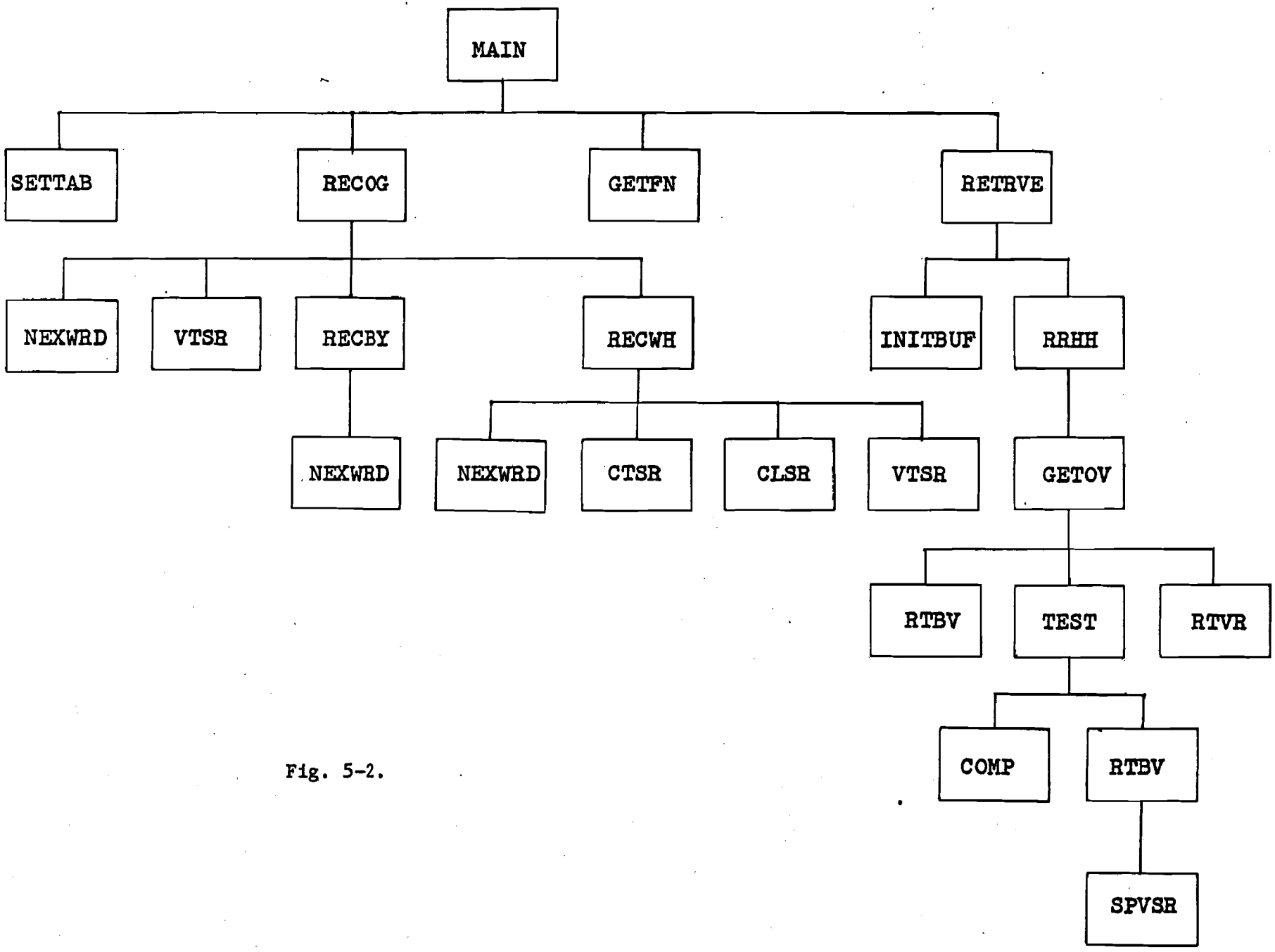
MAIN
SETTAB
RECOG
GETFN
RETRVE
NEXWRD
VTSR
RECBY
RECWH
INITBUF
RRHH
NEXWRD
NEXWRD
CTSR
CLSR
VTSR
GETOV
RTBV
TEST
RTVR
COMP
RTBV
SPVSR

Fig. 5-2.

Finally, control is returned all the way to RETRVE which moves the records from BUF to output file and then initializes the BUF for the next HHCODE.

The listing of the program is given in Appendix F. Illustrations of how the algorithm works is given in Appendix G.

The overall algorithm of the retrieval program is presented below:

```
Step 1 : read all the tables
Step 2 : recognize query and fill appropriate
         structure
Step 3 : for each HHCODE from the HHTABLE do;
Step 4 : for each filename of FNAME do;
Step 5 : get the special variables
Step 6 : if the ordinary variables satisfy the
         condition then
            do;
              get the ordinary variable, as per
                 function specified, into BUF
            end;
Step 7 : repeat from to Step 4
Step 8 : write BUF into OUTFILE file
Step 9 : repeat from step 3
Step 10: end;
```

The detail algorithm for Step 2 is as follows:

```
Step 2.1 : set OVAR_ATTR pointer to 0
Step 2.2 : get next word from input file
Step 2.3 : if word ¬= 'GET' issue error
Step 2.4 : get next word
Step 2.5 : increment pointer of OVAR_ATTR by 1
Step 2.6 : if word is one of the functions then
              do;
                get function value;
                put function value into the next
                   structure of current OVAR_ATTR
                get next word
              end
Step 2.7 : search word in variable table to get
           current OVAR filenames, offsets, and
           lengths
Step 2.8 : if not found then issue error
Step 2.9 : get next word
Step 2.10: if word = '.' then return
Step 2.11: if word = 'BY' then
              do;
                set switch to 1 to indicate 'BY'
```

```
                    preceeds 'WHERE'
                  recognize 'BY'
                  return
                end
Step 2.12: if word = 'WHERE' then
                do;
                  set  switch   to  1   to  indicate
                     'WHERE' preceeds 'BY'
                  recognize 'WHERE'
                  return
                end
Step 2.13: go to Step 2.5
Step 2.14: end
```

Detail Algorithm for Step 2.11 (recognize 'BY')

```
   2.11.1: set SVAR_ATTR pointer to 0
   2.11.2: get next word
   2.11.3: increment SVAR_ATTR pointer by 1
   2.11.4: put the word in SVAR_ATTR.NAME
   2.11.5: get next word
   2.11.6: if word = '.' then return
   2.11.7: if word = 'WHERE' then
                do;
                   recognize 'WHERE'
                end
   2.11.8: go to 2.11.3
   2.11.9: end
```

Detail Algorithm for Step 2.12 (recognize 'WHERE')

```
   2.12.1 : set CVAR_ATTR pointer to 0
   2.12.2 : get next word
   2.12.3 : search in CTABLE
   2.12.4 : if found
            then do;
                    get filename,offset,length,
                      start of code (SL),
                      and end of codes (EL)
                    get next word
                    if not '=' then
                      issue error; stop
                    get next word
                    search word in CODES (SL)
                      to CODES (EC)
                    if not found then
                      issue error; stop
                    put value in current CVAR_ATTR
                  end
            else do;
                    search word in VTABLE
                    if not found then
                      issue error; return
                    get filename, offset, length
```

```
                    get next word
                    recognize condition and
                      put appropriate codes
                    get next word
                    put value in current CVAR_ATTR
                end;
2.12.5 : get next record
2.12.6 : if word = '.' then return
2.12.7 : if word = 'BY' then recognize 'BY'
2.12.8 : go to 2.12.3
2.12.9 : end
```

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER WORK

6.1 Conclusions.

The work presented in this thesis used 33 household samples as test data. These samples were taken from Suphanburi, Thailand with the cooperation of International Rice Research Institute, Philippines and the Department of Agriculture, Thailand. Out of the 30 SURVEY forms supposed to be used for the Farm Mechanization Project, only 28 forms are being used in Thailand. For this thesis study, only 27 forms were used since the 28th form is the data identification form which does not serve the purposes of this study and data was not provided.

Although it was mentioned that mappings of FRK data items to corresponding SURVEY data items will be established, the author was advised through correspondence with IRRI (Phil) and subsequent approval of the thesis advisor to concentrate on SURVEY data and not to be concerned with FRK. As such, FRK data was not considered in this study.

The analysis of the forms showed that it is impractical to store the raw data in their original structure on direct access storage device. This is because of the multiple occurrences of information for the same entity on the same forms and also because of intervening blanks in the forms which occupy unnecessary storage space. In the creation of files the forms were thus split or combined as needed.

This splitting and combination of forms agreed with the independent work being carried out by IRRI in the Philippines. This was confirmed by the agricultural economist from IRRI who visited Thailand in June.

The files thus splitted were used to create VSAM files. The primary reason for choosing VSAM is the availability of alternate index which makes possible an easier and faster retrieval operation.

It was originally planned to use KSDS data sets. However, after a more in-depth study of VSAM facilities, it was found that ESDS will be better in terms of storage and without sacrificing access capability. With ESDS there is no need to determine primary keys and

candidate keys.

For this particular implementation, the VSAM ESDS data set has the following advantages over KSDS. KSDS requires primary key to uniquely identify a record. For most of the forms used in the project, several data items need to be concatenated to form the primary key that would uniquely identify the record. With ESDS however, no primary key is required. For building of alternate index, the primary key is again involved. More space is occupied for longer key in KSDS organization and also more space for alternate index. Since the records are never going to be accessed by primary keys, the choice of ESDS for this particular application results in considerable savings in space.

The query facility implemented is fairly powerful to retrieve a wide variety of data elements. The result of the query is stored in a file for further analysis. If the user rather wishes to see the retrieved data, he can write a short program to access this file. In the result file, the first three bytes is the HHCODE followed by 10-byte of grouping variables. Each retrieved data element occupies six bytes and starts from byte 14 of the result record.

6.2 Limitations & Recommendations for Further Study.

The number of variables that can be retrieved for one query command using this present system is ten. This number was so chosen because the author considers this to be sufficient number of variables for statistical purposes. If need arises then this number can be changed with modifications of the retrieval program.

The other areas that could be improved is the condition used for data retrieval in 'WHERE' clause. At present, if more than one condition is specified they are considered to be 'ANDed'. Further work can be done to allow a Boolean combination of conditions.

With the present implementation, the condition clause has effect on all of the variables to be retrieved. There can be cases where each variable needs to be chosen on a certain condition. Further work on the retrieval system is suggessted whereby the query format will have one condition clause for each variable.

BEST AVAILABLE COPY

APPENDIX A

VARIABLES INDEX TABLES

The following is a description of the index tables listed in this appendix.

SVINDEX TABLE includes variables that appear in more than one file. It contains the variable names, the filenames where they are found, the offset (in bytes) of variables' position in record, and the length (in bytes) of variable field.

VINDEX TABLE includes variables whose values are some measures, e.g., age, amount. Each variable appears in only one file. Each entry in the table contains the variable name, the filename where the variable is found, the offset (in bytes) of variable's position in the record, and the length (in bytes) of variable field.

CINDEX TABLE includes variables whose values are coded. Each variable appears in only one file. Each entry in the table contains the variable name, the filename where the variable is found, the offset (in bytes) of variable's position in the record, the length (in bytes) of variable field, the entry number in the codes list where codes for the variable starts, and the entry number where it ends.

SVINDEX Table. Index Table of Special Variables.

```
PARCEL-ID
912 913 931 932 933 939 940 941 944CROPDRAFPROD
010010010011010013012010010007007007
030303030303030303030303
TENURE-STATUS
912 931 932
036034024
020202
PARCEL-SIZE
912 913 932 941
014014019016
04040404
MACHINE-ID
915 936 937 938
010010011011
01010101
MACHINE-TYPE
915 936 937 938
011011012012
02020202
MACHINE-HP
915 936 937 938
013013014014
03030303
MACHINE-FUEL
936 937 938
016017017
010101
MACHINE-IMPLEMENT
936 937 938
017018018
020202
MACHINE-AGE
915 938
019057
0201
MAIN-CROP
913 939 940CROP
018010016019
03030303
CROP2
913 939 940CROP
021010016019
03030303
CROP3
913 939 940CROP
024010016019
03030303
```

VINDEX Table. Index Table of Ordinary Variables.

