ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA (SISA)

NETWORK PROGRAMMING: A CASE STUDY WITH LIVESTOCK INFORMATION MANAGEMENT SYSTEM (LIMS)

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SCIENCE

BY

HAMBISA BELINA

MAY 1997

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

NETWORK PROGRAMMING: A CASE STUDY WITH LIVESTOCK INFORMATION MANAGEMENT SYSTEM (LIMS)

By

Hambisa Belina Disassa

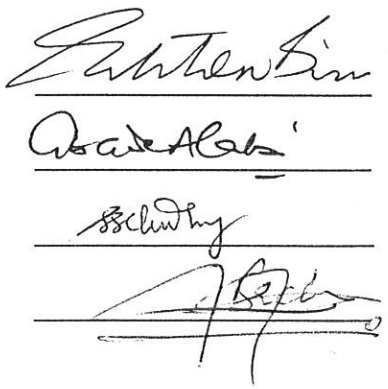
Name and Signature of Members of the Examining Board

Ato Getachew Birru, Chairman, Examining Board

Dr. G.A. Alabi, Advisor

Dr. G.G. Chowdhury, Internal Examiner

Dr. K. Bechkoum, External Examiner



ACKNOWLEDGMENT

To complete the study, I realize that I am in great debt to a number of people. First and foremost to my wife Lemane and our children Sollan and Dettii, who patiently have been waiting for the day when I can commit time to them.

I am also greatly indebted to a number of other people whose support and encouragement have meant more to me than they ever imagine. My special heart-felt thanks are due to my two advisors Dr. Gbade Alabi and Prof. Dr. Erich Bruns whose close guidance, invaluable suggestions and constructive comments have shaped this study in several ways.

I am thankful for the faculty and staff of the School of Information Studies for Africa (SISA) for their dedicated support and guidance during my stay in the School and while undertaking the study.

I am grateful to the German Academic Exchange Service (DAAD) and Institut fuer Tierzucht und Haustiergenetik - Georg-August-Universitaet Goettingen for their financial support for the study. I am also indebted to my employers at the International Livestock Research Institute (ILRI) for giving me leave of absence for the duration of the study and for allowing me to use the LIMS source code for this study.

My thanks are due to my friends and colleagues at ILRI whose encouragement, understanding and support were invaluable in many ways.

ABSTRACT

This study has made an attempt to address the major issues that should be considered when programming application programs that run in a network environment. The implementation of these issues has also been demonstrated through the enhancement of the Livestock Information Management System (LIMS), a software developed by the International Livestock Research Institute (ILRI) the former International Livestock Centre for Africa (ILCA).

During the course of this study, LIMS has been enhanced and the following facilities have been incorporated to the system: 1) the five modules of the LIMS software were integrated so that all the modules could be accessed from a single user interface menu; 2) LIMS was enhanced for simultaneous access by various users in a network environment; 3) password protection was added for options that require special security due to the effects of such processes on the data set; 4) a semaphore protection means was developed for memory files which are not supported by the locking provisions of the programming language used; 5) record and file locking were implemented for keeping the integrity of the data files; 6) a time duration checking was added to allow for several attempts that fail on the first trial; and 7) a sure write procedure has been implemented so that changes are made visible to all users of the shared data.

The study has also made recommendations on the need for rigorously testing of the current work, addition of on-line help facility, and system utilities for better use of the software.

TABLE OF CONTENTS

CHAPTER FOUR: IMPLEMENTING LIMS ENHANCEMENT

CHAPTER FIVE: CONCLUSIONS AND RECOMMENDATIONS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS USED IN THE THESIS

ACP	African, Caribbean, Pacific
API	Applications Program Interfaces
ASCII	American Standard Character for Information Interchange
DOS	Disk Operating System
IBM	International Business Machines Corporation
ID	Identification
ILCA	International Livestock Centre for Africa
ILRI	International Livestock Research Institute (the former ILCA)
INT	Interrupt
LAN	Local Area Network
LCD	Liquid Crystal Display
LIMS	Livestock Information Management System
MS-DOS	Microsoft Disk Operating System
RAM	Random Access Memory
SDF	System Data Format

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND AND JUSTIFICATION

"Whether to hunters and herdsmen of pre-industrial times, or to latter-day captains of industry and commerce, information has always been important. Today, however, something is happening to information, to its roles and status, its form and structure, that suggests developments of an entirely different order. Information has become the talisman, a symbol of political potency and economic prosperity. As a geopolitical phenomenon it carries implications for relations between nations, the future of institutions, for value systems and for ways of life" (Martin 1988). Economically, information has become a factor of such central importance that the developed countries are information economies already.

Information is a basic raw material needed in any sphere of life of the modern society. It is intelligence or knowledge that contributes to the social, economic, cultural and political well-being of any nation. There is no sector or economic activity that can effectively function without timely, accurate and reliable information. Decision makers and policy formulators need pertinent, accurate and timely information in order to make informed decisions. Every economic, social and political system will perform more efficiently if operators of such systems, namely decision makers, planners, managers, executives, researchers, etc. have access to timely, up-to-date, relevant and reliable information.

There are numerous ways of capturing, gathering, storing, processing, and communicating information so that decision makers make informed decisions. Due to the advances made in the area of information technology which enhanced the speed, reliability and accuracy of microcomputers, computer has become the dominant, effective and efficient way of handling, processing and disseminating information. Computers process data speedily and accurately and provide information when and where required at the correct level of detail.

To be useful, information must be structured and mobile. The value of information is directly tied to the entity (person or program) that accesses it. One can increase the value of information simply by moving it to a place where some entity can make use of it. Mobility makes information valuable. The structure and transfer of information are of key importance to successful organizations (Tanenbaum 1989; Rose 1990).

The value of information can further be enhanced through networking, which is the interconnection of large numbers of separate computers. Networks are the movers of information. According to James (1989), network is the single most important advance in the way computers are used since the introduction of personal computers. The data processing requirements of many businesses require the transfer of data from one location to another. James (1989) argues that "while networking offers the benefit of sharing expensive hardware, the greatest potential benefit comes from the sharing of data."

For a computer to capture, process, store and retrieve information, it must first be supplied with an appropriate algorithm. The algorithm is expressed in a form of a program with a

sequence of statements, each of which specifies certain operations the computer is to perform (Goldschlager 1982).

In a network environment, the computer program should specify how shared resources are accessed by multiple users in a synchronized, integrated, and consistent way, besides specifying the algorithm used to perform the required tasks.

Multi-user software integrate the functioning of different parts of an organization. Rather than sharing data files amongst one user at a time, a multi-user application allows many users to share and modify data stored in a file simultaneously. Once data is shared in a multi-user application, the method of sharing quickly becomes the functioning of the company. According to James (1989), this is often the greatest benefit that a network can bring to an organization.

This study is concerned with investigating network programming as applied to a software used to record and manage livestock information.

Livestock are an integral component of the agriculture of many developing nations and go well beyond serving as a major source of good quality food (Sansoucy et al 1995). They are the living bank for many farmers and have a critical role in the agricultural intensification process through the provision of draught power and manure for fertilizer and fuel.

According to Seré (1995) and Masiga (1995), livestock are maintained for various purposes. They produce food, provide security, enhance crop production, generate cash incomes for

rural and urban populations, provide fuel and transport, provide fibre, and produce value added goods which can have multiplier effects and create a need for services. Furthermore, livestock diversify production and income, provide year-round employment, and spread risk as well as form a major capital reserve of farming households.

The ability to meet the ever increasing demand for livestock products requires decisions which are based on relevant and accurate information. In an attempt to provide computer based livestock information which would assist in livestock related decision making, the International Livestock Centre for Africa (ILCA) in 1991/92 developed a computer system called Livestock Information Management System (LIMS). LIMS was designed to facilitate the recording and processing of livestock related data. The system is not specialized in any species or production system, but is applicable for all mammal species in different situations, such as research, extension and commercial production (ILCA 1992).

LIMS was designed to provide users with a computerized tool that allows them to manage livestock performance data sets. It was also meant to provide help in the decision making process of farm management. However, despite the changes and major developments in information technology, particularly microcomputers and networks, no effort has been exerted to enhance LIMS since the issuance of the current version.

This study has made a modest attempt to address two of the basic drawbacks of the current version of LIMS. These are:

1. The five LIMS modules are provided as separate executable programs. A user has to exit from one module in order to load another module. This process is time consuming and is not user friendly.
2. In its present form, the LIMS system cannot be used under a multi-user environment. Thus, the LIMS databases are not sharable. This results in multiple installation of LIMS and multiple LIMS databases in an organization. Such duplicates result in data redundancy, data inconsistency, inefficient use of resources and difficulty in management control.

This study aims at integrating the modules of LIMS, and investigating the considerations in network programming and enhancing LIMS for network so that the LIMS data set files could be accessed simultaneously to update records and files, extract data and produce reports. It is hoped that such an exercise would be beneficial to the users of the software.

1.2 OBJECTIVES

The overall aim of this research project is to investigate issues that should be addressed in network programming and to demonstrate how such issues could be implemented by enhancing LIMS for use in a multi-user (network) environment using the tools that are at the disposal of the candidate.

The specific objectives of the project are to:

a) Integrate the five LIMS modules through a common user interface menu;

b) Investigate the issues and concerns in programming at the network operating system level;

c) Describe the possible design to be followed in enhancing LIMS;

d) Use the Clipper language to develop a demonstrator for an integrated multi-user LIMS.

1.3 SCOPE AND LIMITATIONS OF THE STUDY

LIMS is being used by agricultural research organizations mainly in Africa and other developing countries. Many of the users have already huge data sets which are being used through the provisions of the software. Thus, the current work does not attempt to introduce any change to the original database design of the system. It rather tried to enhance the LIMS software so that it becomes a multi-user and more user-friendly application that utilizes the data files which were created and in use already.

Since the main purpose of the study is to investigate the technical issues in network programming at the network operating system level and to demonstrate those issues by enhancing LIMS, a formal user study is not taken up in this study.

LIMS is a very huge program with many hundreds of pages of code. Only the functions and facilities which are relevant to this study are reworked and presented here.

Due to constraints of time, programming tools, and application program interfaces available to the study, the task of enhancing LIMS is limited to implementing some of the designed modifications.

Due to inaccessibility of other operating system platforms, and due to the fact that the programming language used to develop LIMS utilizes the DOS function calls, DOS network services are considered and utilized in the current work.

1.4 METHODOLOGY

The need for enhancing* LIMS for networks came from the queries of the major users of the LIMS software at the International Livestock Research Institute (ILRI) based in Addis Ababa office. Since the candidate also works for the said Institute, the topic under consideration became an interest worthy of investigation.

The methodology followed in addressing the issues in network programming, designing and implementing the changes is based mainly on literature review. A review was made to understand the issues that should be addressed in network programming. At the same time, the candidate studied the CA-Clipper language in which LIMS was originally developed. Besides, the LIMS software was reviewed using the system documentation, running the system, discussing with users, and studying the LIMS source code.

---

* Please refer to Annex I for comments on the need of enhancing LIMS

With a basic understanding of the issues in networking, the provisions and limitations of the programming language and the LIMS software were established, the possible design to be followed in the implementation was also drawn. Finally, some of the modules of LIMS were enhanced for demonstration purpose.

1.5 ORGANIZATION OF THE THESIS

The thesis is divided into five chapters. Chapter one is the introductory chapter. Chapter two reviews many of the concerns that should be addressed in network programming. It also reviews the Clipper network programming commands, functions and implementation provisions for network programming. Detailed technical review of the LIMS software and the design for enhancements are presented in chapter three. Chapter four deals with the implementation of enhancing LIMS and demonstrates some of the implemented facilities. The conclusions of the current work and recommendations for future work are presented in chapter five. Finally, the list of bibliographic references used in the thesis are annexed.

CHAPTER TWO

NETWORK PROGRAMMING CONSIDERATIONS

2.1 INTRODUCTION

A good multi-user program should allow sharing and simultaneous access to data. It should allow concurrency even to qualify as a multi-user program. Multi-user programs should also ensure that the data being shared is never damaged by the concurrent accesses. It should ensure the integrity of the data at all times, and it should avoid all possibility of data corruption due to simultaneous access to the data files. Another mark of a good multi-user program is consistency. If possible, all users on the network should have the same view of the data even if it is being simultaneously updated (James 1989).

Developing multi-user applications which allow concurrent access while keeping the integrity of data in a consistent way raises a number of programming problems that do not occur on a single user system. In this chapter an attempt is made to investigate these issues closely. Because of their general importance, the provisions of workstation operating systems, particularly DOS, and that of a network operating systems are reviewed first.

2.2 APPLICATIONS PROGRAMMING INTERFACES

Applications Programming Interfaces (APIs) are standard interfaces allowing programmers to produce software in a logical and consistent fashion. They are a suite of function calls allowing programmers to write applications which use particular operating systems, operating

system components, or similar services. APIs offer the advantage of building custom applications rapidly and relatively easily. APIs are intended to define standard sets of calls to operating system and other services to simplify development in complex systems. Published APIs invariably help software developers by providing standard ways to write these calls (Herman and Hess 1996).

2.2.1 DOS Network Functions

The DOS operating system is a piece of software that coordinates the operation of all the hardware components of a computer system. DOS interprets commands issued by users and issues appropriate commands to the hardware. DOS also makes these services available to application programs through a series of interrupts (Booth and Lief 1993).

DOS provides several software interrupts for application programs. Interrupts are signals (pieces of codes) that cause the computers' central processing unit to suspend what it is doing and transfer to a program called an interrupts handler. The interrupt handler determines the cause of the interrupt, takes the appropriate action, and then returns control to the original process that was interrupted. Since DOS might be loaded at any memory location, this provides a sure simple way for applications to call it.

DOS contains a set of functions available to programs as services. DOS interrupt, INT (33) 21H, is by far the most common DOS interrupt that provides services for using and maintaining the file system. By using function codes in the AH region, this interrupt invokes dozens of services for a user program. The two main DOS elements in network programming

are DOS API functions, function 61 (3DH) for opening files and functions 92 (5CH) for locking and unlocking regions of a file. DOS function 104 (68H) commit file is also of paramount importance in a multi-user environment.

