NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A MAGNETIC TAPE LIBRARY SYSTEM
FOR THE COMPUTER SCIENCE DEPARTMENT NPGS;
REQUIREMENTS ANALYSIS, DESIGN, AND IMPLEMENTATION

by

Billie Elizabeth Crawford

December 1985

Thesis Co-Advisors: Daniel Dolk
Barry Frew

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (*and Subtitle*) A Magnetic Tape Library System for the Computer Science Department NPGS; Requirements Analysis, Design, and Implementation		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis December 1985
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(*s*) Billie Elizabeth Crawford		8. CONTRACT OR GRANT NUMBER(*s*)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943-5100		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943-5100		12. REPORT DATE December 1985
		13. NUMBER OF PAGES 201
14. MONITORING AGENCY NAME & ADDRESS(*if different from Controlling Office*)		15. SECURITY CLASS. (*of this report*) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (*of this Report*)

Approved for public release; distribution is unlimited

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20, if different from Report*)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (*Continue on reverse side if necessary and identify by block number*)

Ingres
VM UNIX
database
tape library
data retrieval

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

This thesis describes the development of a magnetic tape library system (TLS) for the Computer Science Department of the Naval Postgraduate School, Monterey, California. The TLS was developed using the Ingres VM UNIX Version 2.1/15 VE.04 database management system on the 4.2 BSD UNIX operating system. The thesis includes a requirements analysis and design, and a description of implementation.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S N 0102- LF- 014- 6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

A Magnetic Tape Library System for the Computer Science Department NPGS; Requirements Analysis, Design, and Implementation

Billie Elizabeth Crawford
Lieutenant Commander, United States Navy
B.A., University of Maryland, 1971

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL

December 1985

ABSTRACT

This thesis describes the development of a magnetic tape library system (TLS) for the Computer Science Department of the Naval Postgraduate School, Monterey, California. The TLS was developed using the Ingres VM UNIX Version 2.1/15 VE.04 database management system on the 4.2 BSD UNIX operating system. The thesis includes a requirements analysis and design, and a description of implementation.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

Documentation is incomplete without my acknowledgement of the hours of work and patience contributed to this thesis by a few generous people. My thesis advisors Barry Frew and Dan Dolk were always available to share their advise, experience, and reassurance. They kept me going when I believed the end would never come. Sue Whalen and Al Wong of the Computer Science Department were forever patient and instructive during the many hours I spent in their offices becoming familiar with the library, the UNIX operating system, Ingres, and "C". The suggestions, patience, and hours of creativity of Nita Raichart, my typist and thesis "molder", deserve special thanks, especially as I compare the final document to the original roughs from which she worked. And most of all I thank my mother for the hours of proofreading and motherly support she provided during her most unusual visit to California.

I. INTRODUCTION

The Computer Science Department maintains a variety of magnetic tapes for their instruction laboratory. The current combination manual/computerized library system is inefficient. The department wants a computer-based system by which librarians may add, locate, update, and provide reports regarding tapes in the library. There are currently six librarians who are staff members of the department. In addition to their librarian duties, they are responsible for some form of maintenance of the department's two VAX 11/780 and one VAX 11/750 computers. The maintenance administration is performed with the VAX 11/780/UNIX operating system. The department wants a tape library to be developed using the Ingres (VM UNIX Version 2.1/15VE.04) database management system (DBMS), the 4.2 BSD UNIX operating system.

The database development process will be used to build a tape library system (TLS). It will be developed in four stages -- requirements specification, evaluation, design, and implementation.

A. REQUIREMENTS SPECIFICATION

The management of the Computer Science Department has already defined the need, and estimated the cost/benefit of the project. They approved its continuation when they requested that the library be developed by a thesis student.

The current tape library is a combination of a casual computer listing of approximately 600 tapes, and opened cabinets in which tapes are segregated by type.

The library system must facilitate use by the six librarians. The need to store and catalog tapes dependably is critical. It must ensure the security of the data and tapes. Guidance for logging data must be well defined and interactively prompted to ensure input of all necessary information.

Development of the library by a thesis student for use on the VAX 11/780/UNIX operating system and Ingres database will keep costs to a minimum. No new hardware, software, or personnel purchases will be required. Because all the librarians have extensive experience with the system, minimum training will be required. This thesis describes development of the TLS.

Detailed user requirements will be determined by extensive interviews with the users. The department has assigned a user team leader. The current system and desired changes will be documented in Chapter II by a narrative and data flow diagrams. The proposed system will be documented by data flow diagrams, process mini-specifications, data items, and a specification data dictionary. The new system will be reviewed and approved by the users before the evaluation stage commences.

B. EVALUATION

The evaluation stage will be limited because the department has already delineated the system hardware, operating system, application language, data technology, and people. The database design, application program, and procedural alternatives will be developed for evaluation and selection by the user. A project plan, outlining major development tasks, will be documented for use by the developer and user in following project development.

C. DESIGN

The design stage will include definition of the database structure application program, security and control, and functions of the database administrator.