WORKMEMBER-AGE	911A0142
YRS-STUDY	911A0162
YRS-FARMING	911A0182
WORKDAYS-JAN	911A0232
WORKDAYS-FEB	911A0252
WORKDAYS-MAR	911A0272
WORKDAYS-APR	911A0292
WORKDAYS-MAY	911A0312
WORKDAYS-JUN	911A0332
WORKDAYS-JUL	911A0352
WORKDAYS-AUG	911A0372
WORKDAYS-SEP	911A0392
WORKDAYS-OCT	911A0412
WORKDAYS-NOV	911A0432
WORKDAYS-DEC	911A0452
INCOME-CASH	911A0476
INCOME-KIND	911A0536
NUM-WIVES-NOWORK	911B0132
NUM-ADULTMALE-NOWORK	911B0152
NUM-ADULTFEMALE-NOWORK	911B0172
NUM-CHILDREN-NOWORK	911B0192
WIFE1-AGE-AT-FIRT-MARRIAGE	930 0102
WIFE1-BORNCHILD-TOTAL	930 0122
WIFE1-DEADCHILD-TOTAL	930 0142
WIFE2-AGE-AT-FIRST-MARRIAGE	930 0162
WIFE2-BORNCHILD-TOTAL	930 0182
WIFE2-DEADCHILD-TOTAL	930 0202
RETURNING-MEMBERS	930 0222
HELP-LP	930 0241
HELP-PLANTING	930 0251
HELP-WEEDING	930 0261
HELP-HARVESTING	930 0271
HELP-TOTAL-DAYS	930 0283
YR-HHMECHANIZED	930 0312
PERMANENTLY-LEFT-AFTER-MECH	930 0332
SEASONWORKERS-AFTER-MECH	930 0551
YR-HHIRRIGATED	930 0562
PERMANENTLY-LEFT-AFTER-IRRIG	930 0582
SEASONWORKERS-AFTER-IRRIG	930 0801
LLMEMBER-AGE	943 0142
YR-VILLAGE-MECHANIZED	943 0172
ESTIMATED-PARCELVALUE	912 0246
LANDTAX	912 0306
FARMER-SHARE-PERCENT	912 0383
RENT-CASH	912 0416
RENT-KIND	912 0476
ORIGINAL-COST-PARCEL	912 0566
PARCEL-ANNUAL-MAINTENANCE-COST	912 0626
CONTRIBUTE-LPCOST	931 0172
CONTRIBUTE-SEEDCOST	931 0192
CONTRIBUTE-PLANTLABOR	931 0212

CONTRIBUTE-FERTILIZER	931 0232
CONTRIBUTE-PESTICIDE	931 0252
CONTRIBUTE-WEEDLABOR	931 0272
CONTRIBUTE-HARVEST	931 0292
CONTRIBUTE-THRESH	931 0312
YR-BEFORE-MECHANIZATION	932 0162
WET-AREA-CULTIVATED	932 0264
DRY1-AREA-CULTIVATED	932 0404
DRY2-AREA-CULTIVATED	932 0544
YR-LPMECH	933 0152
YR-CPCHANGE	933 0262
YR-WSCHANGE	933 0342
YR-TSCHANGE	933 0462
YR-LAND-RECLAIMED	941 0142
OLD-SALESVALUE	941 0216
NEW-SALESVALUE	941 0286
MAN-COUNT	941 0362
MAN-HRS	941 0384
MANHRS-VALUE	941 0425
ANIMAL-COUNT	941 0472
ANIMAL-HRS	941 0494
ANIMALHRS-VALUE	941 0535
MACHINE-COUNT	941 0581
MACHINE-HRS	941 0593
MACHINE-HRS-VALUE	941 0625
WORKUNITS	937 0254
CUSTOM-UNITCHARGE	937 0294
INCOME-CUSTOMWORK	937 0335
LITERS-USED	937 0385
COST-FUEL	937 0435
COST-OIL	937 0485
FILTERS	937 0531
COST-FILTER	937 0543
TIRE-REPAIRS	937 0572
COST-TIRE-REPAIRS	937 0593
HRS-FAMILY-LABOR-USED	937 0625
HRS-HIRED-LABOR-USED	937 0675
COST-HIRED-LABOR	937 0724
HRS-MACHINE-INACTIVE	938 0242
DAYS-JOB-DELAY	938 0262
PARTS-VALUE	938 0324
HRS-FAMILY-LABOR-REPAIR	938 0364
HRS-HIRED-LABOR-REPAIR	938 0404
COSTREPAIR-HIREDLABOR	938 0445
DISTANCE	938 0502
COST-MACHINE-TRANSPORT	938 0523
BREAKDOWN-FREQUENCY	938 0561
BUILDING-CAPACITY	918 0153
BUILDING-AGE	918 0212
BUILDING-MAINTENANCE-COST	918 0235
BUILDING-ORIGINAL-COST	918 0286
FARM-IMPLEMENT-COUNT	919 0143
FARM-IMPLEMENT-VALUE	919 0176
LANDCOUNT	934L0121

LAND-SIZE	934L0144
LAND-SALESVALUE	934L0186
VEHICLE-COUNT	934V0121
VEHICLE-AGE	934V0142
VEHICLE-ORIGINAL-COST	934V0166
VEHICLE-MAINTENANCE-COST	934V0225
HOMEGOODS-AGE	935 0142
HOMEGOODS-ORIGINAL-COST	935 0164
HOMEGOODS-ESTIMATED-LIFE	935 0202
PRODUCT-TOTAL	939 0185
PRODUCT-VALUE	939 0236
PAYHARVESTER-KG	939 0325
PAYHARVESTER-VALUE	939 0375
PAYLANDLORD-KG	939 0435
PAYLANDLORD-VALUE	939 0485
PAYDEBTS-KG	939 0545
PAYDEBTS-VALUE	939 0595
SOLD-KG	939 0655
SOLD-VALUE	939 0705
RETAIN-KG	939 0775
RETAIN-VALUE	939 0825
OTHERS-KG	939 0895
OTHERS-VALUE	939 0946
BYPROD-VALUE	939 1015
CROP1-RETAIN-KG	939 1125
CROP2-RETAIN-KG	939 1175
CROP1-COSTPRODUCT	939 1225
CROP2-COSTPRODUCT	939 1275
SALES-FREQUENCY	940 0191
KG-SOLD	940 0216
PRICE-PER-KG	940 0284
INCOME-SALES	940 0326
MARKET-DISTANCE	940 0393
COST-PRODUCE-TRANSPORT	940 0435
EXT-MEETINGS-ATTENDED	942E0191
OFFICEVISIT-FREQUENCY	942E0201
HOUSEVISIT-FREQUENCY	942E0211
YR-TRAINING	942T0152
COUNT-ORGANIZATION-MEMBERSHIP	94200121
ORG-MEETINGS-HELD	94200191
ORG-MEETINGS-ATTENDED	94200201
DRAFT-ANIMAL-COUNT	945D0142
DELAY-DAYS	945D0183
NOWORK-DAYS	945D0212
JOINT-SHARE-PERCENT	915 0212
COST-ACQUISITION	915 0236
SALVAGE-VALUE	915 0296
ANNUAL-FIXED-COST	915 0355
MACH-ANNUAL-MAINTENANCE-COST	915 0405
FUEL-CONSUMED	915 0454
FUEL-PRICE-PER-LITER	915 0494
OIL-CONSUMED	915 0534
OIL-PRICE-PER-LITER	915 0574
OWN-USE-HRS-SEASON	915 0614

RENT-COST	915 0665
RENT-INCOME	915 0716
DRAFT-COUNT1	916 0223
DRAFT-VALUE1	916 0256
DRAFT-COUNT2	916 0343
DRAFT-VALUE2	916 0376
DRAFT-COUNT3	916 0463
DRAFT-VALUE3	916 0496
DRAFY-COUNT4	916 0583
DRAFT-VALUE4	916 0616
DRAFT-COUNT5	916 0703
DRAFT-VALUE5	916 0736
PROD-COUNT1	917 0223
PROD-VALUE1	917 0256
PROD-COUNT2	917 0343
PROD-VALUE2	917 0376
PROD-COUNT3	917 0463
PROD-VALUE3	917 0496
PROD-COUNT4	917 0583
PROD-VALUE4	917 0616
PROD-COUNT5	917 0703
PROD-VALUE5	917 0736
PASTMONTHS-INCURRED	920 0132
DURATION-MONTHS	920 0152
INTEREST-PERCENT	920 0175
AMOUNT-LIABILITY	920 0236
RICE-VARIETY	913 0302

CINDEX Table. Index Table of Coded Variables.

HEADSEX	911A0111003004
WORKMEMBER-TYPE	911A0122639643
OTHERWORK-TYPE	911A0202005034
EMPLOYMENT-TYPE	911A0221034036
HH-NOT-MECHANIZED	930 0312037037
MEMBER1-PERMANENTLY-LEFTMECH	930 0352639643
MEMBER2-PERMANENTLY-LEFTMECH	930 0372639643
MEMBER3-PERMANENTLY-LEFTMECH	930 0392639643
MEMBER4-PERMANENTLY-LEFTMECH	930 0412639643
MEMBER5-PERMANENTLY-LEFTMECH	930 0432639643
MEMBER1-REASONMECH	930 0452038043
MEMBER2-REASONMECH	930 0472038043
MEMBER3-REASONMECH	930 0492038043
MEMBER4-REASONMECH	930 0512038043
MEMBER5-REASONMECH	930 0532038043
HH-NOT-IRRIGATED	930 0562044044
MEMBER1-PERMANENTLY-LEFTIRRIG	930 0602639643
MEMBER2-PERMANENTLY-LEFTIRRIG	930 0622639643
MEMBER3-PERMANENTLY-LEFTIRRIG	930 0642639643
MEMBER4-PERMANENTLY-LEFTIRRIG	930 0662639643
MEMBER5-PERMANENTLY-LEFTIRRIG	930 0682639643
MEMBER1-REASONIRRIG	930 0702038043
MEMBER2-REASONIRRIG	930 0722038043
MEMBER3-REASONIRRIG	930 0742038043
MEMBER4-REASONIRRIG	930 0762038043
MEMBER5-REASONIRRIG	930 0782038043
LLMEMBER-TYPE	943 0122639643
VILLAGE-NOT-MECHANIZED	943 0172045045
PASTWORK1-TYPE	943 0192005033
PASTWORK2-TYPE	943 0212005033
PASTWORK3-TYPE	943 0232005033
PRESWORK1-TYPE	943 0252005033
PRESWORK2-TYPE	943 0272005033
PRESWORK3-TYPE	943 0292005033
REASON-JOBSHIFT	943 0311046051
PREFWORK1-TYPE	943 0322005033
PREFWORK2-TYPE	943 0342005033
REASON-WORKPREFERENCE	943 0362052056
SOIL-TEXTURE	912 0181057063
SOIL-COLOR	912 0191064068
TOPOGRAPHY	912 0201069076
RESTRICTING-FACTORS	912 0212077086
IRRIGATION-SOURCE	912 0231087092
PARCEL-IMPROVEMENT1	912 0531101106
PARCEL-IMPROVEMENT2	912 0541101106
LANDLORD-CONTRIBUTION	931 0161001002
YR-CODE	932 0151111112
PARCEL-CULTIVATED	932 0181001002
WATERCONTROL-TYPE	932 0231113116
WET-IRRIGSUPPLY-ADEQUACY	932 0301117123
WET-MAINCROP	932 0313531627

WET-CROP2	932	0343531627
WET-CROP3	932	0373531627
DRY1-IRRIGSUPPLY-ADEQUACY	932	0441117123
DRY1-MAINCROP	932	0453531627
DRY1-CROP2	932	0483531627
DRY1-CROP3	932	0513531627
DRY2-IRRIGSUPPLY-ADEQUACY	932	0581117123
DRY2-MAINCROP	932	0593531627
DRY2-CROP2	932	0623531627
DRY2-CROP3	932	0653531627
LPMECHANIZED	933	0141001002
LPMACHINE-TYPE	933	0172690694
REASON1-LPMACH-USE	933	0192124138
REASON2-LPMACH-USE	933	0212124138
REASON3-LPMACH-USE	933	0232124138
CPCHANGE	933	0251001002
CAN-PLANT-ANOTHERCROP-IRRIG	933	0281001002
CAN-PLANT-ANOTHERCROP-MECH	933	0291001002
ANOTHERCROP-MORE-PROFITABLE	933	0301001002
FORMERCROP-LACKWATER	933	0311001002
OTHER-REASON-CPCHANGE	933	0321001002
WSCHANGE	933	0331001002
CONSTRUCTION-CONCRETE-DAM	933	0361001002
CONSTRUCTION-DIVERSION-DAM	933	0371001002
CONSTRUCTION-WATERGATES	933	0381001002
CONSTRUCTION-NEW-CANALS	933	0391001002
REHABILITATION-OLD-SYSTEM	933	0401001002
INSTALLATION-SYSTEM-PUMP	933	0411001002
INSTALLATION-FARM-PUMP	933	0421001002
POOR-IRRIG-SUPPLY	933	0431001002
OTHER-REASON-WSCHANGE	933	0441001002
TSCHANGE	933	0451001002
LANDREFORM	933	0481001002
NOMONEY-TO-ACQUIRE-LAND	933	0491001002
OTHER-REASON-TSCHANGE	933	0501001002
LANDTYPE-BEFORE-RECLAMATION	941	0201139144
LANDTYPE-AFTER-RECLAMATION	941	0271145149
POWERUSED-RECLAMATION	941	0342150156
RECLAMATION-POSSIBLE-NO-MACH	941	0671001002
REASON1-RECLAMATION-MACH-USE	941	0681157161
REASON2-RECLAMATION-MACH-USE	941	0691157161
RECLAMATION-MACHINE-TYPE	941	0702690694
ACTUALWEEK-PLOUGH	944	0142255320
ACTUALDAY-PLOUGH	944	0161321328
IDEALWEEK-PLOUGH	944	0172255320
IDEALDAY-PLOUGH	944	0191321328
REASON-DELAY-PLOUGH	944	0201162168
ACTUALWEEK-HARROW	944	0212255320
ACTUALDAY-HARROW	944	0231321328
IDEALWEEK-HARROW	944	0242255320
IDEALDAY-HARROW	944	0261321328
REASON-DELAY-HARROW	944	0271162168
ACTUALWEEK-PLANT	944	0282255320
ACTUALDAY-PLANT	944	0301321328