2.2.1.1 DOS Open File Function

DOS function 61 (3DH) is used to open files. It is used both under a single user and a network environment. When this function is called, an 8-bit number called the 'open mode' is often used. This function breaks the 8-bit open mode into two 1-bit and two 3-bit sub-fields, and interprets the values in each sub-field in a special way. The values in the two 3-bit sub-fields are taken as instructions to open the file in specified ways that bestow certain capabilities and restrictions on the program when it comes to using the file later (Morgan 1988a).

Tables 2.1a and 2.1b show the parameters of DOS function 61 (3DH) and the 8 bit open mode details, respectively.

Table 2.1a Parameters for DOS function 61 (3DH)

Parameters for function call 3DH	
Register	Description
AH	3DH
AL	OPEN MODE
DS:DX	POINTER TO FILE NAME

Table 2.1b Values for DOS function 61 (3DH) - Open Mode

Mode	Bits	Values	Interpretation
Access Mode	0-2	000	Read Only (Read access)
		001	Write Only (Write access)
		010	Read and Write (Read/Write access)
Reserved	3	----	Reserved
File Sharing Modes	4-6	000	Compatibility mode
		001	Deny All (No other programs may Read or Write)
		010	Deny Write (Other programs may Read but NOT Write)
		011	Deny Read (Other programs may Write but NOT Read)
		100	Deny None (Other programs may Read and Write)
Inheritance	7	0	Child process inherits handle
		1	Child does not inherit handle

The open mode encompasses the **access** and **sharing** modes. The lower three bits (0-2) determine the access mode of a file. The access mode is important even in single user environment to open a file. The open mode in a single user environment will consist of the access mode and compatibility mode for the sharing mode. If the open mode contains disallowed values for the sub-fields, the function will return an error and will not open the file. The access mode determines the interaction of the application with the file itself. It declares which operations a program will perform with a file upon opening it (IBM 1987; Morgan 1988a; Tischer 1992)

The access mode tell DOS which of the three levels of restrictions to place on the usage of the file while it remains open: Read, Write, or Read/Write. If the file is successfully opened, the restrictions apply throughout the interval it remains open, and cannot be changed (Morgan 1988a).

There are two rules of reading and writing that can lead to errors (Morgan 1988a; *Duncan 1988):*

- The terms of the target file should not be violated on opening the file. If the file has read-only attribute, it cannot be opened with read/write access.
- Once the file is open, the original access mode under which the file was opened should not be violated.

Bits 4 through 6 define the operations the other programs are allowed to perform on the specified file in a network setting. They tell DOS whether or not to let subsequent attempts to open the file succeed (IBM 1987; Morgan 1988a). So an application program can determine whether access should be completely denied or to allow only to read the file, write to the file or to allow reading and writing rights.

At open time, the sharing mode setting interacts with those of predecessors, if any, who hold the file open. Thereafter (assuming the current open succeeds), it interacts with successors who might try later (Morgan 1988a).

2.2.1.2 DOS Lock Or Unlock File Region Function

The most important aspect of simultaneous file access by more than one workstation is record locking. Records can be locked with a single DOS function, 5CH. This function requires the correct file handle and the offset address and length of the segment to be locked. Table 2.2 presents the parameters of DOS function 92 (5CH).

Table 2.2 Parameters for DOS function 92 (5CH)

Parameters for function call 5CH	
Register	Description
AH	5CH
AL	Mode: 00 = Lock, 01 = Unlock
BX	File handle
CX:DX	Start offset in the source file as 32 Bit value
SI:DI	Segment (record) length as 32 Bit value
BU	Parameters for function call 5CH

This function locks or unlocks the specified region of a file. The following rules apply to this function:

- This function is useful for file and record synchronization in a network environment. Access to the file as a whole is controlled by the access mode, file sharing parameters passed in open or create calls and by the file's attributes which are stored in its directory entry.
- The beginning location in the file to be locked or unlocked should be supplied and it is used as a byte offset into the file. Similarly, the length of the region to be locked or unlocked should be supplied.
- For every call to lock a region of a file, there must be a subsequent unlock call with exactly the same file offset and length.
- Locking beyond the current end of file is not an error.

The network operating system, ensures that locked records cannot be accessed from other sources by displaying an error message when these function calls occur.

2.2.1.3 DOS Commit File Function

DOS Commit File function 68H causes all buffered data for a file to be written to the device. It forces data in MS-DOS's internal buffers associated with a specified handle to be physically written to the device. This function can be used instead of the close-open sequence. Commit File offers the advantage of not failing due to lack of handles, and the application does not risk loosing control of the file in network environment. It provides a faster and more secure method of committing data in multi-user environments (IBM 1987; Duncan 1988).

2.2.2 Network Operating System Services

The Network Operating System runs on the file-server computer. Its primary function is to accept data requests from the workstations and to respond by providing the requested data or results. A network operating system always consists of a server program, which manages the file server, and a workstation program, which manages each workstation and allows it to address the server. Workstations must be able to use DOS as usual and must access the file server like a normal floppy or hard drive (Tischer 1992; Booth and Lief 1993).

Most networks have what is known as a network shell, which is the program that re-maps the DOS file-access routines to its own code. Some network shells also re-map DOS printer routines, date and time services, and so on. In this manner, the network can seem transparent to a wide variety of applications running on the workstations (Booth and Lief 1993).

The network software intercepts all file operations and passes only local operations, which access the workstation's own disk space, to DOS. Requests which address the server are handled by the network operating system instead of DOS. The application submits a request to DOS, which transforms it and sends it on to the storage device. At that point the network operating system examines the file specification to determine whether or not it references a non-local derive under network management. If so, the request is diverted to that derive (Morgan 1989; Tischer 1992).

The network operating system also provides a host of services which are waiting to be tapped by application programs in order to communicate with the network. Novell, Inc., the company that sells NetWare network operating system, for instance, has released a set of Application Program Interfaces (APIs) that give programmers access to almost all information available with the operating system (Booth and Lief 1993).

All NetWare APIs are available through the DOS interrupt system. These APIs are accessed through DOS interrupt (33) 21H . This interrupt is mapped to NetWare when the shell is loaded, and NetWare looks at all calls. If a call is for a network service, NetWare passes control to one of its own routines; otherwise, it passes control to DOS for handling. (Booth and Lief 1993).

When programming in a network environment, understanding and making proper use of the provisions of the underlying disk operating system (DOS in this case), the provisions of the network system on which our application is to run, and the way the programming language,

used for the development of such application system, implementing these provisions are very important.

2.3 NETWORK PROGRAMMING ISSUES

Under a network environment, there are two or more users who share resources such as hard disks, printers, data files, application programs, etc. Since these users may request access to such shared resources and such request can be conflicting , network programming requires planning for worst-case scenarios. How can we make sure that one user's change does not over-ride that of the other user? How to make resource sharing as effective as possible in our application programs? In this section, we will review such concerns.

2.3.1 File Opening Issues (Concurrency)

In a single user environment, files are opened for single use. Thus only the access mode, which determines the activities that could be performed on the file by the application, is important and thus implemented. Under multi-user (network) environment the sharing mode, which determines the rights of other users on the opened file, is very important as well. The application system, therefore, has to address both the access mode, and the sharing mode (Morgan 1988a).

Files are either opened in non-shared mode or in shared mode. When files are opened in non-shared mode, only the current application (user) has access to that file. Other users can neither read nor write to the file. Non-shared mode is implemented by using the sharing mode **deny**

all of DOS open file function. In a multi-user environment, great caution should be exercised in deciding to open files in non-shared mode, since it denies all other users access to that file.

When files are opened in shared mode, besides the application which opened the file, other users are also allowed to read from, write to, or read from and write to the file. DOS provides three levels of file sharing by allowing others to: only read from the file (**deny write**), only write to the file (**deny read**), or both read from the file and write to the file (**deny none**). The implementation details of the file opening mode, in a network environment, depends on the programming language to be used in developing the application program.

2.3.2 File Contention Issues (Integrity)

In a multi-user environment, files are mostly opened in shared mode. In fact, the ability of opening files in shared mode and its being used by two or more applications / users at the same time, is the minimum requirement of a multi-user application. When files are opened by two or more applications at the same time, the problem of contention for such shared resource arises.

One of the possible ways of solving such a problem is the implementation of locking mechanism. The locking mechanism is a mechanism of giving one of the users full access right to the shared resource and denying other users from accessing the resource temporarily. A lock is used as a signal to other users that the file/record is currently being updated and hence off-limits to them (Booth and Lief 1993).

2.3.2.1 Locking Types

Two types of locking are possible on files opened in shared mode: record and file locking. Record locking is the weakest locking alternative. A record lock allows the update of a single record in the database. It denies any other user the ability to update that record until the current lock is released. Other users are allowed to read the locked record but are not allowed to make changes to the locked record. Other users may lock other records in the same file and make changes to them. Therefore, it is possible to have many locks on different records within a file from different application programs at the same time.

In programming for network, it is good and advisable to use record lock for much of the time as it accommodates many users. It does not deny users access to shared data file and the lock on the records normally stays for a very short time. Although for much of the time simultaneous access can be handled using nothing but record locks, there are times when changes have to be made that affect the entire file. In such circumstances, the need for locking the entire file becomes a necessity (James 1989).

Locking a file is similar to opening a file in non-shared mode for a brief period of time. No other user can update any information in the file until the lock is released. A significant difference between a file lock and opening it in non-shared mode is that other users can continue to read information from a locked file (Booth and Lief 1993). They cannot change it, however.

File locking is usually used when an application wants to make global changes that affect more than one record in the file. Locking an entire file should be done judiciously, since other users are denied the right to make changes for the duration of the lock.

Record or file locks are only required in modules which write data to files. Modules which only extract data from files require no lock.

2.3.2.2 Locking Strategies

There are two possible locking strategies which could be enforced on records or files: pessimistic and optimistic locking.

Pessimistic (static) locking strategy is a locking system where the application program searches for a file or record to be locked and upon gaining control locks the file. Afterwards, modification to the data are effected, and written to file. When modifications are completely written to the shared disk the lock is released. Thus, this locking strategy is expressed as **Find-Lock-Edit-Write-Unlock**. This locking strategy guarantees the editing user that what is on his screen and what is actually on the shared disk are identical. The biggest problem with this strategy is that the duration of the lock interval is unlimited. Thus, it obstructs all other users who would like to edit the same record or file for an indeterminate length of time. This leads to what is commonly known as the lunch-time-locking (Booth et al. 1992; Spence 1989; Blue 1995)

The optimistic locking strategy propagates that an application program should locate the record or file to be edited, make the modifications needed, then lock it and write the changes to disk before releasing the lock. This locking strategy is expressed as **Find-Edit-Lock-Write-Unlock**. Under this locking strategy, the data is available for editing at all times. It is never locked for an extended period, only for brief write intervals. Its drawback is that if another user a step ahead of the current user performs the Lock-Write-Unlock operation, the current user will not know it. She/he will continue to edit the data on the screen, unaware that it is no longer what is on disk. Locks based on this strategy may lead to timing problems discussed later.

2.3.2.3 Locking Problems

There are various problems related to locking. In this section we discuss some of these problems.

Lunch-Time Locks. The lunch-time lock occurs when a record or file lock is placed on a file, and the system allows a wait state before the system unlocks the file. This situation commonly arises where a user will edit a record, then walk away from their computer for an extended period of time. This prevents other users from being able to lock that same record or file until that user returns to exit the edit screen, and unlock the record/file. In a network that depends on simultaneous multi-user access to records and files, this can cause a severe bottleneck (Booth et al, 1992; Blue 1995).

Timing Problems. Timing problems occur when a user transfers the data into memory, and while he/she is working with the memory copy of the data, some other user updates the same record on the shared disk. This happens when the first user makes the modifications without locking the record or file with the intention of gaining the lock just before writing the changes to disk. In this situation, one of the users will lose their changes (Booth et al 1992; Blue 1995).

Even if the application program takes the precaution of checking the record to be sure that it has not changed since the editing session began, the user will still lose any changes made to the data, and must edit the record again.

Dead-Lock (Deadly Embrace). If two network users both need to access two files, say files A and B, then dead-lock can result if they both attempt to lock the files in a different order. The first user cannot continue because it is waiting for the second user to release the file it has locked. Unfortunately, the second user is waiting for the first user to release its files before it can continue. We are concerned here with the failure of two well behaved programs that have conflicting resource requirements. The problem is that they require multiple file locks. (James 1989; Booth et al. 1992)

2.3.3 Semaphores

Most locking schemes make use of some kind of "in-use" indicator or flag called semaphores. Semaphores are methods of communication between applications. They help in signaling to

prevent users from interfering with each other. Semaphores are usually used in the process of resource sharing (James 1989; Morgan 1989).

The first program that wants to use the resources sets the flag. As the subsequent program tries to make use of the same resource it will find the in-use flag set and waits until the first user finishes with the resource and resets the flag. Semaphore is a bit of code that functions as a conditional roadblock to program flow. It is a check point. Pass the test and go through, or fail and wait or turn back. In a programming language that does not support direct locking of some file types, an application can develop semaphores codes so that those file types could be locked and their use is synchronized (Weber 1990).

Problem arises if both users look at the in-use flag at roughly the same time and both decide to set it and use the resource. This problem arises if the operation of testing and setting of the flag occurs in two separate steps. If the test and set operations were to be performed as one single indivisible operations then the problem would vanish (James 1989).

2.3.4 Update Visibility (Consistency)

Since data files are accessed and are updated by different users, in a network environment, the issue of update visibility is a very important one. Normally, RAM work areas (buffers) are used by applications to temporarily store changes done during a session. The updates are later on, when the computer has lesser task, written to disk. Thus, considerable time period could lapse between the time of modification and the time of disk writing. In a network environment, changes made by a user/application may not be visible to other users since they

are not written to the shared disk as soon as the changes are made. In order to make the changes visible to other users, the application should make sure that the changes are written to the shared disk. Changes are permanently written to the appropriate physical device when the application program specifically calls the DOS Commit function.

If the changes are not written to the shared disk as soon as possible, releasing the lock enforced during the write changes does not guarantee that changes are written to the data file on the shared device. DOS function 104 (68H) forces DOS to physically write the data in its internal buffer to a disk.

2.4 THE CLIPPER LANGUAGE

Clipper includes numerous built-in database management commands and functions. These Clipper commands and functions are high-level interfaces to the various DOS functions. The interaction between Clipper and the network is handled through DOS function calls. Clipper does not communicate directly with the network operating system, but rather uses the DOS function calls to handle all access to network files. Most networks adhere to these DOS calls and Clipper only uses these function calls for network processing. This allows Clipper applications to be used on a large variety of networks (Booth and Lief 1993).

According to Computer Associates (1992c), the company that produces and sells CA-Clipper, the "only requirement for using CA-Clipper with a Local Area Network is that the LAN must adhere to DOS 3.1 or greater function calls". Clipper uses DOS calls exclusively for all

network related operations and, consequently, applications that are compiled with clipper can run on any LAN designed to the DOS standard.

Clipper's database management is provided for a file known as .DBF, or Data Base File. It uses an index (.NTX) file that allows a logical ordering to be placed on a .DBF that may be in completely different physical order.

2.4.1 Clipper Open Modes

Clipper utilizes the DOS open mode, function 61 (3DH), in order to allow multiple access of a shared file. Since this DOS function was incorporated in DOS 3.1 or above, each workstation on which Clipper applications run must use DOS 3.1 or higher to allow the application to share files (Morgan 1988a; Morgan 1988b; Spence 1989; Booth et al 1992; Booth and Lief 1993).

Clipper allows for two open modes, exclusive and shared mode. The exclusive open mode corresponds to the **deny all** mode of the DOS open mode function 61 (3DH). When a database file (.DBF) is opened in exclusive mode, only the opening application (user) will be allowed to access the file.

When a database file (.DBF) is opened in shared mode, other users are allowed to read, write, or read and write to the file. The synchronization of shared files are handled through clipper's locking mechanism.

Clipper utilizes its USE command to open files. The USE command has two parameters which are of particular interest in a network environment: the name of the file and the open mode. These parameters are translated to the appropriate DOS file open modes by clipper's internal functionality.

2.4.2 Clipper's Locking Mechanism

Clipper allows for two types of locks: record and file locking. Clipper utilizes the DOS lock or unlock file region, function 92 (5CH) to handle its record and file locking. Clipper's locking mechanism works only for the two file types it supports, .DBF and .NTX file formats. Clipper does not require that the developer perform index file locks, but rather handles these locks transparently.

As per DOS, when a portion of a file is locked, all access (both reading and writing) to that portion is denied. Clipper gets around this by adjusting where the lock is requested. Instead of requesting a lock within the range of bytes within the file, Clipper attempts to lock the record by adding one billion to the record number, and locking one byte. Other clipper applications will check that byte when attempting to lock a record (Booth and Lief 1993).

Clipper employs similar scheme to lock a file. It locks the offset at one billion bytes from the beginning of the file. Other clipper applications will respect this lock when attempting to lock files.

Clipper employs its two functions, RLOCK() and FLOCK() to lock a record and a file respectively. These two functions make only one attempt to lock the record or file and return a logical true, (.T.), if the lock was successful or a logical false, (.F.), if it was not successful. A successful RLOCK() or FLOCK() releases any previous file or record lock placed on the file by the same user. Clipper's UNLOCK and UNLOCK ALL commands are used to release locks that are put on files. The UNLOCK command is used to release lock in the current work area. The UNLOCK ALL command releases locks in all Clipper work areas.

2.4.3 Clipper's Update Visibility

For each file opened in Clipper applications, Clipper sets aside a buffer (work area) in RAM to manage the database file with its associated index and memo files. The CA-Clipper database system defines 255 work areas, of which 250 are available to application programs and the remaining 5 are reserved for internal use (Computer Associates 1992c).

Any changes that an application program performs on the database file, are made to the copy of the database file which is in the Clipper work area. The Clipper COMMIT command is used to request that the updated information in Clipper's work-area buffers gets written to DOS. The COMMIT command checks the work-area for updated information and then transfers the information to DOS buffers. It then calls the DOS commit file function which writes the changes to disk (Booth and Lief 1993).

CHAPTER THREE

EXAMINING AND DESIGNING LIMS FOR ENHANCEMENT

3.1 EXAMINATION OF LIMS[1]

LIMS is a stand alone, single version software developed by the International Livestock Centre for Africa (ILCA), in 1991/92. The software is being distributed free-of-charge to national agricultural institutions in ACP (African, Caribbean and Pacific) countries. All other requesters are charged 250 US Dollars. According to ILRI's record of LIMS distribution, as at August 1, 1996 about 133 copies of LIMS have been distributed to 96 institutes and 37 individuals dealing with livestock research in 40 (36 ACP and 4 European) countries.

The LIMS software package is designed to facilitate the recording and processing of livestock data in developing countries. LIMS was designed to provide users with a computerized tool that allows them to manage livestock performance data sets. It was also meant to provide help in the decision making process of farm management.

According to ILCA (1992), the LIMS software package is designed to address:

a) Balance of flexibility and standardization of data definition;

b) Documentation and definition of data sets;

c) Assistance in deriving non-observed data (breed, parity, mating-parturition connection);

---

1 This examination of the LIMS software is based on:

- Review of the LIMS Documentation Manual;
- Running the software; and
- Studying the LIMS source code.

d) Data validation and error correction;

e) Reporting for animal management; and

f) Data extraction and calculation of standard performance traits for statistical analysis.

In order to address these various aspects of livestock data management, the LIMS software has a modular design, with different modules addressing each of the issues mentioned above. LIMS can be used to record data for various species of animals since it provides the user to have control over the data sets which are defined by the user. LIMS can run on a standard IBM compatible microcomputer without the need for any additional software.

The LIMS give-away system consists of five executing program files (.EXE), a generic data set consisting of the definitions of essential and useful data files and variables in one memory (.MEM) file and four database (.DBF) files, and three different memory (.MEM) files which consist of the default settings for the display video. Table 3.1 shows these files and their purpose/content in the LIMS software.

The generic data set can be individually configured by the user to suit personal requirements by changing existing data files and variables, as well as adding new files and variables. A newly defined data set should have four characters length and should start with an alphabetic character. For each LIMS data set file created, new versions of the generic data set files are created to hold the data set definitions. The first four letters of the defined data files will be the name of the newly created data set.

A LIMS data set is defined on 5 levels: data set; files; variables; codes; and breed rules. Standard definitions and default values are supplied for the first three levels (data set, files, and variables).

Table 3.1. LIMS current version, give-away files

File Name	*Purpose/Content*
LSETUP.EXE	Used to define a user data set
LUPDATE.EXE	Used to add and edit data in a user data set
LVALID.EXE	Used to validate data in a user data set
LREPORT.EXE	Used to create reports from a user data set
LEXTRACT.EXE	Used to extract data and calculate performance traits from a user data set
CFG_REF.MEM	System Configuration, dBASE standard memory variable file
FDL_REF.DBF	File Definition Library, dBASE standard database file
DDL_REF.DBF	Data Definition Library, dBASE standard database file
CDL_REF.DBF	Code Definition Library, dBASE standard database file
BDL_REF.DBF	Breed Definition Library, dBASE standard database file
CLRHERC.MEM	Hercules mode defaults, dBASE standard memory variable.
CLRMONO.MEM	Monochrome mode defaults, dBASE standard memory variable.
CLRLCD.MEM	LCD mode defaults, dBASE standard memory variable.

LIMS was developed in the Clipper language and compiled with Clipper 5.0 compiler. The LIMS program files can be edited using any ASCII editor and they are kept in .PRG files. All data records in a LIMS data set are stored in dBASE III+ compatible (.DBF) data files. The index files are in Clipper index (.NTX) file formats. In addition to .DBF and .NTX files, memory (.MEM) files are also used to store configuration values.

The LIMS software uses 61 program files and two header (.CH) files. Besides, it uses third party library files in addition to those of the Clipper programming platform. Since the current version of LIMS is developed in a way that the various modules could be used as stand alone programs, some files are duplicated in the different modules. For instance, there are five three-sets of library program files which contain almost the same collection of functions used in the different modules of LIMS.

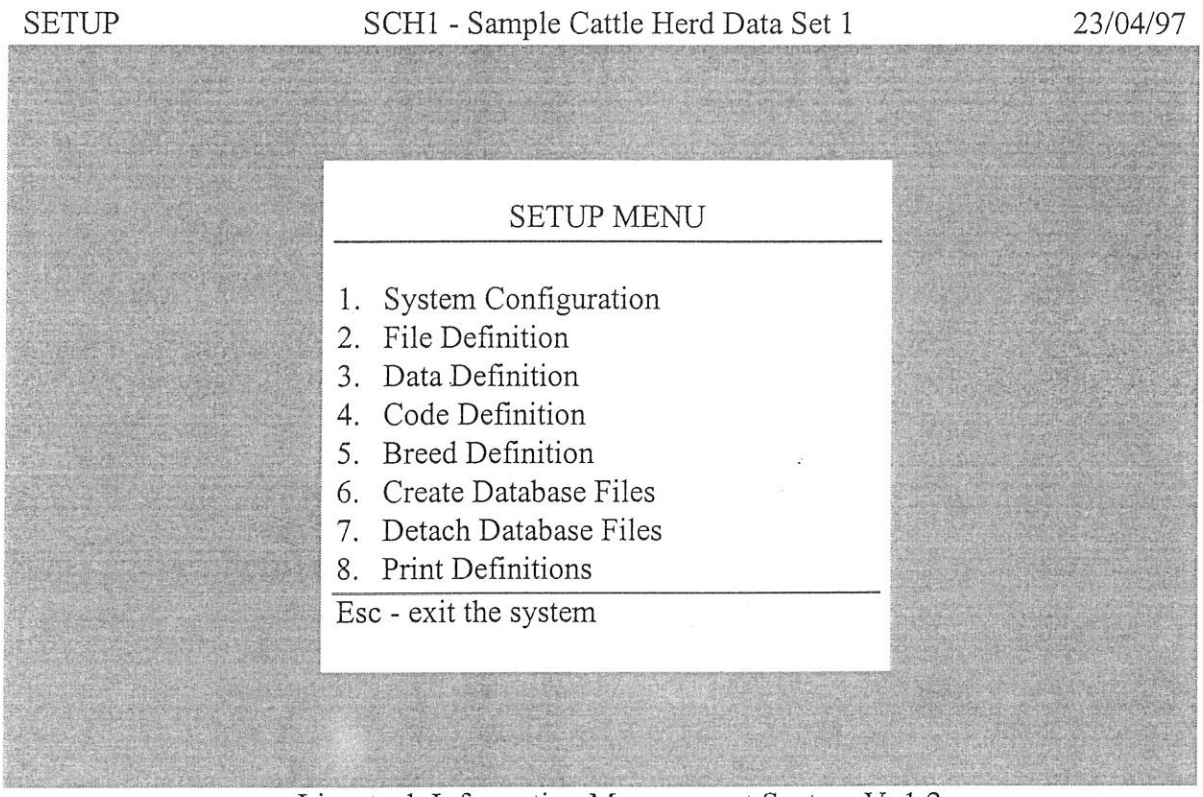
In the following sub-sections, the modules of LIMS will be closely investigated. The aim of the investigation is to find out valuable information needed in designing and enhancing LIMS.

3.1.1 The Setup Module

The Setup module is the only module that can create and modify the structure of a data set. This module is used to configure system values; add, modify, delete, activate and deactivate data files; add, modify, delete, and preview data fields (variables); add, modify, and delete codes; add and delete breed rules; create database files; detach database files; and print (to printer or to a file) the various definitions for proof reading and corrections.

This module has 13 program files and 8 menu options. Except for one of the options that prints the different settings, all the rest read definitions from the generic data set and write to memory and database files of the user defined data set. Figure 3.1 presents the SETUP menu options.

Figure 3.1 SETUP Menu Options



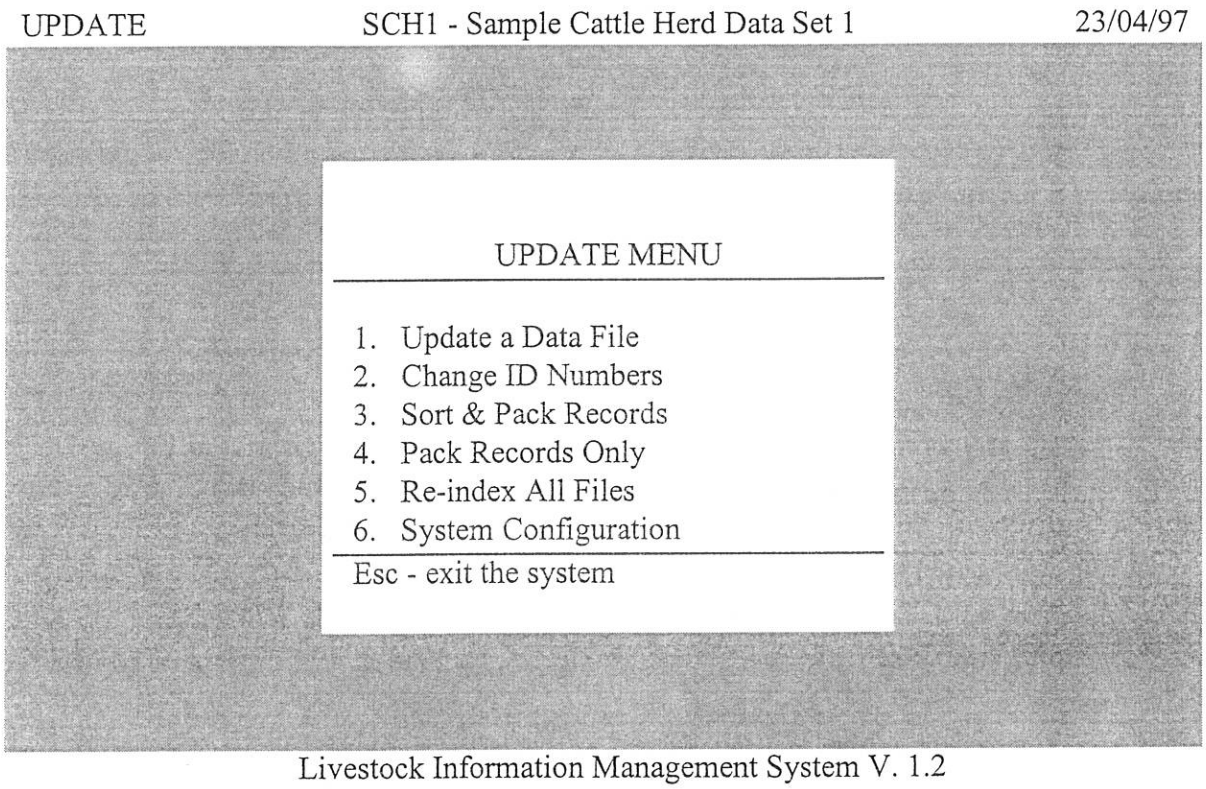
The defined data set level configuration values are stored in a memory file with the name XXXXCFG.MEM where XXXX is the data set name defined by the user. The other definition files are stored in database files prefixed by the data set name and suffixed by the definition library types (File Definition Library/FDL, Data Definition Library/DDL, Code Definition Library/CDL, and Breed Definition Library/BDL). Besides, user data files are created as per user definitions and selections.