IDEALWEEK-PLANT	944	0312255320
IDEALDAY-PLANT	944	0331321328
REASON-DELAY-PLANT	944	0341162168
ACTUALWEEK-HARVEST	944	0352255320
ACTUALDAY-HARVEST	944	0371321328
IDEALWEEK-HARVEST	944	0382255320
IDEALDAY-HARVEST	944	0401321328
REASON-DELAY-HARVEST	944	0411162168
ACTIVITY-WHERE-MACHINE-USED	937	0203803844
FARM-WHERE-MACHINE-USED	937	0231210212
MEASURE-UNIT-USED	937	0241213219
MACHINE-ACCIDENT-TYPE	937	0771220223
MACHINE-ACCIDENT-SERIOUSNESS	937	0781224227
BREAKDOWN-TYPE	938	0202228254
BREAKDOWN-SERIOSNESS	938	0221329330
BREAKDOWNMACHINE-REPAIR	938	0231001002
BREAKDOWN-REPAIR-DELAY	938	0281001002
BREAKDOWN-REPAIR-DELAY-REASON1	938	0291331336
BREAKDOWN-REPAIR-DELAY-REASON2	938	0301331336
ACTION-TO-GET-SPAREPARTS	938	0311337339
PLACE-OF-REPAIR	938	0491340346
REPAIR-SATISFACTORY	938	0551001002
BREAKDOWN-WEEK	938	0582255320
BREAKDOWN-DAY	938	0601321328
BUILDING-TYPE	918	0122347353
BUILDING-MATERIAL	918	0141354356
BUILDING-ENTERPRISE	918	0183531638
FARM-HAND-IMPLEMENT	919	0122192209
LANDTYPE	934L	0131360362
VEHICLE-TYPE	934V	0131363370
HOMEGOODS-TYPE	935	0122371391
PAY-HARVESTER	939	0291001002
PAY-ARRANGEMENT	939	0302392398
PAY-LANDLORD	939	0421001002
PAY-DEBTS	939	0531001002
SOLD-PRODUCT	939	0641001002
RETAIN-PRODUCT	939	0751001002
RETAIN-PURPOSE	939	0761399405
OTHER-PRODUCT	939	0871001002
OTHER-PURPOSE	939	0881399405
BYPRODUCT-TYPE	939	1001406410
BY-CROP1	939	1063531627
BY-CROP2	939	1093531627
DATE-SALES	940	0201411417
PRODUCT-QUALITY	940	0271418422
MARKET-LOCATION	940	0381423427
MARKET-TRANSPORTATION	940	0421428429
EXTSERVICE-OFFICE-LOCATION	942E	0121430432
PLANTING-KNOWLEDGE	942E	0131001002
FERTILIZER-KNOWLEDGE	942E	0141001002
PESTCONTROL-KNOWLEDGE	942E	0151001002
WATERCONTROL-KNOWLEDGE	942E	0161001002
HYV-KNOWLEDGE	942E	0171001002
OTHER-KNOWLEDGE	942E	0181001002

RATE-WORKER	942E0221428429
FARMPROBLEMS-CONSULT1	942E0231433438
FARMPROBLEMS-CONSULT2	942E0241433438
TRAINING-MACHINE-TYPE	942T0122690710
TRAINER	942T0141439442
ORGANIZATION-TYPE	94200131443444
ORGANIZATION-ADVANTAGE1	94200141445450
ORGANIZATION-ADVANTAGE2	94200151445450
ORGANIZATION-DISADVANTAGE1	94200161451454
ORGANIZATION-DISADVANTAGE2	94200171451454
MEMBERSHIP-REQUIRED	94200181001002
DRAFT-ANIMAL-WITH-DISEASE	945D0122512516
DISEASE-TYPE	945D0161517525
WORK-DELAY	945D0171001002
DRAFT-ANIMAL-WITH-ACCIDENT	945A0122512516
ACCIDENT-TYPE	945A0141526530
ACCIDENT-SERIOSNESS	945A0151224227
MIX-PTR	913 0273531627
PLANTING-METHOD	913 0322455459
PLANTING-MONTH	913 0342460471
PLANTING-WEEK	913 0361472475
ENTERPRISE-MACHINE-USED	915 0163531638
RENTAL-USE	915 0651509511
DRAFT-TYPE	916 0112512516
DRAFT-BREED	916 0132476479
DRAFT-MGMT1	916 0152480486
DRAFT-MGMT2	916 0172480486
DRAFT-SUB1	916 0193487501
DRAFT-SUB2	916 0313487501
DRAFT-SUB3	916 0433487501
DRAFT-SUB4	916 0553487501
DRAFT-SUB5	916 0673487501
PROD-TYPE	917 0112899908
PROD-BREED	917 0132476479
PROD-MGMT	917 0152480486
PROD-PURPOSE1	917 0171502508
PROD-PURPOSE2	917 0181502508
PROD-SUB1	917 0193487501
PROD-SUB2	917 0313487501
PROD-SUB3	917 0433487501
PROD-SUB4	917 0553487501
PROD-SUB5	917 0673487501
LIABILITY-TYPE	920 0121861864
LIABILITY-FORM	920 0221865868
FINANCING-SECURITY	920 0291869875
LIABILITY-PURPOSE1	920 0301876883
LIABILITY-PURPOSE2	920 0311876883
LIABILITY-SOURCE1	920 0321884891
LIABILITY-SOURCE2	920 0331884891
DECISION-MAKER1	936 0191892898
DECISION-MAKER2	936 0201892898
PURCHASE-ADVANTAGE1	936 0212124138
PURCHASE-ADVANTAGE2	936 0231124138
PURCHASE-ADVANTAGE3	936 0252124138

APPENDIX B

SAMPLE PROGRAM FOR PHASE 1

A total of 27 programs were run for Phase 1. Records were split or combined depending upon the particular form being processed. Details are given in the analysis of forms, Chapter 4. This phase is in preparation for the creation of VSAM data sets, using the raw data from the Department of Agriculture, Thailand. The commands issued when creating the CMS files using VM facility are given in the following CREATE EXEC file.

```
FI INFILE DISK TDRU DATA D(RECFM F LRECL 80
FI RUCROP DISK RUCROP DATA D(RECFM F LRECL 32
FI RUDRAF DISK RUDRAF DATA D(RECFM F LRECL 32
FI RUPROD DISK RUPROD DATA D(RECFM F LRECL 32
PLIC RU3 PLI A
SORT RUCROP DATA D SRTCROP DATA D
ERASE RUCROP DATA D
SORT RUDRAF DATA D SRTDRAF DATA D
ERASE RUDRAF DATA D
SORT RUPROD DATA D SRTPROD DATA D
ERASE RUPROD DATA D
```

The corresponding program run by the above commands are given below:

```
RU:PROC OPTIONS (MAIN);
   DCL RUCROP FILE RECORD ENV (RECSIZE(32));
   DCL RUDRAF FILE RECORD ENV (RECSIZE(32));
   DCL RUPROD FILE RECORD ENV (RECSIZE(32));
   DCL INFILE FILE RECORD ENV (RECSIZE(80));
   DCL AMT    CHAR(6)       INIT((6)' ');
   DCL KEY    CHAR(21);
   DCL 1  INPUT_RU,
         2  CODE1                    CHAR(6),
         2  IO_DATA(3)               CHAR(21),
         2  CODE2                    CHAR(11);
   DCL 1  RU_STR,
         2  KEY,
           3  PROJ_CODE              PIC '999',
           3  HH_CODE                PIC '999',
           3  PARCEL_ID              PIC '999',
           3  ENTERPRISE             PIC '999',
           3  TYPE                   PIC '999',
```

```
          3  ACTIVITY                PIC '999',
          3  DATE                    PIC '999',
       2  AMOUNT                     FLOAT DEC(5),
       2  DATA,
          3  IND_CASH                PIC '9',
          3  VALUE                   PIC '(6)9';
   OPEN FILE (INFILE) INPUT;
   OPEN FILE (RUCROP) OUTPUT;
   OPEN FILE (RUDRAF) OUTPUT;
   OPEN FILE (RUPROD) OUTPUT;
   ON ENDFILE (INFILE) STOP;
   DO WHILE (6=6);
      READ FILE (INFILE) INTO (INPUT_RU);
      GET STRING (CODE1) EDIT (ENTERPRISE, PARCEL_ID)
                              (P'999', P'999');
      GET STRING (CODE2) EDIT (HH_CODE, PROJ_CODE)
                              (X(5),P'999', P'999');
      /* GET THE 3 TYPES OF I/O CONTAINED IN ONE INFILE RECORD */
      /*      FILERU WILL HAVE 1 TYPE OF I/O IN ONE RECORD     */
      DO I = 1 TO 3;
         GET STRING (IO_DATA(I)) EDIT (TYPE)(P'999');
         IF TYPE = 0 THEN LEAVE;
         GET STRING (IO_DATA(I)) EDIT(ACTIVITY,DATE,AMT,DATA)
                            (X(3),P'999', P'999', A(5), P'9', P'(6)9');
         GET STRING (AMT) EDIT (AMOUNT)(F(5,0));
         KEY = PROJ_CODE || HH_CODE || PARCEL_ID || TYPE || ACTIVITY
               || DATE || ENTERPRISE;
         SELECT;
           WHEN (ENTERPRISE < 400)
                WRITE FILE(RUCROP) FROM (RU_STR);
           WHEN (ENTERPRISE > 500 & ENTERPRISE < 504)
                WRITE FILE(RUDRAF) FROM (RU_STR);
           OTHERWISE
                WRITE FILE(RUPROD) FROM (RU_STR);
         END;
      END;
   END;
END RU;
```

APPENDIX C

DISK INITIALIZATION

VM/370 does virtual disk initialization with the OS utility program IBCDASDI. IBCDASDI formats disk volumes for OS, DOS, and VSAM under CMS usage. The following are the procedures for disk initialization using CMS facility.

Step 1 : A CMS file, INITDISK JOB A, is created which contains the following required IBCDASDI control cards:

```
JOB
MSG   TODEV=3215,TOADDR=009
DADEF TODEV=3340,TOADDR=200,VOLID=SCRATCH,
      cylno=20
VLD   NEWVOLID=VSAM01
VTOCD STRTADR=1,EXTENT=2
END
```

where:

JOB	indicates the beginning of job
MSG	defines output device for operator messages
DADEF	defines DASD device to be initialized
VLD	labels the volume and allocates space for additional labels
VTOCD	contols the location of the volume-table-of-contents
END	ends an IBSDASDI job

note : word following '=' sign is user supplied

Step 2 : The IBCDASDI program, distributed with filename IPL and filetype IBCDASDI, should be spooled to the user's virtual card reader. The IBCDASDI program is invoked by the following commands:

```
CLOSE C HOLD
PURGE RDR ALL
SP PU TO * CL A CON NOHOLD
PU IBCDASDI TXT Y (NOH
PU INITDISK JOB A (NOH
SP PU CLOSE
IPL 00C
```

66

Step 3 : When the program is loaded and waiting for input, signal attention from the VM console device. A message is sent to the user's console device to define input device. Definition is then given as follows:

INPUT=3505,00c

where 3505 is the device type containing the contol statements and 00c is the device address

After disk initialization is finished, bring up the CMS through IPL CMS command.

APPENDIX D

DEFINITION OF MASTER CATALOG AND CLUSTERS

When a catalog is defined, the attributes for the catalog as a whole and for the catalog's components are specified.

The format of the DEFINE command for master catalog is as follows:

```
DEFINE MASTERCATALOG (NAME(entryname) VOL(vol) -
                      CYL(ncyl) FILE(IJSYSCT))
```

For an entry-sequenced cluster, the format of the DEFINE CLUSTER is as follows:

```
DEFINE CLUSTER (NAME (entryname) VOL (vol) -
                NONINDEXED REC(primary secondary) -
                RECSZ(min max) FILE(ddname)
```

The AMS control statements are provided in a CMS file with file type AMSERV.

For this study, the master catalog was defined as follows:

```
DEFINE MASTERCATALOG (NAME(IRRI) VOL(VSAM01) CYL(4) -
                      FILE(IJSYSCT))
```

The commands to define master catalog:

```
DLBL IJSYSCT F DSN IRRI (PERM EXTENT
12 228
AMSERV DEFMAS
```

where F is the filemode of the disk formatted as OS
IRRI is the master catalog name
228 is the number of tracks free for VSAM data space computed as (12*(ncyl-1)) where ncyl is the total number of cylinders in the OS disk
DEFMAS is the filename if the CMS file which contains the commands.