3.1.2 The Update Module

This module is the only module that can modify (input and edit) raw data in LIMS data set. Data edition and modification comprises adding new records to files, changing field values in

existing records, and deleting records in a file. Besides this module allows the sorting and indexing of records in a database file. It also allows the marking of records for deletion and the permanent removal of deleted records. This module presents the option of changing ID numbers as well. The menu options of the UPDATE module are presented in figure 3.2 below.

Figure 3.2 UPDATE Menu Options



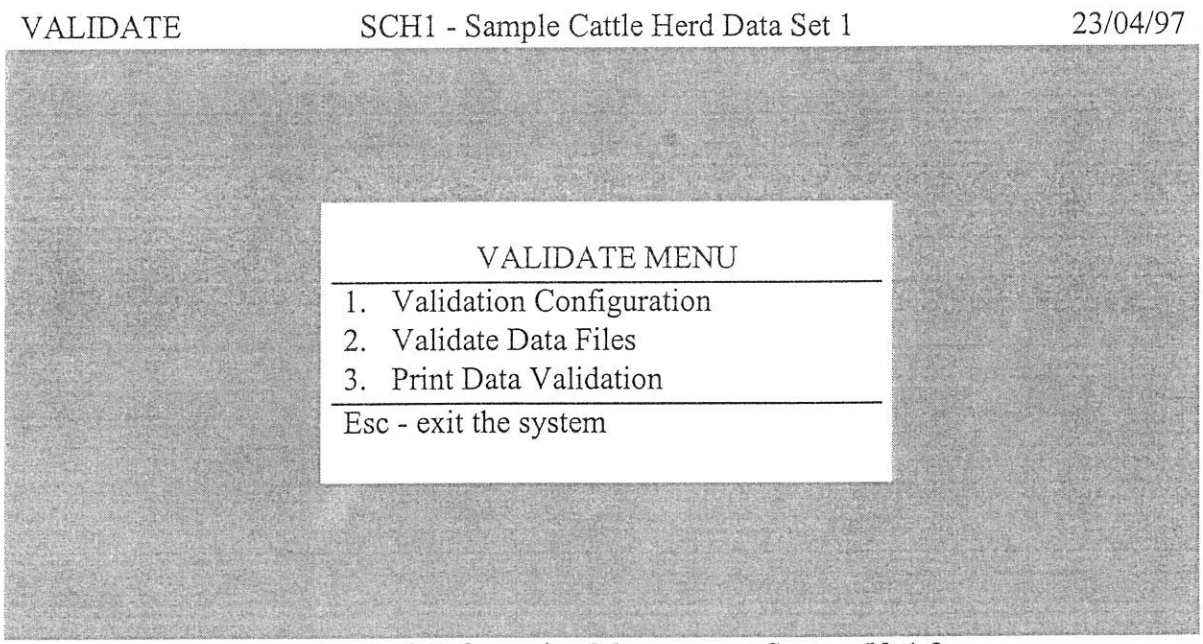
The UPDATE module has 17 program files and 6 menu options. All of the menu options of this module deal with tasks of adding, modifying and reordering of records in the data files. This is the module that is mainly used to enter user data records. The entered data is saved in database files selected/defined during the SETUP procedure.

3.1.3 The Validate Module

This is the part of the LIMS software that performs checks on data. The system performs checks on data as per configuration values specified by the user. The validation is done for each data file in the data set. Errors or inconsistencies encountered are compiled in the form of a printed list, and such list can be printed to a printer or a file.

The VALIDATE module uses 10 program files and has three menu options. Figure 3.3 presents the menu options of this module.

Figure 3.3 VALIDATE Menu Options



The first menu option lets users to enter configuration values to be used during the validation process. The configuration values are saved in a memory file.

The second menu option runs the validation process and subjects the various system and user defined database files to checks. During this process, the database files are checked for inconsistencies that might have been introduced during data entry.

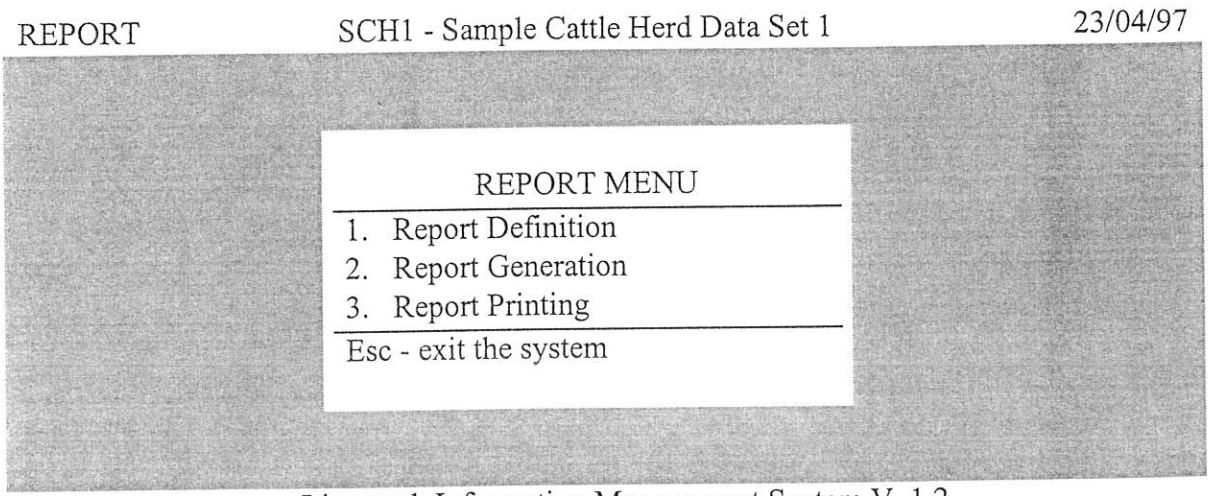
The third option prints the result of the recent validation process. The printed output will be used to make corrections to the data files as may be required.

3.1.4 The Report Module

This module is used to define, generate and print reports. It compiles reports on the animals and the data in the LIMS data set. There are several categories of reports based on the type of data presented in the reports. Within each category one can define individual reports according to user's specific needs and requirements.

The REPORT module uses 10 program files and has three menu options. Figure 3.4 presents the REPORT module's menu.

Figure 3.4 REPORT Menu Options



The first menu option is used to define report structures which will later be used to generate reports from the user data set files. The user can add, edit or delete report definitions using this option. A maximum of 99 report definitions may be created. The report definitions are stored in memory files.

The second menu option is used to generate report(s) using one or more of the report definitions. This option reads the database files in the user data set and writes the report into report database files.

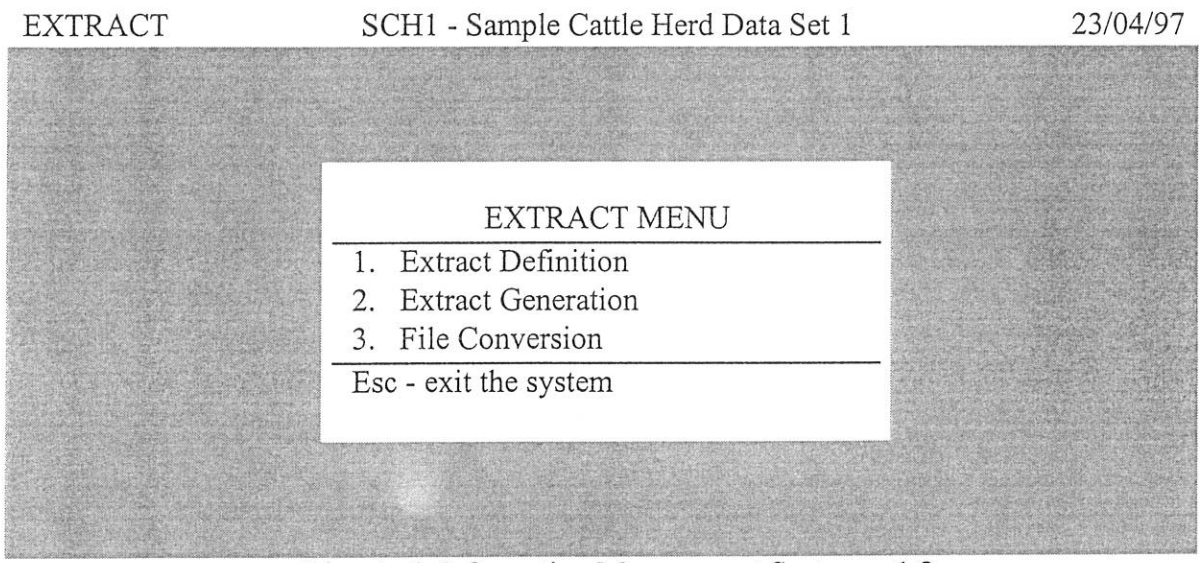
The third option prints the generated report either to a printer or a file.

3.1.5 The Extract Module

This module produces output files containing raw data and calculated performance traits extracted from LIMS data set on an animal and parturition basis. It also allows the conversion of LIMS data files and extracted files to System Data Format (SDF), Delimited, and Report ASCII formats. Such converted files could be called into statistical analysis packages for further data analysis.

The EXTRACT module makes use of 11 program files and has three menu options. The menu of the Extract module is presented in figure 3.5.

Figure 3.5 EXTRACT Menu Options



The first menu option helps in defining and modifying extract configurations. There are three extract types allowed and the extract configurations are kept in three memory files.

The second option uses the extract definitions to generate extracts from the user data set files. The generated extracts are kept in a database file. The third option is used to convert the extract database (.DBF) files to any or all of the three ASCII formats supported by the module.

3.2 DESIGNING LIMS ENHANCEMENT

The first question to be answered in the design of the enhancement of the LIMS software is why the enhancement is required and how to achieve those requirements in the enhancement process. The current study is intended to attain two basic objectives:

1. To make LIMS an integrated package accessed from one user interface menu.

2. To enhance LIMS so that users may share data files to:
    - Make simultaneous update to the shared data files; and
    - Generate reports and extract data from the database files.

In the sections that follow, we will discuss the design issues in enhancing LIMS.

3.2.1 Designing the Common User Interface for LIMS

Designing LIMS as an integrated single menu system requires structuring a common platform that interfaces with each of the modules. As has been mentioned above, the LIMS modules are menu based. To provide consistent user interface capacity, the common user interface for the integrated and multi-user LIMS version also follows the same pattern.

Clipper supports a module making structure similar to that of the C language. It has one main and other subsidiary functions and procedures as needed. Since the current version of LIMS has presented the modules as stand alone separate programs, LIMS had five main functions and other sub-functions. The integrated LIMS version with a common user interface is based on the following design.

1. A *Main()* function which initializes local and global variables, sets screen colors, displays welcome screen, checks for default path, checks the existence of library reference files, validates the data set, checks for a data set library files, and calls the LIMS main menu.
2. Another function which displays the LIMS Main Menu, accepts users selection, and proceeds selection processing as per user requirements is also incorporated.

3. A third function is defined to be used by the *Main()* function in setting the data set code as per parameters passed by a user at the LIMS DOS prompt.

3.2.2 Designing LIMS for Network

The main reason why LIMS has to be enhanced for network is to enable data sharing. Data sharing is required to update data simultaneously and to produce reports and extracts from the databases. The design issues addressed in this study deal with how to allow multiple access to the LIMS database files while ensuring the integrity and consistency of the files. Below are some of the general design issues that are to be followed in enhancing LIMS.

1. Data files will be opened in shared mode except where the Clipper PACK and ZAP commands are used. Where PACK and ZAP are used the files should be opened in exclusive (the DOS deny all) mode. In such cases the Clipper USE ... EXCLUSIVE command will be employed.
2. File open command's success will be tested using the Clipper NETERR() function.
3. The Clipper REINDEX command will be abandoned since it only recreates the index pages without recreating the index header and thus does not solve problems related to corrupt indexes.
4. Though Clipper supports the opening of index files on the same command line that opens database files, this will not be used through the enhancement of LIMS. An attempt is made to open the database file and only if the database file open is successful, the index files will be opened with SET INDEX TO command.

5. Index files will be accessed in the same order in all programs to avoid wrong index information that can be caused by timing errors when indexes are updated in scattered fashion.
6. Indexes will not be created on open and unlocked shared files. Prior to index creation, the database files will be locked.
7. Since the index structure requires that the key size be the same for all records in the index, variable length records will not be used as index keys.
8. As index expression's code is executed every time the index is updated, simple index key expressions will be used so that the index update process would be fast.
9. For every lock on a record or a file, there will an unlock procedure.
10. Since Clipper uses work area buffers to keep changes made to a database file, the Clipper COMMIT command will be issued after a series of REPLACE commands so that the changes get written to the shared device.
11. The Clipper APPEND BLANK is used to add blank records before new records are added to the database files. After each use of this command, the program will check its success by using the Clipper NETERR() function.
12. The pessimistic locking strategy (*Find - Lock - Edit - Write - Unlock*) will be employed in enhancing LIMS. This strategy is the safest in ensuring that updates won't be lost.
13. Clipper makes only one attempt to lock a record or a file. This is, however, not enough in practical life. Thus, the strategy of retrying for a fixed period of time is adopted in this study. The lock will be attempted a number of times and when the allotted duration is over, the success or failure will be reported back.

14. Since Clipper does not support the locking of files other than .DBF/.NTX, semaphore locking will be used to lock the memory (.MEM) files used in LIMS to hold configuration data.
15. When two or more files are to be updated, updates are done only if all the locks are successful.

Specific design issues applicable to each module are discussed in the sections that follow.

3.2.3 Designing the SETUP Module

The SETUP module is used to define the global configuration variables, to select and define the data files, select and define the fields in each file, define the codes and breed rules, detach some of the files no more needed in the data set and print the definitions. After the initial SETUP procedure, this module is used only if any changes are to be made to the data set. Since changes made in this module have global effect, the options of the module, except the *Print Definitions* option, are accessed only by one person at a time. Normally it is required that one person is responsible for the creation and later modification of the SETUP configurations and definitions.

The design and utilization of this module then will follow the following path:

1. When a data set is created the first time, the user is required to define a password to be used for accessing the options of the SETUP module. This password will be used to protect access of the options of the module and only one person (database

manager) who defines data set files and who makes future modifications will be allowed access to most of the options of the module.

2. Other users may contact the database manager whenever they have alterations to the defined data set.
3. Since all the options of the module do write to database and configuration files, and possibly change the database structures as well, exclusive access is required before any change can be made.
4. A semaphore locking mechanism will be developed to protect the memory file used to store the configuration values.

3.2.4 Designing the UPDATE Module

After a data set is defined using the SETUP procedure, the next main task is entering the data into the database files. Data are input into the LIMS database files using the UPDATE module. This module is actually the heart of the LIMS software. Without the UPDATE module the LIMS software would be worthless. All the options of this module do alter the contents of the database files used in the system. The issues of multiple access of files while ensuring data integrity and data consistency are of special concern in the design of this module. The UPDATE module is the most important module to deserve all the care and attention in the LIMS network enhancement project.

One of the reasons for developing LIMS for network is to be able to simultaneously update records in shared data files. Since this module is meant exactly for such purpose, the issues of

simultaneous file opening, shared and exclusive use of files, record and file locking, committing disk writes, semaphore locking, etc. are all relevant here.

The design procedures that may be adopted for each of the options of this module are discussed in the following sections:

3.2.4.1 Update a Data File

This option is used to enter data records into the LIMS data set. When a user selects this option he/she will be prompted to select a database file which is to be used for record alterations. Once a database file is selected, a menu with various operation options is displayed. Three of the options of this menu deal with adding, editing and deleting a single record in the selected database file. Three other options affect all the records in the selected database file. One option is used to mark data fields in the selected database file for carry-over or jump field values during data entry. The mark data fields options does not alter the content of the database file and as a result is not of much concern in the design process for network.

The design for adding, editing and deleting a record in the *Update A Data File* option follows the path outlined below:

1. The database file selected should be opened in shared mode;
2. Since a single record is added, edited or deleted at a time, a record lock should be gained before each record update is effected.
3. After data fields are accepted, the DOS commit command which flushes memory contents to the shared device should be called.

4. After each commit call, the locked records should be released.
5. Besides, the design outlined for general application in section 3.2.2 will be applied.

The design for Sort /Pack process in the *Update A Data File* option will be as follows:

1. The database file selected should be opened in exclusive mode;
2. The sorting/packing action will then be performed;
3. The design outlined for general application in section 3.2.2 will be applied.

The design for index file creation process in the *Update A Data File* option will be as follows:

1. The database file selected should be opened in shared mode;
2. File locking will be gained before creating the index files;
3. The file lock will be released after the index files are created;
4. Besides, the design outlined for general application in section 3.2.2 will be applied.

3.2.4.2 Change ID Numbers, Sort & Pack Records and Pack Records Only

The *Change ID Numbers* option changes the ID numbers of animals in the whole user data set files. It checks every file on file-by-file basis and every record in a file on record-by-record basis.

The *Sort & Pack Records* option rearranges the order of the records in a database file and removes records previously marked for deletion. The *Pack Records Only* option on the other

hand only removes records that were marked for deletion from the database files permanently. Thus, these three options require exclusive use of the files in the LIMS data set before any of them is executed. These options have global effect on the data set. Therefore, they should normally be used infrequently and only one responsible person should access them. The following design is recommended for these options:

1. The menu option should be protected by a password;
2. The files should all be opened in exclusive mode.

3.2.4.3 Re-Index All Files

This option is used to re-create the index files for the database files used in the data set. The index files rearrange the records in each database file for easy access. The only requirement to use this option is gaining file locks on them. Thus, the following design is used for this option:

1. Each database file should be opened in shared mode;
2. A file lock is attempted on each file in the data set;
3. Index files will be recreated only for files which are successfully locked.

3.2.4.4 System Configuration

This option displays the default configuration values set during the SETUP procedure and prompts some of them for user change. The values are stored in memory variables. These configuration values are used during the current session of the UPDATE process. Thus, this

option does not require any special treatment while enhancing LIMS for multi-user environment.

3.2.5 Designing the VALIDATE Module

Once the LIMS data set is setup and the data records are input to them, the next step would be cleaning errors that were introduced during the data entry phase. The VALIDATE module helps in identifying such errors by conducting checks on the database files, so that corrective actions are taken. Though the VALIDATE procedure reads only from the database files, the data read should show the correct state of the data files. Therefore, care should be exercised so that either the validation configuration values or the contents of the database files are not changed while checks are being conducted on the database files. As a result, the design of the module should follow the following path:

1. The module should be protected by password to limit its access;
2. Each database file should be opened in shared mode;
3. File lock should be gained on all database files before the VALIDATE procedure is initiated;
4. The *Print Data Validation* option does not require password protection since it doesn't alter the contents of the database files.

3.2.6 Designing the REPORT and EXTRACT Modules

Since the REPORT and EXTRACT modules of the LIMS software have similar purposes and designs, they are discussed in one section here.

Upon completion of making corrections resulting from the VALIDATE procedure, the next step would be producing reports from the database files of the LIMS data set. The REPORT and EXTRACT modules provide the means of producing such reports and extracts from the LIMS database files. Besides, the EXTRACT module produces output files by extracting data from LIMS data files based on calculated performance traits.

3.2.6.1 Report Definition and Extract Definition Options

The *Report Definition* option is used to add, modify, and delete report definitions to be used to produce reports from the database files of the LIMS data set. The *Extract* Definition option is used to modify the values used in the *Extract* Generation process. The report and extract definition values are stored in memory (.MEM) files. The following design path will be followed in enhancing these options for network:

- Since Clipper does not provide a mechanism of locking for memory files, a semaphore locking should be developed to communicate the status of the memory files to other users. This will be accomplished through a flag which will be set to indicate that the file is in use or released to indicate that the file is free. Thus, the report or extract definitions will be altered only if the in-use status for the specific file show that the file is free.

3.2.6.2 Report Generation and Extract Generation

The Report Generation and Extract Generation options are used to generate reports or generate extracts from the database files using their respective definition(s). In the current

version of LIMS, the resultant reports are stored in database files which have the same file name as the report files with .DBF formats. This has a practical problem in a multi-user environment, since multiple report generations or extract generations by different users could overwrite the database files. To ensure that users report files and/or extracts are not deleted by other users, the following design is recommended for these options.

1. A default path for work file(s) will be defined when the system is started.
2. The source data base file(s) (LIMS user data set files) will be opened in shared mode;
3. When reports are generated and/or extracts are made, users will be required to specify the path and directory where the work files are to be stored. Users may utilize the default path to store such files or may specify different ones;
4. The destination file(s), that hold the outputs will be opened in shared mode;
5. A file lock will be gained on the file(s);
6. The outputs will then be written to the file(s).

3.2.6.3 Report Printing and File Conversion

The Report Printing option is used to print the reports generated from the database files according to specific report definition(s). The File Conversion, on the other hand, convert the result of the Extract Generation procedure to the three file formats supported in the LIMS conversion process. In the current version of LIMS, the files to be printed or converted are selected using the Report Definition and Extract Definition configurations, which are presented as a list on the screen. For the same practical reasons mentioned in section 3.2.6.2, these options require change as follows:

1. When the user wants to print or convert files, he will be asked to specify the path. The default path defined at system startup will be used if the user doesn't specify one.
2. The files in the default path will be displayed for user selection.
3. Printing or file conversion will progress after user selection is accepted.

CHAPTER FOUR

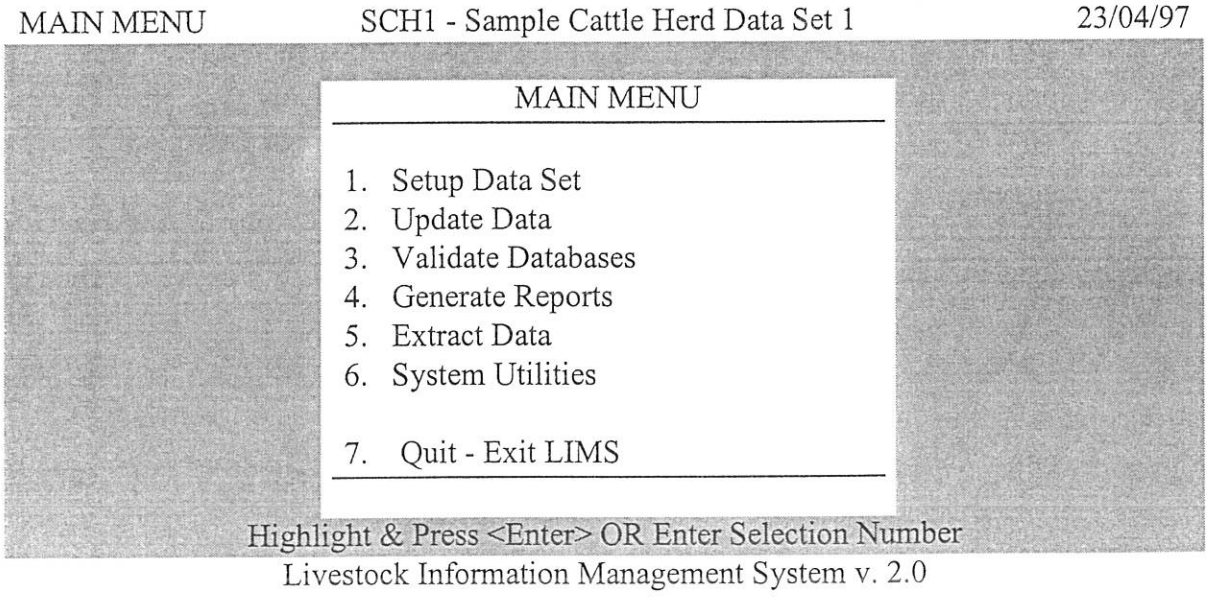
IMPLEMENTING LIMS ENHANCEMENT

In the preceding chapter, the design for the enhancement of LIMS was dealt with by discussing each module and its unique requirements. In the current chapter, the implemented facilities, utilized in the enhancement of one or more the modules, are presented section by section with some examples of such implementations.

4.1 INTEGRATING THE LIMS MODULES

The first task of this study was to integrate the various LIMS modules. Integrating the modules reduces access time and increases the user friendliness of the software. Accordingly, the five modules of LIMS have been integrated and a common user interface menu has been implemented. Figure 4.1 shows the new integrated LIMS main menu.

Figure 4.1 LIMS User Interface Menu



Highlighting the required option and pressing the <Enter> key or entering the option number would start the selected module. Options 1 through 5 correspond to the five LIMS modules. Selecting any of the options 1 through 5 would pop-up the menu corresponding to the specific module. Selecting option 1, for example, brings the SETUP menu as shown in Figure 4.2.

The sixth option of the LIMS main menu is added for future incorporation of system utilities for the conversion of non-LIMS data to LIMS, appending data from other LIMS data sets, printer selections, etc. Pressing this option in the current version would pop-up the following message at the bottom of the screen.

SORRY ... UTILITIES NOT AVAILABLE IN THIS VERSION!

This message would be displayed for 3 seconds and then the cursor moves to option 1.

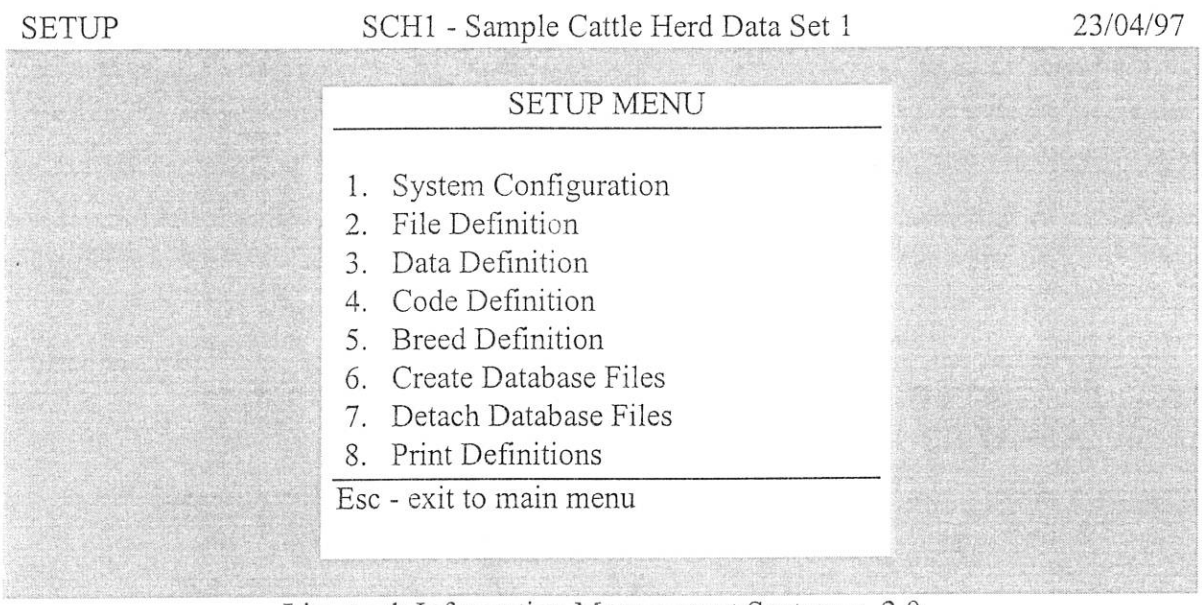
Though the <Esc> key is used to progress one menu level up in all the modules, on the main menu a separate explicit quitting option is provided in order to avoid exiting the system with an accidental press of the <Esc> key.

If a data set is just being created, only option 1 "*Setup Data Set*" would be the right option. In such cases, if an option other than option 1 is selected, the following message would be displayed on the last line of the screen.

ERROR - SYSTEM SETUP MUST BE COMPLETED FIRST!

This message would be displayed for 5 seconds and then the cursor moves to option 1.

Figure 4.2 SETUP Menu



The integration of the LIMS modules led to the reduction of program files which were duplicated in the different modules before. Accordingly, the enhanced LIMS will have 48 program files instead of 61 in the current version. The two header files in the current version are also kept.