For the clusters, a total of 32 VSAM clusters were defined. The following are the contents of a CMS file with filetype AMSERV.

```
DEFINE CLUSTER (NAME (CL911A) VOL (VSAM01) -
       NONINDEXED REC(89,10) RECSZ(58 58) FILE(DD911A))
DEFINE CLUSTER (NAME (CL911B) VOL (VSAM01) -
       NONINDEXED REC(33,10) RECSZ(20 20) FILE(DD911B))
DEFINE CLUSTER (NAME (CL912) VOL (VSAM01) -
       NONINDEXED REC(57,10) RECSZ(67 67) FILE(DD912))
DEFINE CLUSTER (NAME (CL913) VOL (VSAM01) -
       NONINDEXED REC(79,10) RECSZ(37 37) FILE(DD913))
DEFINE CLUSTER (NAME (CL915) VOL (VSAM01) -
       NONINDEXED REC(23,10) RECSZ(76 76) FILE(DD915))
DEFINE CLUSTER (NAME (CL916) VOL (VSAM01) -
       NONINDEXED REC(15,10) RECSZ(78 78) FILE(DD916))
DEFINE CLUSTER (NAME (CL917) VOL (VSAM01) -
       NONINDEXED REC(56,10) RECSZ(78 78) FILE(DD917))
DEFINE CLUSTER (NAME (CL918) VOL (VSAM01) -
       NONINDEXED REC(90,10) RECSZ(33 33) FILE(DD918))
DEFINE CLUSTER (NAME (CL919) VOL (VSAM01) -
       NONINDEXED REC(465,10) RECSZ(22 22) FILE(DD919))
DEFINE CLUSTER (NAME (CL920) VOL (VSAM01) -
       NONINDEXED REC(20,10) RECSZ(34 34) FILE(DD920))
DEFINE CLUSTER (NAME (CL930) VOL (VSAM01) -
       NONINDEXED REC(33,10) RECSZ(80 80) FILE(DD930))
DEFINE CLUSTER (NAME (CL931) VOL (VSAM01) -
       NONINDEXED REC(12,10) RECSZ(33 33) FILE(DD931))
DEFINE CLUSTER (NAME (CL932) VOL (VSAM01) -
       NONINDEXED REC(101,10) RECSZ(67 67) FILE(DD932))
DEFINE CLUSTER (NAME (CL933) VOL (VSAM01) -
       NONINDEXED REC(55,10) RECSZ(51 51) FILE(DD933))
DEFINE CLUSTER (NAME (CL934L) VOL (VSAM01) -
       NONINDEXED REC(26,10) RECSZ(23 23) FILE(DD934L))
DEFINE CLUSTER (NAME (CL934V) VOL (VSAM01) -
       NONINDEXED REC(55,10) RECSZ(26 26) FILE(DD934V))
DEFINE CLUSTER (NAME (CL935) VOL (VSAM01) -
       NONINDEXED REC(154,10) RECSZ(21 21) FILE(DD935))
DEFINE CLUSTER (NAME (CL936) VOL (VSAM01) -
       NONINDEXED REC(23,10) RECSZ(26 26) FILE(DD936))
DEFINE CLUSTER (NAME (CL937) VOL (VSAM01) -
       NONINDEXED REC(44,10) RECSZ(78 78) FILE(DD937))
DEFINE CLUSTER (NAME (CL938) VOL (VSAM01) -
       NONINDEXED REC(40,10) RECSZ(61 61) FILE(DD938))
DEFINE CLUSTER (NAME (CL939) VOL (VSAM01) -
       NONINDEXED REC(41,10) RECSZ(131 131) FILE(DD939))
DEFINE CLUSTER (NAME (CL940) VOL (VSAM01) -
       NONINDEXED REC(29,10) RECSZ(48 48) FILE(DD940))
DEFINE CLUSTER (NAME (CL941) VOL (VSAM01) -
       NONINDEXED REC(24,10) RECSZ(71 71) FILE(DD941))
DEFINE CLUSTER (NAME (CL942E) VOL (VSAM01) -
       NONINDEXED REC(30,10) RECSZ(25 25) FILE(DD942E))
DEFINE CLUSTER (NAME (CL942T) VOL (VSAM01) -
       NONINDEXED REC(3,1) RECSZ(17 17) FILE(DD942T))
DEFINE CLUSTER (NAME (CL942O) VOL (VSAM01) -
       NONINDEXED REC(16,10) RECSZ(21 21) FILE(DD942O))
DEFINE CLUSTER (NAME (CL943) VOL (VSAM01) -
       NONINDEXED REC(6,3) RECSZ(36 36) FILE(DD943))
```

```
DEFINE CLUSTER (NAME (CL944) VOL (VSAM01) -
       NONINDEXED REC(54,10) RECSZ(42 42) FILE(DD944))
DEFINE CLUSTER (NAME (CL945D) VOL (VSAM01) -
       NONINDEXED REC(2,1) RECSZ(23 23) FILE(DD945D))
DEFINE CLUSTER (NAME (CLCROP) VOL (VSAM01) -
       NONINDEXED REC(1838,10) RECSZ(32 32) FILE(DDCROP))
DEFINE CLUSTER (NAME (CLDRAF) VOL (VSAM01) -
       NONINDEXED REC(1046,10) RECSZ(32 32) FILE(DDDRAF))
DEFINE CLUSTER (NAME (CLPROD) VOL (VSAM01) -
       NONINDEXED REC(828,10) RECSZ(32 32) FILE(DDPROD))
```

The commands issued are:

```
DLBL DD911A F DSN CL911A(VSAM
DLBL DD911B F DSN CL911B(VSAM
DLBL DD912 F DSN CL912(VSAM
DLBL DD913 F DSN CL913(VSAM
DLBL DD915 F DSN CL915(VSAM
DLBL DD916 F DSN CL916(VSAM
DLBL DD917 F DSN CL917(VSAM
DLBL DD918 F DSN CL918(VSAM
DLBL DD919 F DSN CL919(VSAM
DLBL DD920 F DSN CL920(VSAM
DLBL DD930 F DSN CL930(VSAM
DLBL DD931 F DSN CL931(VSAM
DLBL DD932 F DSN CL932(VSAM
DLBL DD933 F DSN CL933(VSAM
DLBL DD934L F DSN CL934L(VSAM
DLBL DD934V F DSN CL934V(VSAM
DLBL DD935 F DSN CL935(VSAM
DLBL DD936 F DSN CL936(VSAM
DLBL DD937 F DSN CL937(VSAM
DLBL DD938 F DSN CL938(VSAM
DLBL DD939 F DSN CL939(VSAM
DLBL DD940 F DSN CL940(VSAM
DLBL DD941 F DSN CL941(VSAM
DLBL DD942E F DSN CL942E(VSAM
DLBL DD942T F DSN CL942T(VSAM
DLBL DD942O F DSN CL942O(VSAM
DLBL DD943 F DSN CL943(VSAM
DLBL DD944 F DSN CL944(VSAM
DLBL DD945D F DSN CL945D(VSAM
DLBL DD945A F DSN CL945A(VSAM
AMSERV DEFCLUS
```

APPENDIX E

DEFINITION OF ALTERNATE INDEX

When an alternate index is defined, the attributes for the alternate index as a whole and the alternate index's components are specified. The format for the DEFINE ALTERNATEINDEX command is:

```
DEFINE ALTERNATEINDEX (NAME(entryname) -
                       RELATE(base clustername) -
                       VOL (vol) -
                       KEYS(length offset) _
                       NONUNIQUEKEY _
                       RECORDS(primary secondary) -
                       RECSZ(average maximum)
```

The average record size of the alternate index for ESDS is computed as follows:

RECSZ = 5 + AIXKL + n * 4

where AIXKL is the alternate keylength and n is the number of data records containing the same alternate key value when NONUNIQUEKEY is specified.

The BLDINDEX command is used to build an alternate index. Prerequisites to this are the definition and loading of base clusters, and definition of alternate index relating it to the base cluster. When an alternate index is built by BLDINDEX processing, AMS opens the base cluster to sequentially read the data records, sorts the information obtained from the data records, and builds the alternate index records. The format of the BLDINDEX command is:

```
BLDINDEX INFILE(dname1) -
         OUTFILE(dname2)
```

The DEFINEPATH command is used to establish the relationship, and the path between an alternate index and its base cluster. The base cluster and its alternate index must be defined before they can be related. The format of the DEFINEPATH command is:

```
DEFINE PATH (NAME(entryname)
             PATHENTRY(aixentryname))
```

The following are the definitions used in this study. They are contained in a CMS file with filetype AMSERV.

```
DEFINE AIX ( NAME(AIX911A) RELATE(CL911A) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(89 10) RECSZ(28 48) )
BLDINDEX INFILE (D1911A) -
         OUTFILE (D2911A)
DEFINE PATH ( NAME(PATH911A) -
              PATHENTRY(AIX911A) )
DEFINE AIX ( NAME(AIX911B) RELATE(CL911B) -
             VOL(VSAM01) KEYS(6 9) NONUNIQUEKEY -
             RECORDS(33 10) RECSZ(12 12) )
BLDINDEX INFILE (D1911B) -
         OUTFILE (D2911B)
DEFINE PATH ( NAME(PATH911B) -
              PATHENTRY(AIX911B) )
DEFINE AIX ( NAME(AIX912) RELATE(CL912) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(57 10) RECSZ(20 28) )
BLDINDEX INFILE (D1912) -
         OUTFILE (D2912)
DEFINE PATH ( NAME(PATH912) -
              PATHENTRY(AIX912) )
DEFINE AIX ( NAME(AIX913) RELATE(CL913) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(79 10) RECSZ(28 48) )
BLDINDEX INFILE (D1913) -
         OUTFILE (D2913)
DEFINE PATH ( NAME(PATH9313 -
              PATHENTRY(AIX913) )
DEFINE AIX ( NAME(AIX915) RELATE(CL915) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(23 10) RECSZ(20 28) )
BLDINDEX INFILE (D1915) -
         OUTFILE (D2915)
DEFINE PATH ( NAME(PATH915) -
              PATHENTRY(AIX915) )
DEFINE AIX ( NAME(AIX916) RELATE(CL916) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(15 10) RECSZ(20 28) )
BLDINDEX INFILE (D1916) -
         OUTFILE (D2916)
DEFINE PATH ( NAME(PATH916) -
              PATHENTRY(AIX916) )
DEFINE AIX ( NAME(AIX917) RELATE(CL917) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(56 10) RECSZ(20 28) )
BLDINDEX INFILE (D1917) -
         OUTFILE (D2917)
DEFINE PATH ( NAME(PATH917) -
              PATHENTRY(AIX917) )
DEFINE AIX ( NAME(AIX918) RELATE(CL918) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
```

```
             RECORDS(90 10) RECSZ(28 48) )
BLDINDEX INFILE (D1918) -
         OUTFILE (D2918)
DEFINE PATH ( NAME(PATH918) -
              PATHENTRY(AIX918) )
DEFINE AIX ( NAME(AIX919) RELATE(CL919) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(465 10) RECSZ(68 108) )
BLDINDEX INFILE (D1919) -
         OUTFILE (D2919)
DEFINE PATH ( NAME(PATH919) -
              PATHENTRY(AIX919) )
DEFINE AIX ( NAME(AIX920) RELATE(CL920) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(20 10) RECSZ(20 28) )
BLDINDEX INFILE (D1920) -
         OUTFILE (D2920)
DEFINE PATH ( NAME(PATH920) -
              PATHENTRY(AIX920) )
DEFINE AIX ( NAME(AIX930) RELATE(CL930) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(33 10) RECSZ(12 12) )
BLDINDEX INFILE (D1930) -
         OUTFILE (D2930)
DEFINE PATH ( NAME(PATH930) -
              PATHENTRY(AIX930) )
DEFINE AIX ( NAME(AIX931) RELATE(CL931) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(12 10) RECSZ(20 28) )
BLDINDEX INFILE (D1931) -
         OUTFILE (D2931)
DEFINE PATH ( NAME(PATH931) -
              PATHENTRY(AIX931) )
DEFINE AIX ( NAME(AIX932) RELATE(CL932) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(101 10) RECSZ(28 48) )
BLDINDEX INFILE (D1932) -
         OUTFILE (D2932)
DEFINE PATH ( NAME(PATH932) -
              PATHENTRY(AIX932) )
DEFINE AIX ( NAME(AIX933) RELATE(CL933) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(55 10) RECSZ(20 28) )
BLDINDEX INFILE (D1933) -
         OUTFILE (D2933)
DEFINE PATH ( NAME(PATH933) -
              PATHENTRY(AIX933) )
DEFINE AIX ( NAME(AIX934L) RELATE(CL934L) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(26 10) RECSZ(20 28) )
BLDINDEX INFILE (D1934L) -
         OUTFILE (D2934L)
DEFINE PATH ( NAME(PATH934L) -
              PATHENTRY(AIX934L) )
```

```
DEFINE AIX ( NAME(AIX934V) RELATE(CL934V) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(55 10) RECSZ(28 68) )
BLDINDEX INFILE (D1934V) -
         OUTFILE (D2934V)
DEFINE PATH ( NAME(PATH934V) -
              PATHENTRY(AIX934V) )
DEFINE AIX ( NAME(AIX935) RELATE(CL935) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(154 10) RECSZ(28 68) )
BLDINDEX INFILE (D1935) -
         OUTFILE (D2935)
DEFINE PATH ( NAME(PATH935) -
              PATHENTRY(AIX935) )
DEFINE AIX ( NAME(AIX936) RELATE(CL936) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(23 10) RECSZ(20 28) )
BLDINDEX INFILE (D1936) -
         OUTFILE (D2936)
DEFINE PATH ( NAME(PATH9336 -
              PATHENTRY(AIX936) )
DEFINE AIX ( NAME(AIX937) RELATE(CL937) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(44 10) RECSZ(28 48) )
BLDINDEX INFILE (D1937) -
         OUTFILE (D2937)
DEFINE PATH ( NAME(PATH937) -
              PATHENTRY(AIX937) )
DEFINE AIX ( NAME(AIX938) RELATE(CL938) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(40 10) RECSZ(28 48) )
BLDINDEX INFILE (D1938) -
         OUTFILE (D2938)
DEFINE PATH ( NAME(PATH938) -
              PATHENTRY(AIX938) )
DEFINE AIX ( NAME(AIX939) RELATE(CL939) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(41 10) RECSZ(     ) )
BLDINDEX INFILE (D1939) -
         OUTFILE (D2939)
DEFINE PATH ( NAME(PATH939) -
              PATHENTRY(AIX939) )
DEFINE AIX ( NAME(AIX940) RELATE(CL940) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(29 10) RECSZ(20 28) )
BLDINDEX INFILE (D1940) -
         OUTFILE (D2940)
DEFINE PATH ( NAME(PATH940) -
              PATHENTRY(AIX940) )
DEFINE AIX ( NAME(AIX941) RELATE(CL941) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(24 10) RECSZ(20 28) )
BLDINDEX INFILE (D1941) -
         OUTFILE (D2941)
```

```
DEFINE PATH ( NAME(PATH941) -
              PATHENTRY(AIX941) )
DEFINE AIX ( NAME(AIX942E) RELATE(CL942E) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(30 10) RECSZ(12 12) )
BLDINDEX INFILE (D1942E) -
         OUTFILE (D2942E)
DEFINE PATH ( NAME(PATH942E) -
              PATHENTRY(AIX942E) )
DEFINE AIX ( NAME(AIX942T) RELATE(CL942T) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(3 1) RECSZ(12 16) )
BLDINDEX INFILE (D1942T) -
         OUTFILE (D2942T)
DEFINE PATH ( NAME(PATH942T) -
              PATHENTRY(AIX942T) )
DEFINE AIX ( NAME(AIX942O) RELATE(CL942O) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(12 10) RECSZ(12 16) )
BLDINDEX INFILE (D1942O) -
         OUTFILE (D2942O)
DEFINE PATH ( NAME(PATH942O) -
              PATHENTRY(AIX942O) )
DEFINE AIX ( NAME(AIX943) RELATE(CL943) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(6 3) RECSZ(20 28) )
BLDINDEX INFILE (D1943) -
         OUTFILE (D2943)
DEFINE PATH ( NAME(PATH943) -
              PATHENTRY(AIX943) )
DEFINE AIX ( NAME(AIX944) RELATE(CL944) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(54 10) RECSZ(20 28) )
BLDINDEX INFILE (D1944) -
         OUTFILE (D2944)
DEFINE PATH ( NAME(PATH944) -
              PATHENTRY(AIX944) )
DEFINE AIX ( NAME(AIX945d) RELATE(CL945d) -
             VOL(VSAM01) KEYS(3 6) NONUNIQUEKEY -
             RECORDS(2 1) RECSZ(20 28) )
BLDINDEX INFILE (D1945d) -
         OUTFILE (D2945d)
DEFINE PATH ( NAME(PATH945d) -
              PATHENTRY(AIX945d) )
DEFINE AIX ( NAME(AIXCROP) RELATE(CLCROP) -
             VOL(VSAM01) KEYS(3 3) NONUNIQUEKEY -
             RECORDS(50 10) RECSZ(248 650) )
BLDINDEX INFILE (D1CROP) -
         OUTFILE (D2CROP)
DEFINE PATH ( NAME(PATHCROP) -
              PATHENTRY(AIXCROP) )
DEFINE AIX ( NAME(AIXDRAF) RELATE(CLDRAF) -
             VOL(VSAM01) KEYS(3 3) NONUNIQUEKEY -
             RECORDS(50 10) RECSZ(248 450) )
```

```
BLDINDEX INFILE (D1DRAF) -
         OUTFILE (D2DRAF)
DEFINE PATH ( NAME(PATHDRAF) -
              PATHENTRY(AIXDRAF) )
DEFINE AIX ( NAME(AIXPROD) RELATE(CLPROD) -
             VOL(VSAM01) KEYS(3 3) NONUNIQUEKEY -
             RECORDS(50 10) RECSZ(248 450) )
BLDINDEX INFILE (D1PROD) -
         OUTFILE (D2PROD)
DEFINE PATH ( NAME(PATHPROD) -
              PATHENTRY(AIXPROD) )
```

For the above definitions, the commands are contained in a CMS file with filetype EXEC.

```
DLBL D1911A F DSN CL911A (VSAM
DLBL D2911A F DSN AIX911A (VSAM
DLBL D1911B F DSN CL911B (VSAM
DLBL D2911B F DSN AIX911B (VSAM
DLBL D1912 F DSN CL912 (VSAM
DLBL D2912 F DSN AIX912 (VSAM
DLBL D1913 F DSN CL913 (VSAM
DLBL D2913 F DSN AIX913 (VSAM
DLBL D1915 F DSN CL915 (VSAM
DLBL D2915 F DSN AIX915 (VSAM
DLBL D1916 F DSN CL916 (VSAM
DLBL D2916 F DSN AIX916 (VSAM
DLBL D1917 F DSN CL917 (VSAM
DLBL D2917 F DSN AIX917 (VSAM
DLBL D1918 F DSN CL918 (VSAM
DLBL D2918 F DSN AIX918 (VSAM
DLBL D1919 F DSN CL919 (VSAM
DLBL D2919 F DSN AIX919 (VSAM
DLBL D1920 F DSN CL920 (VSAM
DLBL D2920 F DSN AIX920 (VSAM
DLBL D1930 F DSN CL930 (VSAM
DLBL D2930 F DSN AIX930 (VSAM
DLBL D1931 F DSN CL931 (VSAM
DLBL D2931 F DSN AIX931 (VSAM
DLBL D1932 F DSN CL932 (VSAM
DLBL D2932 F DSN AIX932 (VSAM
DLBL D1933 F DSN CL933 (VSAM
DLBL D2933 F DSN AIX933 (VSAM
DLBL D1934L F DSN CL934L (VSAM
DLBL D2934L F DSN AIX934L (VSAM
DLBL D1934V F DSN CL934V (VSAM
DLBL D2934V F DSN AIX934V (VSAM
DLBL D1935 F DSN CL935 (VSAM
DLBL D2935 F DSN AIX935 (VSAM
DLBL D1936 F DSN CL936 (VSAM
DLBL D2936 F DSN AIX936 (VSAM
DLBL D1937 F DSN CL937 (VSAM
DLBL D2937 F DSN AIX937 (VSAM
```

```
DLBL D1938 F DSN CL938 (VSAM
DLBL D2938 F DSN AIX938 (VSAM
DLBL D1939 F DSN CL939 (VSAM
DLBL D2939 F DSN AIX939 (VSAM
DLBL D1940 F DSN CL940 (VSAM
DLBL D2940 F DSN AIX940 (VSAM
DLBL D1941 F DSN CL941 (VSAM
DLBL D2941 F DSN AIX941 (VSAM
DLBL D1942E F DSN CL942E(VSAM
DLBL D2942E F DSN AIX942E (VSAM
DLBL D1942T F DSN CL942T (VSAM
DLBL D2942T F DSN AIX942T (VSAM
DLBL D19420 F DSN CL9420 (VSAM
DLBL D29420 F DSN AIX9420 (VSAM
DLBL D1CROP F DSN CLCROP (VSAM
DLBL D2CROP F DSN AIXCROP (VSAM
DLBL D1DRAF F DSN CLDRAF (VSAM
DLBL D2DRAF F DSN AIXDRAF (VSAM
DLBL D1PROD F DSN CLPROD (VSAM
DLBL D2PROD F DSN AIXPROD (VSAM
AMSERV DEFAIX
```

APPENDIX F

PROGRAM LISTING

```
MAIN: PROC OPTIONS (MAIN);                                          RET00010
      DCL OUTFILE FILE RECORD ENV(RECSIZE(80));                     RET00020
      DCL VARF  FILE RECORD ENV (VSAM);                             RET00030
      DCL BYF   FILE RECORD ENV (VSAM);                             RET00040
      DCL CONDF FILE RECORD ENV (VSAM);                             RET00050
      DCL (VARF_STR,BYF_STR,CONDF_STR) CHAR(150) VAR;               RET00060
      DCL SWBW  PIC'9';                                             RET00070
      DCL 1 VTABLE(300),                                            RET00080
            2 NAMEV        CHAR(30),                                RET00090
            2 FNV          CHAR(4),                                 RET00100
            2 OFFSETV      PIC'999',                                RET00110
            2 LENGTHV      PIC'9';                                  RET00120
      DCL 1 CTABLE(300),                                            RET00130
            2 NAMEC        CHAR(30),                                RET00140
            2 FNC          CHAR(4),                                 RET00150
            2 OFFSETC      PIC'999',                                RET00160
            2 LENGTHC      PIC'9',                                  RET00170
            2 CODESTART    PIC'999',                                RET00180
            2 CODEEND      PIC'999';                                RET00190
      DCL 1 CODES(1000),                                            RET00200
            2 CODENAME     CHAR(30),                                RET00210
            2 CODENO       PIC'999';                                RET00220
      DCL 1 VAR_ATTR(20),                                           RET00230
            2 VAR_FN       CHAR(4),                                 RET00240
            2 VAR_OS       PIC'999',                                RET00250
            2 VAR_LEN      PIC'9',                                  RET00260
            2 FUNCTION     PIC '9';                                 RET00270
      DCL 1 BY_ATTR(20),                                            RET00280
            2 BY_NAME      CHAR(30);                                RET00290
      DCL BY_FN        CHAR(4);                                     RET00300
      DCL BY_OS        PIC'999';                                    RET00310
      DCL BY_LEN       PIC'9';                                      RET00320
      DCL 1 COND_ATTR(20),                                          RET00330
            2 COND_FN      CHAR(4),                                 RET00340
            2 COND_OS      PIC'999',                                RET00350
            2 COND_LEN     PIC'9',                                  RET00360
            2 COND         PIC '9',                                 RET00370
            2 VALUE        PIC '(6)9';                              RET00380
      DCL 1 SPVAR(50),                                              RET00390
            2 NAME CHAR(30) INIT((50)(30)' '),                      RET00400
            2 FORM(30) CHAR(4) INIT((50)(30)'    '),                 RET00410
            2 OS(30) PIC'999',                                      RET00420
            2 LEN(30) PIC'9';                                       RET00430
      DCL OUT_STR      CHAR(80);                                    RET00440
      DCL VFNAME(20) CHAR(4), VFNPT FIXED INIT(0);                  RET00450
      DCL HH_CODE      PIC'999';                                    RET00460
      DCL COND_VAL     PIC'999';                                    RET00470
      DCL ACCEPT       PIC'9', (VAPT,BAPT,CAPT) FIXED;              RET00480
      DCL IPT          FIXED STATIC INIT(81);                       RET00490
      DCL LINE(80)     CHAR(1)       INIT((80)' ');                 RET00500
      DCL HHC_TAB(320)PIC '999', HHC_NO FIXED;                      RET00510
                                                                    RET00520
   /* SET UP TABLES */                                              RET00530
      CALL SETTAB;                                                  RET00540
   /* RECOGNIZE QUERY & FILL VAR_ATTR, BY_ATTR, COND_ATTR, COND_VAL */ RET00550
```