Due to the integration of the LIMS modules, the give-away LIMS system will have only one executable file (LIMS.EXE) instead of five in the currently used version. One additional memory file *CLRNORM.MEM* has also been added to store the default color mode values used in LIMS version 2.0. Table 4.1 lists those give-away program and resource files.

Table 4.1 LIMS enhanced version, give-away files

File Name	*Purpose/Content*
LIMS.EXE	An integrated LIMS executing file which is used to define a user data set, enter data, validate the data entered, and produce reports and extracts.
CFG_REF.MEM	System Configuration, dBASE standard memory variable file
FDL_REF.DBF	File Definition Library, dBASE standard database file
DDL_REF.DBF	Data Definition Library, dBASE standard database file
CDL_REF.DBF	Code Definition Library, dBASE standard database file
BDL_REF.DBF	Breed Definition Library, dBASE standard database file
CLRHERC.MEM	Hercules mode defaults, dBASE standard memory variable.
CLRMONO.MEM	Monochrome mode defaults, dBASE standard memory variable.
CLRLCD.MEM	LCD mode defaults, dBASE standard memory variable.
CLRNORM.MEM	Color mode defaults (LIMS default values), dBASE standard memory variable

4.2 SEMAPHORE PROTECTION

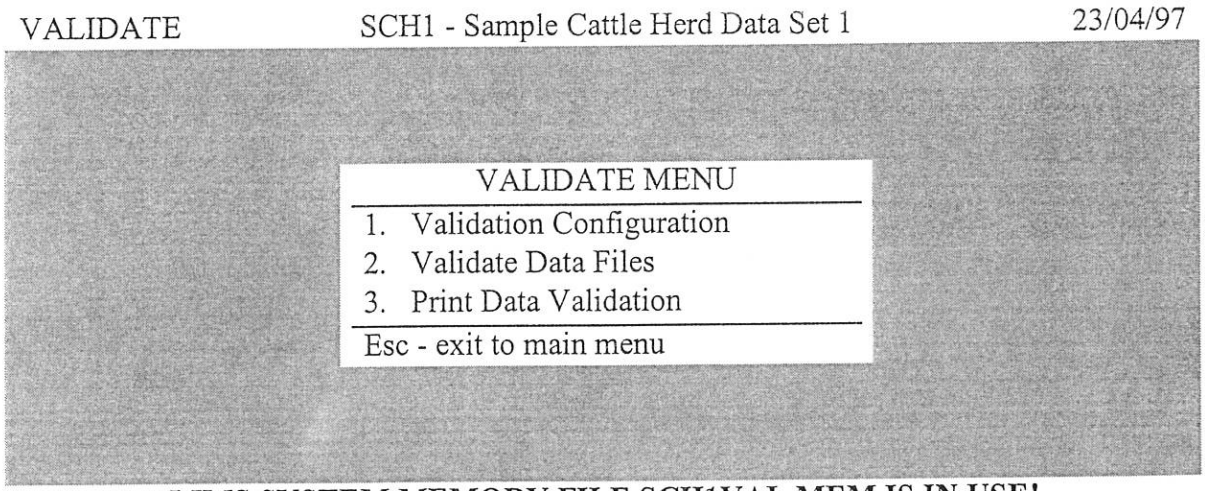
Clipper provides locking protection only for .DBF and .NTX file formats. It, therefore, does not protect memory (.MEM) files which are used to store configuration values in LIMS. In order to protect these memory files from corruption due to uncontrolled multiple access, a semaphore locking strategy is adopted.

A dBASE standard database file is added to the generic LIMS data set. This file is created when a user data set is created. The file is used to record the name of the memory file and a flag indicating whether the memory file is being used for writing or not. Whenever a user

wants to open a memory file for writing, the system will check for the in-use flag in the semaphore database file.

If the memory file is being used by another user, that is the in-use flag is set, the system tries to gain exclusive access to the file for 30 seconds. If control is not gained during these trials the message **"LIMS SYSTEM MEMORY FILE *XXXXYYY*.MEM IS IN USE!"** will be displayed on the last line of the screen (where *XXXX* is the name of the user data set and *YYY* is the name of the memory file used by the particular module being accessed). Figure 4.3 presents the message as displayed in VALIDATE module when trying to access the validation configuration option.

Figure 4.3 Semaphore Locking Message - VALIDATE Module



If the memory file is not in use, an in-use flag will be set and the user may continue with the configuration process. Any other user will be denied access to the memory file for writing until the first user is done.

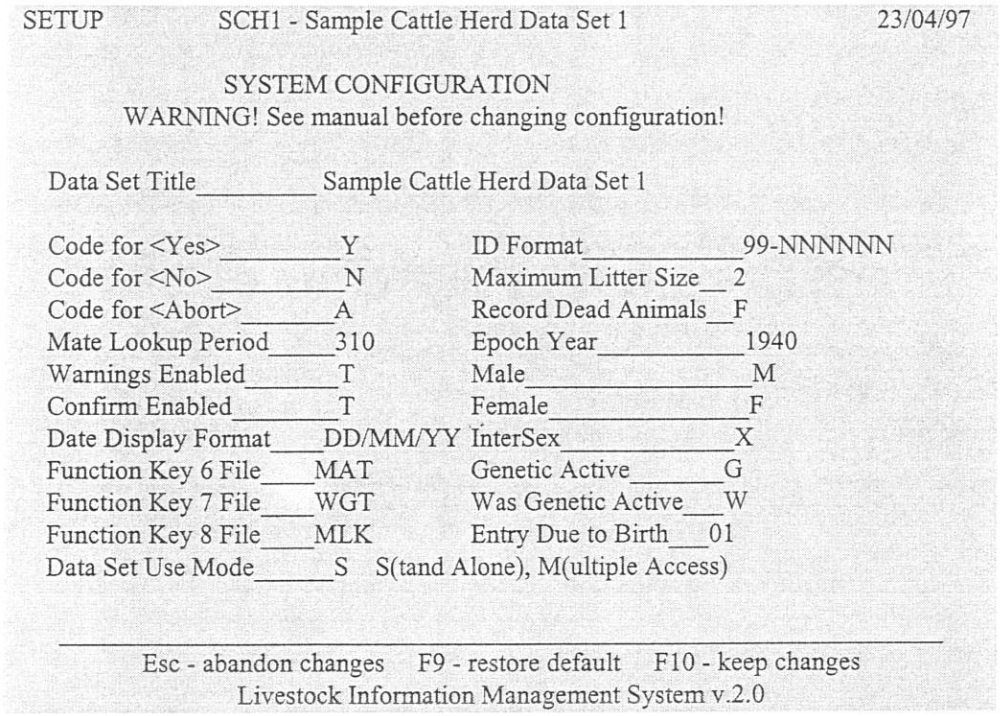
Upon completion of any modifications of the configuration values, the configuration values would be written to the memory file. The in-use flag set for that memory file will then be released and the memory file will be available for other users.

This protection mechanism is available to the program as an internal facility. The only time the users encounters this facility is when the memory file is in-use by another user and the error message "**LIMS SYSTEM MEMORY FILE *XXXXYYY*.MEM IS IN USE!**" is displayed as explained above.

4.3 PASSWORD PROTECTION

In order to provide access protection for some of the LIMS modules, password security has been implemented in the *System Configuration* option of the SETUP module. When a user data set is created for the first time, using the *System Configuration* option, the user will be prompted to choose whether the data set is to be used as stand alone (only one user environment) or to be shared between multiple users (network environment). Figure 4.4 presents the *System Configuration* screen.

Figure 4.4 Date Set Use Mode - System Configuration Screen



User's input to the prompt **Data Set Use Mode------S S(tand alone), M(ultiple Access** will determine as to whether the data set is to be accessed by a single user or multiple users. If the value of "S" is set, the data set will be accessed only by one user at a time. On the other hand, the value of "M" would mean the data set is to be accessed by multiple users.

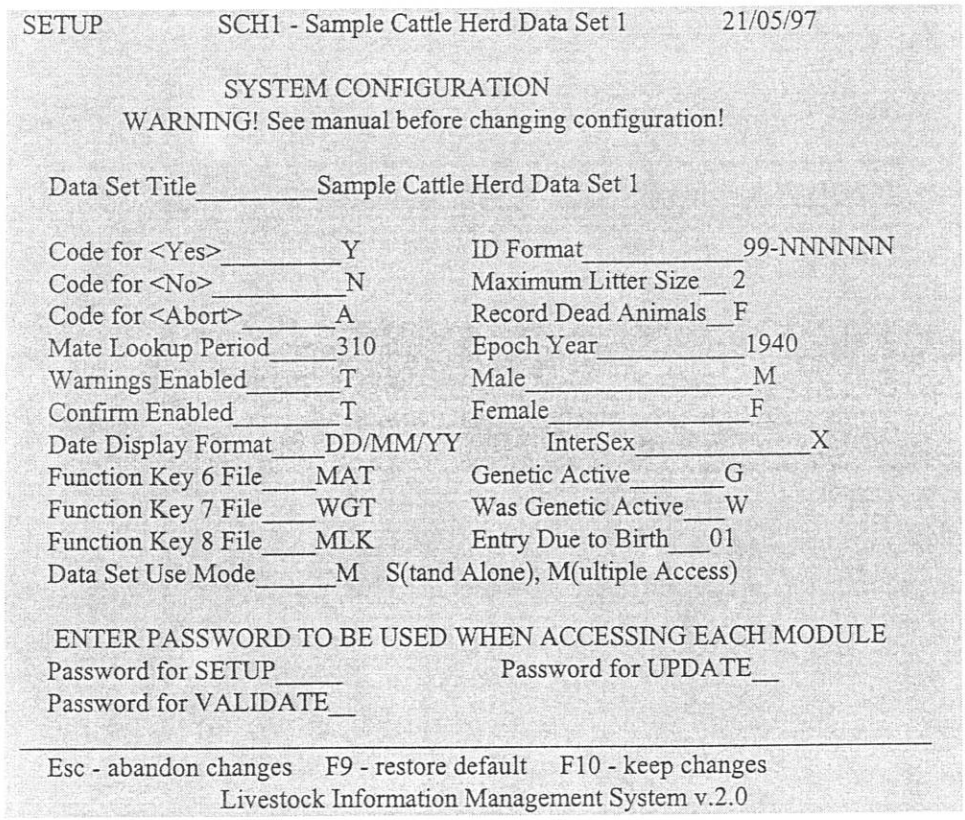
If a data set is defined as a stand alone, it will be used only by one user at a time. A data set defined as a stand alone could be changed to a multi-user data whenever required. This is done by accessing the *System Configuration* screen in SETUP and changing the value for *Data Set Use Mode* from *"S"* to *"M"*. It is also possible to change the *Data Set Use Mode* from multiple access to stand alone mode by reversing the values.

If the user inputs "M" for the prompt **Data Set Use Mode------S S(tand alone), M(ultiple Access)**, three more lines are displayed at the bottom of the screen shown above. In such

cases, the users is required to enter the passwords to be used for Setup, Update and Validate modules. The new screen will look like the one in Figure 4.5.

The password values input here will be used to verify the password values entered by users when the options that require such password protection are invoked. The defined passwords could be altered by accessing the *System Configuration* screen in SETUP and re-entering the values for each password.

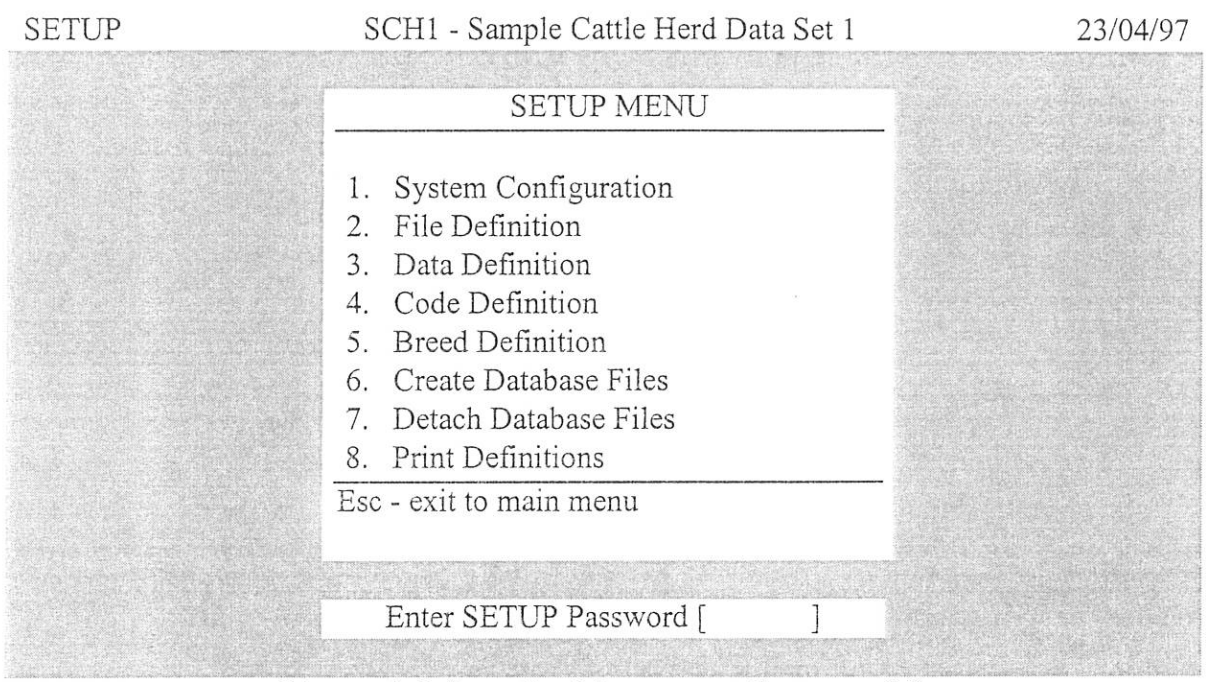
Figure 4.5 Password Definition on the System Configuration Screen



If multiple access of the data files is set as outlined above, the user will be asked to enter the password when he/she invokes the options that require password in each of the three modules where password protection is important. For instance, if the user selects *System*

Configuration option in SETUP after multiple access option is set, the prompt "*Enter SETUP Password [ ]*" is displayed at the bottom of the screen as shown in Figure 4.6.