BEST AVAILABLE COPY

```
      CALL RECOG;                                                       RET00560
   /* COLLECT UNIQUE FILE NAMES INVOLVED IN THE QUERY */                RET00570
      CALL GETFN;                                                       RET00580
   /* RETRIEVE VARIABLES */                                             RET00590
      CALL RETRVE;                                                      RET00600
                                                                        RET00610
                                                                        RET00620
   /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */ RET00630
   /* * * * * *  PROCEDURE TO RECOGNIZE THE QUERY * * * * * * * * * */ RET00640
   /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */ RET00650
RECOG: PROC;                                                            RET00660
       DCL WORD CHAR(30) VAR, N30 CHAR(30);                             RET00670
       VAPT = 0; BAPT = 0; CAPT = 0;                                    RET00680
       SWBW = 0;                                                        RET00690
       WORD = NEXWRD(IDUM); /* RECOGNIZE GET */                         RET00700
       IF WORD ¬= 'GET' THEN                                            RET00710
          DO;                                                           RET00720
             PUT EDIT ('ERROR: STATEMENT NOT RECOGNIZED')(A);           RET00730
             STOP;                                                      RET00740
          END;                                                          RET00750
       WORD = NEXWRD(IDUM); /* RECOGNIZE FUNCTION OR VAR */             RET00760
RECV:  VAPT = VAPT + 1;                                                 RET00770
       FUNCTION(VAPT) = 0;                                              RET00780
       SELECT;                                                          RET00790
          WHEN (WORD='COUNT')    FUNCTION(VAPT) = 1;                   RET00800
          WHEN (WORD='SUM')      FUNCTION(VAPT)= 2;                    RET00810
          WHEN (WORD='AVERAGE') FUNCTION(VAPT) = 3;                    RET00820
          OTHERWISE GO TO NOFUNC;                                       RET00830
       END;                                                             RET00840
       /* RECOGNIZE THE VARIABLE */                                     RET00850
       WORD = NEXWRD(IDUM);                                             RET00860
NOFUNC:N30 = ' ';                                                       RET00870
       N30 = WORD;                                                      RET00880
       /* SEARCH VARIABLE TABLE */                                      RET00890
       CALL VTSR(N30,VAR_FN(VAPT),VAR_OS(VAPT),VAR_LEN(VAPT));          RET00900
       IF VAR_OS(VAPT) = 0 THEN                                         RET00910
          DO;                                                           RET00920
             PUT EDIT('ERROR: VARIABLE NO. ',VAPT,'NOT RECOGNIZED')     RET00930
                     (A,F(2,0),A);                                      RET00940
             STOP;                                                      RET00950
          END;                                                          RET00960
       WORD = NEXWRD(IDUM);                                             RET00970
       SELECT;                                                          RET00980
          WHEN (WORD = '.')      RETURN;                                RET00990
          WHEN (WORD = 'BY')     DO;                                    RET01000
                                    SWBW = 1;                           RET01010
                                    CALL RECBY;                         RET01020
                                    RETURN;                             RET01030
                                 END;                                   RET01040
          WHEN (WORD = 'WHERE') DO;                                     RET01050
                                    CALL RECWH;                         RET01060
                                    RETURN;                             RET01070
                                 END;                                   RET01080
          OTHERWISE GO TO RECV;                                         RET01090
       END;                                                             RET01100
```

```
END RECOG;                                                              RET01110
                                                                        RET01120
                                                                        RET01130
/* ------------------------------------------------------------------ */RET01140
/* ----------- PROCEDURE TO RECOGNIZE THE BY CLAUSE ---------------- */RET01150
/* ------------------------------------------------------------------ */RET01160
RECBY: PROC;                                                            RET01170
       DCL WORD CHAR(30) VAR, N30 CHAR(30);                             RET01180
       WORD = NEXWRD(IDUM);                                             RET01190
  RBY: N30 = ' ';                                                       RET01200
       N30 = WORD;                                                      RET01210
       BAPT = BAPT + 1;                                                 RET01220
       BY_NAME(BAPT) = N30;                                             RET01230
       WORD = NEXWRD(IDUM);                                             RET01240
       SELECT;                                                          RET01250
          WHEN (WORD = '.')      RETURN;                               RET01260
          WHEN (WORD = 'WHERE') DO;                                    RET01270
                                   CALL RECWH;                          RET01280
                                   RETURN;                              RET01290
                                END;                                    RET01300
          OTHERWISE GO TO RBY;                                          RET01310
       END;                                                             RET01320
END RECBY;                                                              RET01330
                                                                        RET01340
                                                                        RET01350
/* ------------------------------------------------------------------ */RET01360
/* ---------- PROCEDURE TO RECOGNIZE THE WHERE CLAUSE ------------- */RET01370
/* ------------------------------------------------------------------ */RET01380
RECWH: PROC;                                                            RET01390
       DCL WORD CHAR(30) VAR, N30 CHAR(30), CS PIC'999', CE PIC'999';   RET01400
       WORD = NEXWRD(IDUM);                                             RET01410
  RWH: N30 = ' ';                                                       RET01420
       N30 = WORD;                                                      RET01430
       CAPT = CAPT + 1;                                                 RET01440
       /* SEARCH COND TABLE */                                          RET01450
       CALL CTSR(N30,COND_FN(CAPT),COND_OS(CAPT),COND_LEN(CAPT),CS,CE);RET01460
       /* IF WORD IS IN CONDITION TABLE */                              RET01470
       IF COND_OS(CAPT) ¬= 0 THEN                                       RET01480
          DO;                                                           RET01490
             WORD = NEXWRD(IDUM);  /* RECOGNIZE '=' */                  RET01500
             IF WORD ¬= '=' THEN                                        RET01510
                DO;                                                     RET01520
                   PUT EDIT('ERROR: ',WORD,                             RET01530
                            ' NOT RECOGNIZED FOR WHERE CONDITION') (A); RET01540
                   STOP;                                                RET01550
                END;                                                    RET01560
             COND(CAPT) = 3;                                            RET01570
             WORD = NEXWRD(IDUM);  /* GET CONDITION VALUE */            RET01580
             N30 = ' ';                                                 RET01590
             N30 = WORD;                                                RET01600
             CALL CLSR(N30,VALUE(CAPT),CS,CE);                          RET01610
          END;                                                          RET01620
       ELSE                                                             RET01630
          DO;  /* WORD IS NOT IN CONDITION TABLE */                     RET01640
               /* SEARCH VARIABLE TABLE */                              RET01650
```

```
            CALL VTSR(N30,COND_FN(CAPT),COND_OS(CAPT),COND_LEN(CAPT));RET01660
            IF COND_OS(CAPT) = 0 THEN                                 RET01670
               DO;                                                    RET01680
                  PUT EDIT('ERROR: CONDITION NAME ',WORD,             RET01690
                           ' IS UNDEFINED')(A);                       RET01700
                  STOP;                                               RET01710
               END;                                                   RET01720
              /* WORD IS IN VARIABLE TABLE */                         RET01730
              /* RECOGNIZE CONDITION */                               RET01740
            WORD = NEXWRD(IDUM);                                      RET01750
            SELECT;                                                   RET01760
               WHEN (WORD = '<')  COND(CAPT) = 1;                     RET01770
               WHEN (WORD = '<=') COND(CAPT) = 2;                     RET01780
               WHEN (WORD = '=')  COND(CAPT) = 3;                     RET01790
               WHEN (WORD = '>=') COND(CAPT) = 4;                     RET01800
               WHEN (WORD = '>')  COND(CAPT) = 5;                     RET01810
               WHEN (WORD = '¬=') COND(CAPT) = 6;                     RET01820
               OTHERWISE DO;                                          RET01830
                            PUT EDIT('ERROR: CONDITION ',WORD,        RET01840
                                     ' NOT RECOGNIZED')(A);           RET01850
                            STOP;                                     RET01860
                         END;                                         RET01870
            END;                                                      RET01880
              /* RECOGNIZE CONDITION VALUE */                         RET01890
            WORD = NEXWRD(IDUM);                                      RET01900
            N30 = WORD;                                               RET01910
            GET STRING(N30) LIST(VALUE(CAPT));                        RET01920
         END;  /* END FOR ELSE STATEMENT */                           RET01930
      WORD = NEXWRD(IDUM);                                            RET01940
      SELECT;                                                         RET01950
         WHEN (WORD = '.')  RETURN;                                   RET01960
         WHEN (WORD = 'BY') DO;                                       RET01970
                               CALL RECBY;                            RET01980
                               RETURN;                                RET01990
                            END;                                      RET02000
         OTHERWISE GO TO RWH;                                         RET02010
      END;                                                            RET02020
END RECWH;                                                            RET02030
                                                                      RET02040
                                                                      RET02050
/* ------------------------------------------------------------------ */RET02060
/* ------- PROCEDURE TO GET NEXT WORD FROM THE INPUT BUFFER -------- */RET02070
/* ------------------------------------------------------------------ */RET02080
NEXWRD:PROC(IDUM) RETURNS (CHAR(30)VARYING);                          RET02090
      DCL WORD CHAR(30) VARYING, TEMP CHAR(30);                       RET02100
      WORD = '';                                                      RET02110
      IF IPT > 80 THEN DO;                                            RET02120
                          GET EDIT (LINE) ((80)A(1));                 RET02130
                          IPT = 1;                                    RET02140
                       END;                                           RET02150
      DO WHILE (LINE(IPT) = ' ');                                     RET02160
         IPT = IPT + 1;                                               RET02170
         IF IPT > 80 THEN DO;                                         RET02180
                            GET EDIT (LINE) ((80)A(1));               RET02190
                            IPT = 1;                                  RET02200
```

```
                              END;
       END;
       J = 0;
       DO WHILE (LINE(IPT) ¬= ' ');
          J = J + 1;
          SUBSTR (TEMP,J,1) = LINE(IPT);
          IPT = IPT + 1;
       END;
       GET STRING (TEMP) EDIT (WORD)(A(J));
       RETURN (WORD);
END NEXWRD;


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* * * * * * * PROCEDURE TO LOAD TABLES  * * * * * * * * * * * * * * * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
SETTAB:PROC;
       DCL VINDEX FILE RECORD ENV (RECSIZE(38));
       DCL CINDEX FILE RECORD ENV (RECSIZE(44));
       DCL CLIST  FILE RECORD ENV (RECSIZE(33));
       DCL SVINDEX FILE STREAM ENV (RECSIZE(150));
       DCL HHLIST FILE RECORD ENV(RECSIZE (3));
       DCL TNAME  CHAR(30);
       OPEN FILE (VINDEX) INPUT;
       OPEN FILE (CINDEX) INPUT;
       OPEN FILE (CLIST) INPUT;
       OPEN FILE (SVINDEX) INPUT;
       ON ENDFILE (VINDEX) GO TO READ2;
       ON ENDFILE (CINDEX) GO TO READ3;
       ON ENDFILE (CLIST) GO TO READ4;
       ON ENDFILE (HHLIST) GO TO READ5;
       ON ENDFILE (SVINDEX) GO TO RET;
READ1: I = 1;
       DO WHILE (6=6);
          READ FILE (VINDEX) INTO (VTABLE(I));
          I = I + 1;
       END;
READ2: I = 1;
       DO WHILE (6=6);
          READ FILE (CINDEX) INTO (CTABLE(I));
          I = I + 1;
       END;
READ3: I = 1;
       DO WHILE (6=6);
          READ FILE (CLIST) INTO (CODES(I));
          I = I + 1;
       END;
READ4: I = 1; HHC_NO = 0;
       DO WHILE (6=6);
          READ FILE(HHLIST) INTO (HHC_TAB(I));
          I = I + 1;
          HHC_NO = HHC_NO + 1;
       END;
READ5: DO I = 1 TO 50;
          GET FILE(SVINDEX)SKIP EDIT (TNAME) (A(30));
```

```
          IF TNAME = ' ' THEN RETURN;                                   RET02760
          SPVAR(I).NAME = TNAME;                                        RET02770
          /* GET FILENAMES, OFFSETS, AND LENGTHS */                     RET02780
          GET FILE(SVINDEX) SKIP EDIT                                   RET02790
                  ((SPVAR(I).FORM(J) DO J = 1 TO 30))((30)A(4));        RET02800
          GET FILE(SVINDEX) SKIP EDIT                                   RET02810
                  ((SPVAR(I).OS(J) DO J = 1 TO 30))((30)P'ZZZ');        RET02820
          GET FILE(SVINDEX) SKIP EDIT                                   RET02830
                  ((SPVAR(I).LEN(J) DO J = 1 TO 30))((30) (X(1),P'Z'));RET02840
       END;                                                             RET02850
RET: RETURN;                                                            RET02860
END SETTAB;                                                             RET02870
                                                                        RET02880
                                                                        RET02890
/* ---------------------------------------------------------------- */RET02900
/*  PROCEDURE TO SEARCH THE SPECIAL VAR TABLE  FOR OFFSET & LENGTH   */RET02910
/* ---------------------------------------------------------------- */RET02920
SPVSR: PROC (FN,SVN,OST,LNT);                                           RET02930
       DCL FN CHAR(4), SVN CHAR(30), OST PIC '999', LNT PIC'9';         RET02940
       DCL (I,J) FIXED;                                                 RET02950
       OST = 0; /* OST= 0 MEANS VAR NOT FOUND */                        RET02960
                /* OST=-1 MEANS VAR NOT IN THIS FILE */                 RET02970
       DO I = 1 TO 50;                                                  RET02980
          IF SPVAR(I).NAME = ' ' THEN RETURN;                           RET02990
          IF SPVAR(I).NAME = SVN THEN                                   RET03000
             DO J = 1 TO 30;                                            RET03010
               IF SPVAR(I).FORM(J) = FN THEN                            RET03020
                 DO; OST = SPVAR(I).OS(J);                              RET03030
                     LNT = SPVAR(I).LEN(J);                             RET03040
                     RETURN;                                            RET03050
                 END;                                                   RET03060
               OST = -1;                                                RET03070
             END;                                                       RET03080
       END;                                                             RET03090
END SPVSR;                                                              RET03100
                                                                        RET03110
                                                                        RET03120
/* ---------------------------------------------------------------- */RET03130
/* PROCEDURE TO SEARCH ORDINARY VARIABLE TABLE FOR OFFSET AND LENGTH */RET03140
/* ---------------------------------------------------------------- */RET03150
VTSR: PROC(NAME,FN,OST,LEN);                                            RET03160
      DCL NAME CHAR(30), FN CHAR(4), OST PIC'999',LEN PIC'9';           RET03170
      OST = 0;  /* OST = 0 MEANS NOT FOUND */                           RET03180
      DO I = 1 TO 300;                                                  RET03190
         IF VTABLE(I).NAMEV = ' ' THEN RETURN;                          RET03200
         IF VTABLE(I).NAMEV = NAME THEN                                 RET03210
            DO;                                                         RET03220
               FN = VTABLE(I).FNV;                                      RET03230
               OST = VTABLE(I).OFFSETV;                                 RET03240
               LEN = VTABLE(I).LENGTHV;                                 RET03250
               RETURN;                                                  RET03260
            END;                                                        RET03270
      END;                                                              RET03280
END VTSR;                                                               RET03290
                                                                        RET03300
```

```
                                                                      RET03310
/* ------------------------------------------------------------------ */RET03320
/* ----- PROCEDURE TO SEARCH CONDITION TABLE FOR OFFSET LENGTH ----- */RET03330
/* ----- STARTING AND ENDING POSITIONS OF CODES IN CODE TABLE  ----- */RET03340
/* ------------------------------------------------------------------ */RET03350
CTSR: PROC(NAME,FN,OST,LEN,CS,CE);                                    RET03360
      DCL NAME CHAR(30), FN CHAR(4), OST PIC'999', LEN PIC'9',        RET03370
                                     CS PIC'999', CE PIC'999';        RET03380
      OST = 0;                                                        RET03390
      DO I = 1 TO 300;                                                RET03400
         IF CTABLE(I).NAMEC = ' ' THEN RETURN;                        RET03410
         IF CTABLE(I).NAMEC = NAME THEN                               RET03420
            DO;                                                       RET03430
               FN = CTABLE(I).FNC;                                    RET03440
               OST = CTABLE(I).OFFSETC;                               RET03450
               LEN = CTABLE(I).LENGTHC;                               RET03460
               CS  = CTABLE(I).CODESTART;                             RET03470
               CE  = CTABLE(I).CODEEND;                               RET03480
               RETURN;                                                RET03490
            END;                                                      RET03500
      END;                                                            RET03510
END CTSR;                                                             RET03520
                                                                      RET03530
                                                                      RET03540
/* ------------------------------------------------------------------ */RET03550
/* -------PROCEDURE TO SEARCH THE CODE LIST FOR CODE VALUE --------- */RET03560
/* ------------------------------------------------------------------ */RET03570
CLSR: PROC(NAME,VALUE,ST,EN);                                         RET03580
      DCL NAME CHAR(30), VALUE PIC '(6)9', EN PIC '999', ST PIC'999';  RET03590
      DO I = ST TO EN;                                                RET03600
         IF CODES(I).CODENAME = NAME THEN                             RET03610
            DO;                                                       RET03620
               VALUE = CODES(I).CODENO;                               RET03630
               RETURN;                                                RET03640
            END;                                                      RET03650
      END;                                                            RET03660
      PUT SKIP EDIT('ERROR: CONDITION VALUE ',NAME,' IS NOT VALID')(A);RET03670
END CLSR;                                                             RET03680
                                                                      RET03690
                                                                      RET03700
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */RET03710
/*  PROCEDURE TO STORE FILENAMES OF THE VARIABLES INVOLVED IN QUERY  */RET03720
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */RET03730
GETFN: PROC;                                                          RET03740
       DCL FN CHAR(4) INIT('   '), I FIXED;                           RET03750
       DO I = 1 TO VAPT;                                              RET03760
          IF VAR_FN(I) ¬= FN THEN DO;                                 RET03770
                                     FN = VAR_FN(I);                  RET03780
                                     VFNPT = VFNPT + 1;               RET03790
                                     VFNAME(VFNPT) = FN;              RET03800
                                  END;                                RET03810
       END;                                                           RET03620
END GETFN;                                                            RET03830
                                                                      RET03840
                                                                      RET03850
```

```
/* # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # */  RET03860
/* # # # #  PROCEDURE TO RETRIEVE ALL VARIABLES IN QUERY # # # # # # */  RET03870
/* # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # */  RET03880
RETRVE: PROC;                                                            RET03890
  DCL 1 BUF(100), /* ONE FOR EACH COMBINATION OF SPECIAL VAR */          RET03900
       2 HH_CODE PIC'999' INIT((100)0),                                   RET03910
       2 BY_VAL PIC'(10)9' INIT((100)0),                                  RET03920
       2 VARVAL(10) PIC'(6)9' INIT((1000)0),                              RET03930
       2 VARCNT(10) PIC'999' INIT((1000)0);                               RET03940
  DCL FN CHAR(4), BUFPT FIXED INIT(0), OUTBUF CHAR(80) INIT((80)' '),    RET03950
      HHC PIC'999';                                                      RET03960
  DO I = 1 TO HHC_NO;        /* FOR EACH HH_CODE */                      RET03970
    HHC = HHC_TAB(I);                                                    RET03980
    CALL INITBUF;            /* INITIALIZE BUF */                        RET03990
    DO J = 1 TO VFNPT;       /* FOR EACH FILE */                         RET04000
      FN = VFNAME(J);                                                    RET04010
      CALL RRHH(HHC,FN);     /* RETRIEVE RECORDS WITH GIVEN HH CODES */  RET04020
    END;                                                                 RET04030
    /* THE RETRIEVED RECORDS ARE IN BUF */                               RET04040
    DO K = 1 TO BUFPT;                    /* FOR EACH RECORD RETRIEVED */ RET04050
      SUBSTR(OUTBUF,1,3) = HH_CODE(K); /* COPY HH CODE */                RET04060
      SUBSTR(OUTBUF,4,10) = BY_VAL(K); /* COPY SPECIAL VARS */           RET04070
      M = 14;                                                            RET04080
      DO L = 1 TO 10;                     /* COPY ALL 10 VARIABLES */    RET04090
        SUBSTR(OUTBUF,M,6) = BUF(K).VARVAL(L);                           RET04100
        M = M + 6;                                                       RET04110
      END;                                                               RET04120
      WRITE FILE (OUTFILE) FROM (OUTBUF);                                RET04130
    END;                                                                 RET04140
  END;                                                                   RET04150
  STOP;                                                                  RET04160
                                                                         RET04170
                                                                         RET04180
/* ------------------------------------------------------------------ */RET04190
/* ----------------  PROCEDURE TO INITIALIZE BUF  ---------------- */RET04200
/* ------------------------------------------------------------------ */RET04210
INITBUF: PROC;                                                           RET04220
  DCL (I,J) FIXED;                                                       RET04230
  IF BUFPT = 0 THEN RETURN;                                              RET04240
  DO I = 1 TO BUFPT;                                                     RET04250
    HH_CODE(I) = 0;                                                      RET04260
    BY_VAL(I) = 0;                                                       RET04270
    DO J = 1 TO 10;                                                      RET04280
      BUF(I).VARVAL(J) = 0;                                              RET04290
      BUF(I).VARCNT(J) = 0;                                              RET04300
    END;                                                                 RET04310
  END;                                                                   RET04320
  BUFPT = 0;                                                             RET04330
  RETURN;                                                                RET04340
END INITBUF;                                                             RET04350
                                                                         RET04360
                                                                         RET04370
/* ------------------------------------------------------------------ */RET04380
/* ----------  PROCEDURE TO RETRIEVE ORDINARY VARIABLES  ---------- */RET04390
/* ---------- FROM GIVEN FILE 'FN' FOR GIVEN HHCODE 'HHC' ---------- */RET04400
```

```
/* ------------------------------------------------------------------ */RET04410
RRHH: PROC(HHC,FN);                                                     RET04420
  DCL HHC PIC'999', FN CHAR(4), BVAL PIC'(10)9', ACCEPT PIC'9';         RET04430
  DCL (I,J)  FIXED;                                                     RET04440
  ON KEY(VARF) BEGIN;                                                   RET04450
                 CLOSE FILE (VARF);                                     RET04460
                 GO TO R;                                               RET04470
               END;                                                     RET04480
  OPEN FILE(VARF) KEYED INPUT TITLE('PTH' || FN);                       RET04490
  READ FILE (VARF) INTO (VARF_STR) KEY(HHC);/* READ FIRST RECORD */     RET04500
  CALL GETOV;                                /* GET ORDINARY VARIABLE */RET04510
  DO WHILE(SAMEKEY(VARF));                   /* GET NEXT RECORD */      RET04520
    READ FILE(VARF) INTO (VARF_STR);                                    RET04530
    CALL GETOV;                                                         RET04540
  END;                                                                  RET04550
  CLOSE FILE (VARF);                                                    RET04560
  /* SEARCH FOR FUNCTION = AVERAGE */                                   RET04570
  DO I = 1 TO VAPT;                                                     RET04580
    IF FUNCTION(I) = 3 THEN                                             RET04590
      DO J = 1 TO BUFPT;                                                RET04600
        IF BUF(J).VARCNT(I) ¬= 0                                        RET04610
        THEN   BUF(J).VARVAL(I) = BUF(J).VARVAL(I)/BUF(J).VARCNT(I);    RET04620
      END;                                                              RET04630
  END;                                                                  RET04640
R:RETURN;                                                               RET04650
                                                                        RET04660
                                                                        RET04670
/* ------------------------------------------------------------------ */ RET04680
/* -----------  PROCEDURE TO RETRIEVE ORDINARY VAR  --------------- */ RET04690
/* ------------------------------------------------------------------ */ RET04700
GETOV: PROC;                                                            RET04710
  DCL I FIXED;                                                          RET04720
  CALL RTBV(VARF_STR,BVAL,FN);/* RETRIEVE SPECIAL VARIABLES INTO BVAL*/RET04730
  CALL TEST(ACCEPT,BVAL);     /* TEST CONDITION */                      RET04740
  IF ACCEPT ¬= 1 THEN RETURN; /* IF CONDITION NOT TRUE RETURN */        RET04750
  IF BUFPT = 0                /* IF BUF IS EMPTY THEN PUT THIS*/        RET04760
  THEN  DO;                   /* SPECIAL VAR IN THE FIRST ELEMENT */    RET04770
          BUFPT = 1;                                                    RET04780
          BUF(1).HH_CODE = HHC;                                         RET04790
          BUF(1).BY_VAL = BVAL;                                         RET04800
          /* RETRIEVE ORDINARY VARIABLES INTO THIS ELEMENT OF BUF */    RET04810
          CALL RTVR (BUFPT,FN);                                         RET04820
          RETURN;                                                       RET04830
        END;                                                            RET04840
  DO I = 1 TO BUFPT;            /* SEARCH FOR SPECIAL VARIABLE */       RET04850
    IF BUF(I).BY_VAL = BVAL & BUF(I).HH_CODE = HHC                      RET04860
    THEN DO;                                                            RET04870
           /* RETRIEVE ORDINARY VARIABLES INTO THIS ELEMENT OF BUF */   RET04880
           CALL RTVR (I,FN);                                            RET04890
           RETURN;                                                      RET04900
         END;                                                           RET04910
  END;                                                                  RET04920
  /* HERE SPECIAL VARIABLE NOT FOUND IN BUF, INSERT ONE */              RET04930
  BUFPT = BUFPT + 1;                                                    RET04940
  BUF(BUFPT).HH_CODE = HHC;                                             RET04950
```

```
  BUF(BUFPT).BY_VAL = BVAL;                                          RET04960
  CALL RTVR(BUFPT,FN);                                               RET04970
  RETURN;                                                            RET04980
END GETOV;                                                           RET04990
                                                                     RET05000
                                                                     RET05010
/* ---------------------------------------------------------------- */RET05020
/*    PROCEDURE TO TEST IF A GIVEN RECORD SATISFIES THE CONDITION   */RET05030
/* ---------------------------------------------------------------- */RET05040
TEST: PROC(ACCEPT,BVAL);                                             RET05050
  DCL ACCEPT PIC'9',(BVAL,BCVAL)PIC'(10)9',RDBUF CHAR(20)INIT((20)' '),RET05060
      VAL FIXED BIN(31), (I,OPEN) FIXED;                             RET05070
  CONDF_STR = VARF_STR;                                              RET05080
  ACCEPT = 1;                                                        RET05090
  OPEN = 0;                                                          RET05100
  IF CAPT = 0 THEN RETURN;                                           RET05110
  DO I = 1 TO CAPT;          /* FOR EACH CONDITION */                RET05120
    OPEN = 0;                                                        RET05130
    ON KEY(CONDF) BEGIN;                                             RET05140
                    CLOSE FILE (CONDF);                              RET05150
                    GO TO R;                                         RET05160
                  END;                                               RET05170
    IF COND_FN(I) ¬= FN      /* IF THE CODED VARIABLE COMES FROM */  RET05180
    THEN DO;                 /* A DIFFERENT FILE THEN OPEN THAT FILE */ RET05190
          OPEN FILE (CONDF) KEYED INPUT TITLE('PTH' || COND_FN(I));  RET05200
          READ FILE (CONDF) INTO (CONDF_STR) KEY (HHC);              RET05210
          OPEN = 1;                                                  RET05220
        END;                                                         RET05230
    IF BAPT = 0 | SWBW = 1                                           RET05240
    THEN DO;                 /* IF THERE IS NO BY-CLAUSE OR BY CLAUSE */RET05250
          CALL COMP;         /* PRECEEDS THE WHERE-CLAUSE THEN      */RET05260
          IF ACCEPT = 0 THEN DO;  /* COMPARE WITH HH CODE        */  RET05270
                               IF OPEN = 1 THEN CLOSE FILE(CONDF);   RET05280
                               RETURN;                               RET05290
                             END;                                    RET05300
        END;                                                         RET05310
    ELSE DO;                 /* IF THE BY-CLAUSE FOLLOWS THE WHERE */ RET05320
                             /* CLAUSE THEN GET SPECIAL VARIABLES  */ RET05330
          CALL RTBV(CONDF_STR,BCVAL,COND_FN(I));                     RET05340
          IF BVAL = BCVAL                                            RET05350
          THEN DO;           /* IF CURRENT RECORD HAS GIVEN SPECIAL */ RET05360
                CALL COMP;/* VARIABLE THEN COMPARE */                RET05370
                IF ACCEPT = 0 THEN                                   RET05380
                               DO; IF OPEN = 1 THEN CLOSE FILE(CONDF);RET05390
                                   RETURN;                           RET05400
                               END;                                  RET05410
              END;                                                   RET05420
          ELSE DO;           /* READ NEXT RECORD WITH THIS HH CODE */ RET05430
L1:           DO;                                                    RET05440
                DO WHILE(SAMEKEY(CONDF));                            RET05450
                  READ FILE (CONDF) INTO (CONDF_STR);                RET05460
                  CALL RTBV (CONDF_STR,BCVAL,COND_FN(I));            RET05470
                  IF BVAL = BCVAL                                    RET05480
                  THEN DO;                                           RET05490
                        CALL COMP;                                   RET05500
```

```
                          IF ACCEPT = 0                                  RET05510
                          THEN DO;                                       RET05520
                                 IF OPEN = 1 THEN CLOSE FILE (CONDF);    RET05530
                                 RETURN;                                 RET05540
                               END;                                      RET05550
                          LEAVE L1;                                      RET05560
                        END;                                             RET05570
                  END;                                                   RET05580
                  ACCEPT = 0;                                            RET05590
                  IF OPEN = 1 THEN CLOSE FILE(CONDF);                    RET05600
                  RETURN;                                                RET05610
                END L1;                                                  RET05620
                END;                                                     RET05630
        END;                                                             RET05640
  IF OPEN = 1 THEN CLOSE FILE(CONDF);                                    RET05650
  END;                                                                   RET05660
R:RETURN;                                                                RET05670
                                                                         RET05680
                                                                         RET05690
/* ---------------------------------------------------------------- */RET05700
/*    PROCEDURE TO TEST IF CONDITION IN WHERE CLAUSE IS SATISFIED    */RET05710
/* ---------------------------------------------------------------- */RET05720
COMP: PROC;                                                              RET05730
  DCL LEN PIC'9';                                                        RET05740
  ACCEPT = 0;                                                            RET05750
  LEN = COND_LEN(I);    /* GET LENGTH OF THE CONDITION VARIABLE */       RET05760
  /* GET THE VALUE OF THE CONDITION VARIABLE */                          RET05770
  SUBSTR(RDBUF,1,LEN) = SUBSTR(CONDF_STR,COND_OS(I),LEN);                RET05780
  GET STRING (RDBUF) LIST (VAL);                                         RET05790
  SELECT (COND(I));                                                      RET05800
    WHEN(1) IF VAL >= VALUE(I) THEN RETURN;                              RET05810
    WHEN(2) IF VAL >  VALUE(I) THEN RETURN;                              RET05820
    WHEN(3) IF VAL ¬= VALUE(I) THEN RETURN;                              RET05830
    WHEN(4) IF VAL <  VALUE(I) THEN RETURN;                              RET05840
    WHEN(5) IF VAL <= VALUE(I) THEN RETURN;                              RET05850
    WHEN(6) IF VAL =  VALUE(I) THEN RETURN;                              RET05860
    OTHERWISE;                                                           RET05870
  END;                                                                   RET05880
    ACCEPT = 1;                                                          RET05890
    RETURN;                                                              RET05900
END COMP;                                                                RET05910
END TEST;                                                                RET05920
                                                                         RET05930
                                                                         RET05940
/* ---------------------------------------------------------------- */RET05950
/*             PROCEDURE TO ACCUMULATE VALUE INTO BUF               */RET05960
/*      AS PER FUNCTION SPECIFIED FOR SINGLE OCCURENCE OF RECORD    */RET05970
/* ---------------------------------------------------------------- */RET05980
RTVR: PROC(J,FN);                                                        RET05990
  DCL FN CHAR(4), (I,J) FIXED,                                           RET06000
      LNT PIC'9', OST PIC'999', FUN PIC'9', TEMP FIXED BIN (31),         RET06010
      RDBUF CHAR(20);                                                    RET06020
  DO I = 1 TO VAPT;       /* FOR EACH VARIABLE */                        RET06030
    /* MATCH FOR FILE NAME */                                            RET06040
    IF VAR_FN(I) = FN                                                    RET06050
```

```
    THEN DO;                                                          RET06060
          RDBUF = ' ';                                                RET06070
          LNT = VAR_LEN(I);                                           RET06080
          OST = VAR_OS(I);                                            RET06090
          FUN = FUNCTION(I);                                          RET06100
          /* GET THE VARIABLE VALUE INTO READ BUFFER */               RET06110
          SUBSTR(RDBUF,1,LNT) = SUBSTR(VARF_STR,OST,LNT);             RET06120
          /* READ VALUE WITH APPROPRIATE P FORMAT */                  RET06130
          GET STRING (RDBUF) LIST (TEMP);                             RET06140
          SELECT (FUN);                                               RET06150
            WHEN(0) BUF(J).VARVAL(I) = TEMP;                          RET06160
            WHEN(1) BUF(J).VARVAL(I) = BUF(J).VARVAL(I) + 1;          RET06170
            WHEN(2) BUF(J).VARVAL(I) = BUF(J).VARVAL(I) + TEMP;       RET06180
            WHEN(3) DO;                                               RET06190
                      BUF(J).VARVAL(I) = BUF(J).VARVAL(I) + TEMP;     RET06200
                      BUF(J).VARCNT(I)= BUF(J).VARCNT(I)+ 1;          RET06210
                    END;                                              RET06220
             OTHERWISE;                                               RET06230
          END;                                                        RET06240
        END;                                                          RET06250
  END;                                                                RET06260
  RETURN;                                                             RET06270
END RTVR;                                                             RET06280
                                                                      RET06290
                                                                      RET06300
/* ------------------------------------------------------------------ */RET06310
/* PROCEDURE TO RETRIEVE SPECIAL VARIABLE VALUES FROM A GIVEN RECORD */RET06320
/* ------------------------------------------------------------------ */RET06330
RTBV: PROC(REC,BVAL,FN);                                              RET06340
  DCL REC CHAR(150) VARYING, BVAL PIC'(10)9', FN CHAR(4),             RET06350
      OST PIC'999', LNT PIC'9', (I,J) FIXED,                          RET06360
      TEMP FIXED BIN (31), RDBUF CHAR(20) INIT((20)' ');              RET06370
      J = 1;                                                          RET06380
      BVAL = 0;                                                       RET06390
      IF BAPT = 0 THEN RETUEN;                                        RET06400
      DO I = 1 TO BAPT;                                               RET06410
        CALL SPVSR(FN,BY_NAME(I),OST,LNT);                            RET06420
        IF OST = 0 | OST = -1 THEN                                    RET06430
        DO; PUT SKIP EDIT('ERROR: VARIABLE NAME ',BY_NAME(I),         RET06440
               'USED IN THE BY CLAUSE IS NOT A SPECIAL VARIABLE') (A);RET06450
            STOP;                                                     RET06460
        END;                                                          RET06470
        SUBSTR(RDBUF,J,LNT) = SUBSTR(REC,OST,LNT);                    RET06480
        J = J + LNT;                                                  RET06490
      END;                                                            RET06500
      GET STRING (RDBUF) LIST (TEMP);                                 RET06510
      BVAL = TEMP;                                                    RET06520
      RETURN;                                                         RET06530
END RTBV;                                                             RET06540
END RRHH;                                                             RET06550
END RETRVE;                                                           RET06560
END MAIN;                                                             RET06570
```

```
&CONTROL OFF
DLBL CL911A F DSN CL911A (VSAM
DLBL CL911B F DSN CL911B (VSAM
DLBL CL912  F DSN CL912  (VSAM
DLBL CL913  F DSN CL913  (VSAM
DLBL CL915  F DSN CL915  (VSAM
DLBL CL916  F DSN CL916  (VSAM
DLBL CL917  F DSN CL917  (VSAM
DLBL CL918  F DSN CL918  (VSAM
DLBL CL919  F DSN CL919  (VSAM
DLBL CL920  F DSN CL920  (VSAM
DLBL CL930  F DSN CL930  (VSAM
DLBL CL931  F DSN CL931  (VSAM
DLBL CL932  F DSN CL932  (VSAM
DLBL CL933  F DSN CL933  (VSAM
DLBL CL934L F DSN CL934L (VSAM
DLBL CL934V F DSN CL934V (VSAM
DLBL CL935  F DSN CL935  (VSAM
DLBL CL936  F DSN CL936  (VSAM
DLBL CL937  F DSN CL937  (VSAM
DLBL CL938  F DSN CL938  (VSAM
DLBL CL939  F DSN CL939  (VSAM
DLBL CL940  F DSN CL940  (VSAM
DLBL CL941  F DSN CL941  (VSAM
DLBL CL942E F DSN CL942E (VSAM
DLBL CL942T F DSN CL942T (VSAM
DLBL CL942O F DSN CL942O (VSAM
DLBL CL943  F DSN CL943  (VSAM
DLBL CL944  F DSN CL944  (VSAM
DLBL CL945D F DSN CL945D (VSAM
DLBL CL945A F DSN CL945A (VSAM
DLBL PTH911A F DSN PATH911A (VSAM
DLBL PTH911B F DSN PATH911B (VSAM
DLBL PTH912  F DSN PATH912  (VSAM
DLBL PTH913  F DSN PATH913  (VSAM
DLBL PTH915  F DSN PATH915  (VSAM
DLBL PTH916  F DSN PATH916  (VSAM
DLBL PTH917  F DSN PATH917  (VSAM
DLBL PTH918  F DSN PATH918  (VSAM
DLBL PTH919  F DSN PATH919  (VSAM
DLBL PTH920  F DSN PATH920  (VSAM
DLBL PTH930  F DSN PATH930  (VSAM
DLBL PTH931  F DSN PATH931  (VSAM
DLBL PTH932  F DSN PATH932  (VSAM
DLBL PTH933  F DSN PATH933  (VSAM
DLBL PTH934L F DSN PATH934L (VSAM
DLBL PTH934V F DSN PATH934V (VSAM
DLBL PTH935  F DSN PATH935  (VSAM
DLBL PTH936  F DSN PATH936  (VSAM
DLBL PTH937  F DSN PATH937  (VSAM
DLBL PTH938  F DSN PATH938  (VSAM
DLBL PTH939  F DSN PATH939  (VSAM
DLBL PTH940  F DSN PATH940  (VSAM
DLBL PTH941  F DSN PATH941  (VSAM
DLBL PTH942E F DSN PATH942E (VSAM
```

```
DLBL PTH942T F DSN PATH942T (VSAM
DLBL PTH9420 F DSN PATH9420 (VSAM
DLBL PTH943  F DSN PATH943  (VSAM
DLBL PTH944  F DSN PATH944  (VSAM
DLBL PTH945D F DSN PATH945D (VSAM
DLBL PTH945A F DSN PATH945A (VSAM
FI VINDEX DISK VINDEX TABLE D(RECFM F LRECL 38
FI CINDEX DISK CINDEX TABLE D(RECFM F LRECL 44
FI CLIST  DISK CLIST  TABLE D(RECFM F LRECL 33
FI SYSIN  DISK QUERY  DATA  D(RECFM F LRECL 80
FI OUTFILE DISK RESULT DATA D(RECFM F LRECL 80
FI SVINDEX DISK SVINDEX TABLE D(RECFM F LRECL 150
FI HHLIST DISK HHCODES DATA D(RECFM F LRECL 3
```