Figure 4.6 Password Prompt - SETUP Menu



At this prompt, the password is accepted from the user. If the correct password is entered, the system configuration menu would be displayed and modifications are allowed - provided that no other user is making such modifications at that particular time. If the password given is incorrect, the following message is displayed at the bottom of the screen:

"ERROR - INCORRECT PASSWORD!"

and the user would be prompted to input the password for two more times. If the correct password is not given by the third trial, the system will go back to the setup menu.

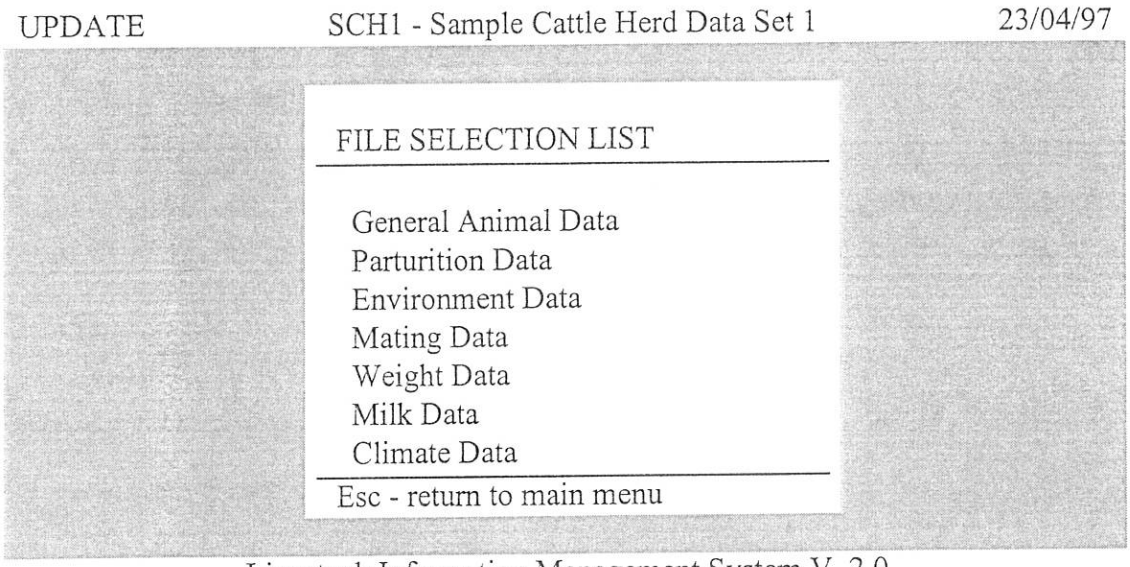
An eight character generic value is supplied as a separate key. This value is used to enter the *System Configuration* in SETUP menu to redefine passwords, in case a user forgets his/her password for the data set. The user should supply this generic value at the "*Enter SETUP Password [ ]*" prompt in such events.

4.4 OPENING DATABASE FILES

One basic issue in a multi-user environment is the simultaneous access of a data file by multiple users. There are situations which require exclusive access of a data file, though. The discussion here on the implementation of opening files in LIMS is both important when a database file is opened for shared and exclusive use.

Once a user invokes the "Update a Data File" option of the UPDATE module, a list of files will be displayed from which the users selects the file to which such alterations are made. Figure 4.7 presents the *File Selection List* as in the **Update a Data File** option in the UPDATE menu.

Figure 4.7 File Selection List in the UPDATE module



When the user selects the data file to be used during the current session, LIMS will attempt to open the file. If the file open was not successful on the first trial, the system will make several attempts to open the specified file for 30 seconds, every second. If the open file is successful the user will be prompted for next action and the process will continue. On the other hand, if the open file was not successful (because the file was exclusively used by another program, for instance), the following error message is displayed at the end of the screen:

NET ERROR OPENING FILE cFileName!

where cFileName is the name of the current database file. The user will be forced to abandon current operation and return control returns to the calling function or menu.

4.5 RECORD AND FILE LOCKING

After a database file is opened as explained in section 4.4 the next step, in a multi-user environment, is to make sure that data could be appended to the file in a safe manner. This safety measure is implemented in LIMS through the provisions of record and file locking. Since LIMS adds or updates a single record at a time, record lock is used mostly. File locking is used only in few cases where the whole file is affected by the process such as when creating index files.

The difference between record and file locking is that of scope. The scope of a record lock function is limited to a single record in a database file. On the other hand, the file lock function affects the whole database file. The explanation given below for record locking is true for file locking as well except for the difference in scope as been discussed above.

When a user inputs the key fields which would access the specified record in a database file, the system first locates the record. When the record is located, an attempt will be made to lock that particular record. If the first locking attempt fails, the system makes several locking attempts for 30 seconds with the interval of half-second. If the lock was not successful at the end of the specified trail duration, the following error message is displayed at the bottom of the screen:

NET ERROR LOCKING THE RECORD IN cFileName

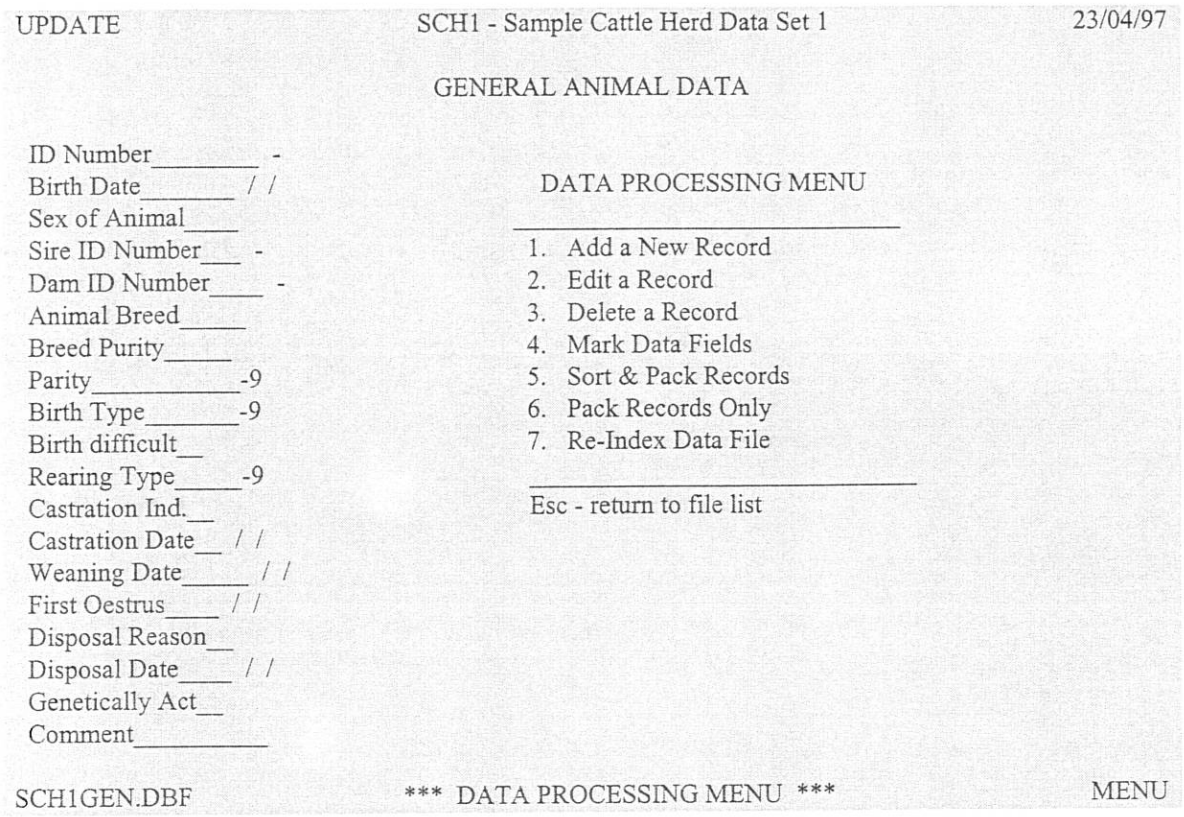
where cFileName is the name of the current database file. The current operation fails and control returns to the calling menu.

If the record lock was successful then the user would be allowed to continue the intended operation.

4.6 ALTERING DATA RECORDS

After a database file is opened as explained in section 4.4 and the necessary security measure is taken as explained in section 4.5, the next step would be making changes to it. Three main changes to a database file are discussed in this section: adding new records, updating existing records, and deleting existing records. These issues will be demonstrated here as it appears in the *Update a Data File* option of the UPDATE module. Figure 4.8 presents the *Data Processing Menu* of the *Update a Data File* option (it is assumed that the General Animal Data file is selected during the database opening).

Figure 4.8 Data Processing Menu - UPDATE Module



If option 1 "Add New Record" of the *Data Processing Menu* is selected, the system makes an attempt to add a blank record to the database file. If the addition of a blank record is not successful by the first trial, several attempts will be made for 30 seconds by the interval of half-second. If all the attempts fail, the operation of adding new record fails and the following error message is displayed at the bottom of the screen.

NET ERROR ADDING NEW RECORD TO cFileName

where cFileName is the name of the current database file. The current operation is abandoned and control returns to the Data Processing Menu.

If the addition of a blank record is successful, the system locks the newly added blank record and allows for data entry.

If option 2 "Edit a Record" or option 3 "Delete a Record" is selected from the Data Processing Menu, the user is prompted to enter the key fields which would help in locating the record. Once the record is located, the system tries to lock the record as per the locking routine discussed in section 4.5. If the lock is successful, the system accepts users changes.

4.7 UPDATE VISIBILITY

When modifications (addition of new records, update or deletion of existing records) are made to database files, only the memory copy of the records are updated immediately. Unless such changes are explicitly written to the shared device, the modifications will not visible to other users on the shared device under a multi-user environment.

The Clipper commit command which calls the DOS commit file function for writing the changes to the shared device has been implemented in LIMS. The DOS commit file function will flush the changes from the memory buffer to the shared device. The call to the function is made immediately after the changes are made to the data records and before the current lock is released. Since the current lock is released just after the commit function, changes made by the current user will be made available to the rest of the users upon the release of the lock.

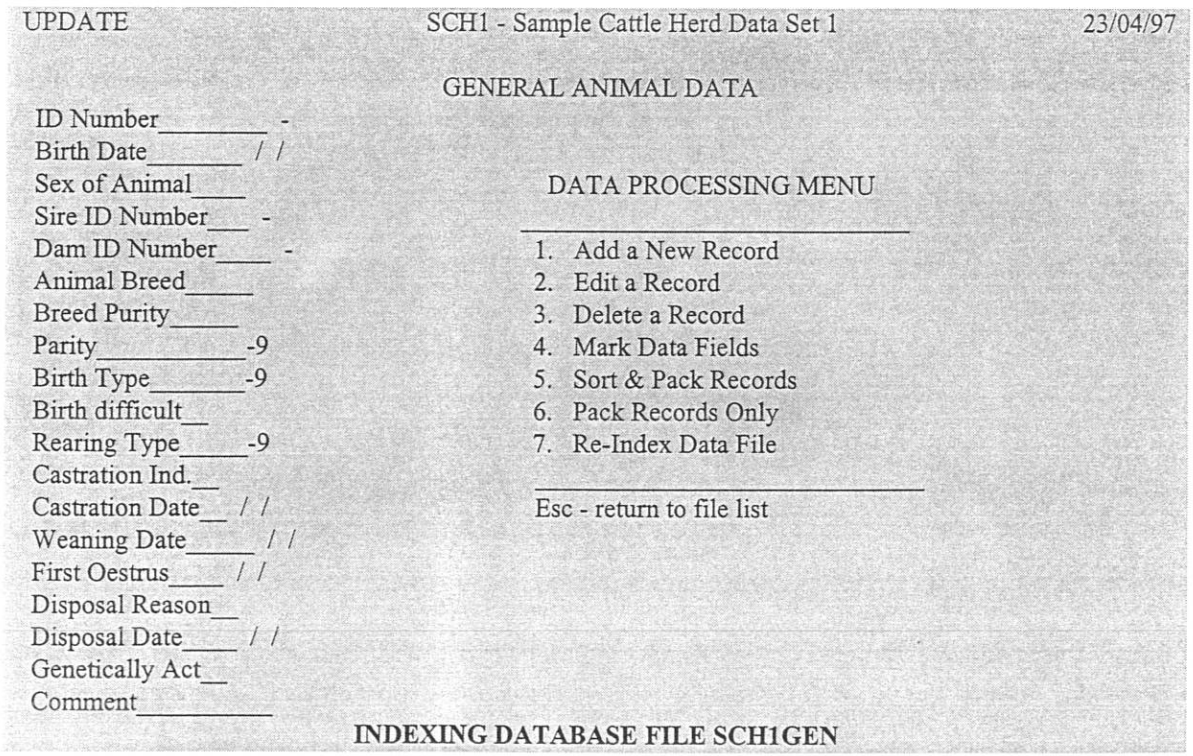
4.8 INDEXING DATABASE FILES

Once records are added to the database file, the need to place a logical ordering to the records arises. Such logical ordering is placed on the data files so that retrieving records from the data file becomes an easy task.

The LIMS software allows for the creation of index files either for a single database file through an option in the *Data Processing Menu* of the *Update a Data File* option or all database files through the *Re-Index All Files* option of the UPDATE menu.

When a user invokes the re-index data file process, the system tries to open the specified database file in shared mode as described in the section 4.4, Opening Database Files. If the open file is successful, the system then will attempt to place a file lock as per the procedure for file locking. If the file lock is successful, the system identifies the index key fields defined by the user during the SETUP procedure and creates the index file(s). When the indexing taking place, the message "**INDEXING DATABASE FILE cFileName**" is displayed on the last line of the screen as been presented in Figure 4.9 (where cFileName is the name of the database file being indexed).