APPENDIX G

ILLUSTRATIONS

HHCODE	PARCEL ID	A	B
100	01	10	100
100	02	15	213
101	02	13	415
103	01	19	104
103	02	20	160

HHCODE	C	D
100	21	40
101	20	49
102	19	50

Q1: GET A B BY PARCEL-ID WHERE C > 20 .

A1:

HHCODE	PARCEL-ID	A	B
100	01	10	100
100	02	15	213

In this format where BY preceeds WHERE the files where A and B are found must have PARCEL-ID in the record.

In this query the files are related by HH-CODE.

Q2: GET A B WHERE C > 20 BY PARCEL-ID .

A2: ERROR will occur because the file where C is found does not have PARCEL-ID which in this format where WHERE preceeds BY, both PARCEL-ID and HHCODE need to be used to relate the files.

92

HHCODE	PARCEL ID	F	G
100	01	101	200
101	01	98	167
101	02	180	174

HHCODE	C	D
100	21	40
101	20	49
102	19	50

Q3: GET A B WHERE F > 100 BY PARCEL-ID .

A3:

HHCODE	PARCEL-ID	A	B
100	01	10	100
101	02	13	415

Q4: GET A B BY PARCEL-ID WHERE F > 100 .

A4:

HHCODE	PARCEL-ID	A	B
100	01	10	100
100	02	15	213