Figure 4.9 Indexing Database File - Data Processing Menu



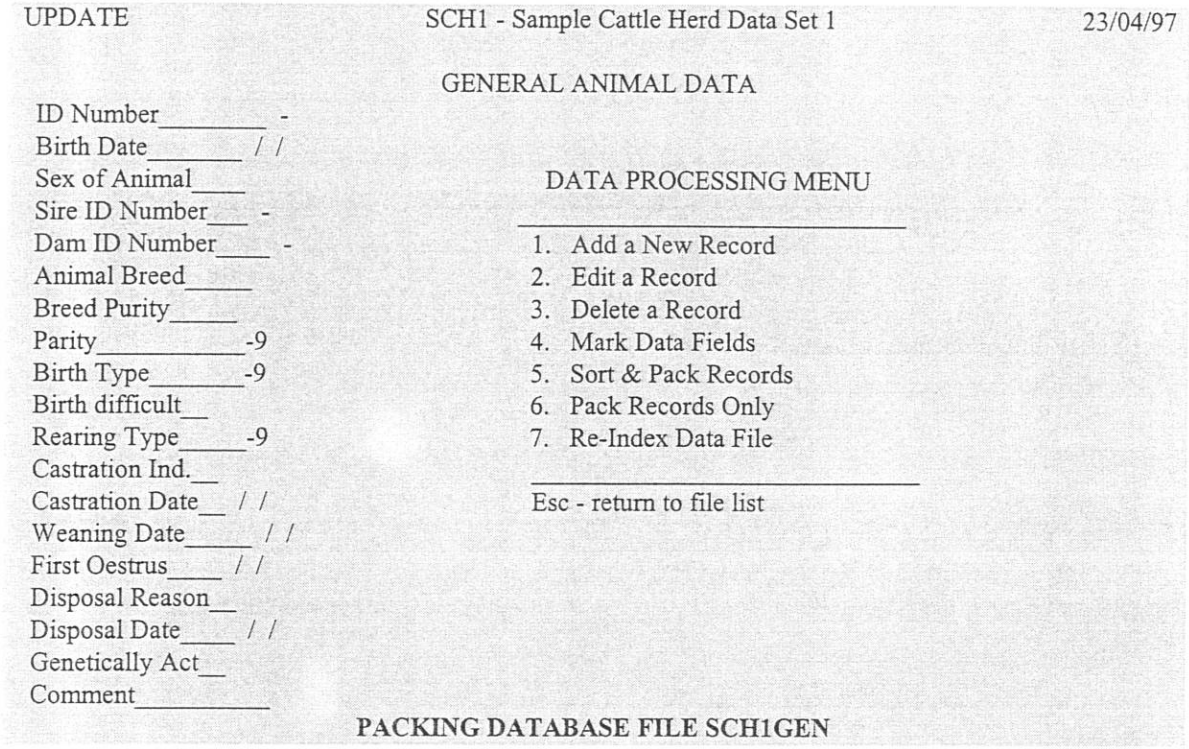
In LIMS all index files are built on a fixed length key fields. After the index operation, control returns to the calling menu.

4.9 REMOVING DELETED RECORDS

At times permanent removal of deleted records might be required. There are two two-set options provided in LIMS for such purpose. One set is available on the UPDATE menu whose scope is for all database files defined in the user data set. The other set is available through the *Update a Data File* option and its scope is restricted to the currently open database file.
One option allows only the removal of deleted records from database file(s) while the other allows for the removal of deleted records and the sorting of the records.

When a user invokes the *Pack Records Only* option, the system tries to open the specified database file in exclusive mode as described in the Opening Database Files section. If the exclusive file open is successful, the system re-creates the index files then displays the message "PACKING DATABASE FILE cFileName" (where cFileName is the name of the database file in use) as shown in Figure 4.9 and removes deleted records from the database file. After the deleted records are removed the system re-creates the index files and control returns to the calling menu.

Figure 4.10 Packing Database File - Data Processing Menu



If the option *Sort & Pack Records* is invoked by the user, the system first re-creates the index files. Then it will remove deleted records from the database file and sorts the remaining records. After that, it will re-create the index files and control returns to the calling menu. The message displayed while these actions are taking place would be "**PACKING & SORTING DATABASE FILE cFileName**".

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The main objective of this study was to examine the considerations in network programming and to demonstrate these concerns by enhancing Livestock Information Management System (LIMS). Accordingly, a modest attempt was made to reveal the issues that are of most concern in multi-user environment by studying available literature. The LIMS software was also examined so as to elicit the necessary information relevant to the study and to draw pertinent design in enhancing the software.

Based on the knowledge gained from literature and the examination of LIMS (by running the system, reviewing the documentation and studying the source code), LIMS was enhanced for multi-user environment. Since the LIMS software is a huge software with several pages of source code, only relevant functions which were added and or modified to implement the relevant issues raised are presented in the report. The following are the major achievements of this study:

a. The five LIMS modules have been integrated and a single user interface menu, with a set of options to run the modules, has been provided.
b. Password protection has been introduced and implemented. The password protections are used for modules which require central management and/or special protection due to the scope of the changes that might be introduced by their access.

c. LIMS has been enhanced so that its data files are simultaneously accessed for data entry and reading the contents to produce reports, while ensuring data integrity and consistency.

d. Shared data access has been implemented for data entry, data validation, report production and data extraction. Exclusive opening has been set only where exclusive access is required by the process such as removing deleted records from files.

e. Record locking has been incorporated for the addition, update or deletion of data in LIMS data files. File locking was used only where the process affects many records such as creating index files.

f. Since Clipper does not support locking of dBASE standard memory files, semaphore locking was developed using the available provisions of this programming language.

g. A time period concept was introduced for functions which open, add, update or lock a record/file in LIMS so that several attempts are made before reporting back success or failure.

5.2 Recommendations

There are many reported and implied weaknesses of LIMS that need to be improved. Due to its scope and the resources available for it, the current study dealt only with integrating LIMS modules and enhancing the software for network. Besides, due mainly to time constraint, the designed system is not fully implemented for two of the modules, namely the REPORT and

EXTRACT modules. Based on the achievements of the current study and the need for further facilities in LIMS, the following recommendations are drawn:

a. Using the lesson learned in this work, it is recommended that a top-down-approach should be considered for the development of a fully fledged multi-user LIMS. A network architecture has to be defined. User groups have to be identified and the different types of data sets have to be categorized. This should lend to a proper hierarchical directory structure within which access rights are controlled.

b. Also, in developing the fully fledged LIMS the involvement of users (from specification to evaluation) is highly recommended.

c. The current LIMS version doesn't have an on-line help facility which could guide users in setting up the system and using it for livestock information management. Since such a facility will enhance the user friendliness and usability of the system, its inclusion is recommended.

d. The addition of system utilities, such as the possibilities of converting and appending non-LIMS data files into LIMS data set, printer selection, and appending data from another LIMS data set files, is of paramount importance for the successful usage of the system. It is thus recommended that such utilities are incorporated into the system.

The recommendations presented above, obviously, require the commitment of financial and manpower resources. Thus, it is further recommended that, those directly responsible for the production and distribution of the LIMS software assign required manpower and financial resources for the implementation of the recommendations.

BIBLIOGRAPHY

1. Akthar, S. and Melese, M. 1994. Africa, information and development: IDRC's experience. Journal of Information Science 20 (5): 314-322.
2. Avison, D. E. and Fitzgerald, G. 1993. Information systems development: methodologies, techniques and tools. Oxford: Alfred Waller Limited.
3. Blue, T. 1995. Preventing lunch-time locks. The Aquarium/Clipper August 1995.
4. Booth, J. D. and Lief, G. 1993. Network programming in CA-Clipper 5.2. Emeryville: Ziff-Davis Press.
5. Booth, J. D., Lief, G., and Yellick, C. 1992. Clipper 5: a developer's guide. San Mateo: M&T Books.
6. Computer Associates, Inc. 1992a. CA-Clipper: Reference guide Vol. 1 Version 5.2. Computer Associates International, Inc.
7. Computer Associates, Inc. 1992b. CA-Clipper: Reference guide Vol. 2 Version 5.2. Computer Associates International, Inc.
8. Computer Associates, Inc. 1992c. CA-Clipper: Programming and utilities guide version 5.2. Computer Associates International, Inc.
9. Duncan, R. 1988. Advanced MS-DOS programming. 2nd ed. Redmond: Microsoft Press.
10. Forgionne, Guisseppi A. 1991. Decision technology systems: A vehicle to consolidate decision making support. Information Processing & Management 27(6):679-697.
11. Goldschlager, L. 1982. Computer Science: A modern introduction. London: Prentice-Hall International, INC.
12. Haugdahl, J. S. 1987. The DOS-LAN juncture. PC Tech Journal July 1987: 78-90
13. Heimendinger, L. 1988. The four commandments bullet-proofing your LAN applications. DBMS. December 1988: 42-54.

14. Herman, G. and Hess, D. A. 1996. Applications programming interfaces. Datapro. Delran: The McGraw-Hill Companies, Inc.

15. IBM. 1987. DOS technical reference. Glasgow: Collins.

16. ILCA. 1992. Livestock Information Management System: System Documentation. Addis Ababa: International Livestock Centre for Africa.

17. James, M. 1989. Low cost networking. Oxford: Heinemann Professional Publishing.

18. Martin, W. J. 1988. The information society. London: Aslib.

19. Masiga, W. N. 1995. Livestock research requirements in sub- Saharan Africa. In Global agenda for livestock research: proceedings of a consultation held at Nairobi, 18-20 January 1995, edited by Gardiner P and Devendra C, 77-80. Nairobi: International Livestock Research Institute.

20. Milligan, C. 1990. NetWare 386 user's guide. Redwood City: M&T Books.

21. Morgan, D. 1988a. Open Modes: behind the-scenes file considerations. Nantucket News 3(2): 1-5

22. Morgan, D. 1988b. Clipper mirrors DOS file capabilities: Low level file functions. Nantucket News 3(3): 14-15

23. Morgan, D. 1989. One if by land and two if by sea: semaphores in multi-user environments. Nantucket News 4*(2)*: 7-11

24. National Science and Technology Information and Documentation Centre. undated (a). The of information in national development. unpublished.

25. National Science and Technology Information and Documentation Centre. undated (b). Information and information science. Unpublished.

26. O'Brien, James A. 1993. Management information systems: A managerial end user perspective. Boston: IRWIN

27. Rose, Marshall T. 1990. The open book: A practical perspective on OSI. New Jersey: Prentice Hall.

28. Sansoucy, R., Jabber, M. A., Ehui, S. and Fitzhugh, H. 1995. Keynote paper: The contribution of livestock in food security and sustainable development. In Livestock development strategies for low income countries: Proceedings of the joint FAO/ILRI roundtable for low income countries held at Addis Ababa, Ethiopia, 27 February - 02 March 1995, edited by Wilson R T, Ehui S and Mack S, 9-21. Addis Ababa: Food and Agriculture Organization / International Livestock Research Institute.

29. Seré, C., Steinfeld H. and Groenwold J. 1995. World livestock Production systems: Current status, issues and trends. In Global agenda for livestock research: proceedings of a consultation held at Nairobi, 18-20 January 1995, edited by Gardiner P and Devendra C, 11-40. Nairobi: International Livestock Research Institute.

30. Slone, J. P. 1997. Handbook of local area networks. 3rd ed. Boston: RIA Group.

31. Spence, R. 1989. Clipper Programming Guide. San Marcos: Microtrend Books.

32. Suomi, Reima. 1994. What to take into account when building an inter-organizational information system. Information Processing & Management 30(1): 151-159.

33. Tanenbaum, A. S. 1989. Computer Networks. 2nd ed. Englewood Cliffs: Prentice-Hall International, Inc.

34. Tischer, M. 1992. PC intern system programming: The encyclopedia of DOS programming know-how. Grand Rapids: Abacus

35. Weber, E. 1990. Advanced multi-user programming techniques. Nantucket News 5(3): 3-11

ANNEX I - COMMENTS ON LIMS SOFTWARE

This annex presents comments by two major groups, the department that developed LIMS and the major user group at ILRI. This presentation is intended to show that there is a need to enhance LIMS in many aspects. As has been indicated in the objectives of the study, the current study deals with the integration of the various LIMS modules and enhancement of the same for use in network environment.

1. The LIMS software was developed by the Computer & Biometrics Section of the International Livestock Research Institute (ILRI). Discussion with the Head of the Section regarding LIMS has revealed the following main points:
    - LIMS was basically designed for long-term herd monitoring and management. Many comments have been received by the section that LIMS is good for this purpose.
    - LIMS is being distributed to ACP countries free of charge. It is expected that it will be distributed free to Asian, Latin American and African countries as ILRI has a global mandate to work with these countries.
    - LIMS was developed for national agricultural research institutions of ACP countries. As these institutions had no adequate computer hardware (specially disk space) LIMS was provided in five separate stand-alone modules. LIMS was not considered for a network environment when its development was under-way since the institutes in the ACP countries had no network at that time.
    - The major commendable point received from users is about the good documentation the system provides for archiving purposes. On the other hand, the major complaint is that the software is not suitable for short-term designed experiments.
    - There is a need for simultaneous access of LIMS data files. Different programs / departments would certainly like to share the same database.
    - Enhancing LIMS for network is a worthwhile endeavor. Local area networks are emerging in the national agricultural research systems these days. The enhancement of LIMS for network is, thus, a good step forward in meeting the needs of these research systems.

2. The major user of LIMS in ILRI is the Animal Genetics Resources Division. This group has about 200,000 records in one of its LIMS data set. Discussion with the personnel of this department has pinpointed the following issues:
    - LIMS is being used for herd monitoring and management.
    - The most important aspect of LIMS is that it guards against erroneous data entry since it checks entered data against user defined configuration values. Errors committed during data entry are brought to the attention of the data entry personnel when validation process is conducted.
    - The most important drawback of LIMS is that it has no data sharing facility. Thus, it does not allow for simultaneous data entry or report production. As a result data is being shared between different users through the exchange of disks.
    - There is a need to access LIMS database files both for simultaneous data entry and for reading to produce reports and data extracts.

3. The following excerpt is taken from "Comments on LIMS Software" Ref. ABPS/145/93, dated 08/03/93 - made by a Senior Animal Breeder at ILCA/ILRI.

    "... Transfer of data into LIMS is clearly very time-consuming. It requires developing separate programs for different data sets. That is, although the requirement of LIMS data set is always the same, one needs to write different programs because data come in different formats.

    ... The other limitation of LIMS is that no more than one person can work on the data entry at the same time. That is, the system is stand-alone. If data has to be entered urgently and different data entry personnel are available, it is impossible to utilise them at the same time. In order to solve this problem, it is very important to upgrade LIMS. Such upgrades should include development of a version that can operate in a network environment.

DECLARATION

The thesis is my original work and has not been presented for a degree in any other university.

Hambisa Belina Disassa

May 17, 1997

The thesis has been submitted for examination with my approval as a university advisor.

Dr. Gbade Alabi

May 17, 1997