In Chapter III, the database structure will first be defined logically, independent of the requirements of Ingres, according to the rationale of the data record relationships and fields. These relationships will be illustrated by data structure diagrams (Bachman diagrams). They will be defined in a logical schema.

In Chapter IV, the logical structure will be mapped to the physical structure of Ingres. The relations, data, formats, and the functions and interfaces of the modules will be included in the design data dictionary. The application program will be designed in parallel with the database design. Security and control procedures will be designed into the application program to ensure that only authorized users may access the library.

A tutorial will be developed in parallel with the application program and will be included as Appendix G. It will include operating procedures, security specifications, and the database administrator's responsibilities for maintenance of the library.

D. IMPLEMENTATION

The application will be coded in the Ingres DBMS language. A small representative database will be loaded into the database. All modules will be tested. Users will be trained. Full loading of the 600 tapes will be accomplished by the staff librarians. Instructions for implementation are provided in Appendix F. Final conclusions and recommendations will be included in Chapter V.

II. REQUIREMENTS ANALYSIS

A. NARRATIVE

1. Background

The Computer Science Department's current tape library is a combination of a casual computer listing of 600 tapes, and opened cabinets in which tapes are segregated by type. Tape labels contain as much information as possible in case the information in the library listing is inaccurate, incomplete, or nonexistent. The information in the listing is not standardized. It consists of whatever one of eight librarians considered necessary for identification or retrieval when (s)he entered the tape into the library. Frequently customers help themselves to shelved tapes, and new boxed tapes sometimes disappear. Few staff and student personnel request the library to store their tapes. The department would like to provide secure, dependable library services.

Compounding the problem, the computer facilities are rapidly expanding. Dependence on the facilities is increasing and people are requesting better library services. Concurrently, requests for backup tapes are increasing as is the length of time between when a backup is created and retrieved. More tapes are getting lost. The need to store and catalog tapes dependably is critical.

2. Users

Approximately six to seven staff personnel, all extensively familiar with computers, perform librarian duties and will have access to the library and tapes. Because of the large number of librarians, guidance for logging data must be well-defined and interactively prompted to ensure that all necessary information is input.

3. Storage

The tapes will be stored in locked cabinets with separate slots for each tape. The staff would prefer not to physically separate the tapes by category and not to have descriptive information on the tape labels. This will help to prevent customers from helping themselves to tapes they need if cabinets are inadvertently left unlocked. Only the librarians should have access to the tapes.

4. Tape Entry

Tapes are received into the library in three ways: 1) boxes of new tapes; 2) tapes from vendors, and 3) written tapes requested for safekeeping by private owners. To prevent pilferage, it is desired that new tapes be identified and placed into the library immediately. Physical information about these blank tapes must be included. They will eventually be used to write backups. Vendor tapes are identified and catalogued as soon as possible. They are never retired for use as backups. Tapes input for safekeeping must be identified. If abandoned by the user, they may be used for backups. Essentially, the moment a tape is received by the library, it should be identified with as little information as possible on the label. It should be labelled with an identification number only, which can be used to locate assigned space in the cabinet and its descriptive information in the library.

5. <u>Backups</u>

It is the function of the tape librarians to perform backups. They do routine system backups, and "individual" backups by special request for staff, faculty, or student personnel. For individual requests, the owner's tapes maintained by the library may be reviewed to determine if there is sufficient space available on one to read the disk files requested to the partially used tape. To perform an individual backup when an appropriate, partially used tape cannot be found, or to perform any other backup, the librarian must first find a "scratch" tape, either new or previously used. It must be of sufficient condition, density capacity, and length to accommodate the backup. After the information on backup tapes is retired, the librarian makes the tape available for rewrite.

a. Routine Backup

Daily disk backups are written to tape each week. Weekly, disks are written to tapes. Full backups of the system are written to tape monthly. Information kept in the library must enable the librarian to access the tapes and write them back to disk. They must be identified by the date and time of backup, the system, utility, software used to generate the tape, and path information such as partition or disk, or source. Each daily backup is 1-2 tapes long, weeklys require 2-3 tapes, and monthly's 4-6 tapes. The library will save dailies for one month, weeklys for two months, and monthlys for six months.

When a Computer Science class graduates (14-17 students) the librarian creates a tape backup of graduated student disk files for each system. Each is 1-2 tapes long. Again, the librarian must find a tape of the appropriate length, condition, and density capacity. After writing the tape, the librarian must catalogue information which will enable retrieval of a student's files to write them to disk or another tape. This information must include date, student name, the specific files, the operating system hardware and utility used to generate, and device. The tapes will be kept for two years.

DATA RETENTION

DAILY BACKUPS	1 MONTH
WEEKLY BACKUPS	2 MONTHS
MONTHLY BACKUPS	3 MONTHS
GRADUATION BACKUPS	2 YEARS
INDIVIDUAL BACKUPS	2 YEARS
VENDOR BACKUPS	5 YEARS

b. Nonroutine Backups (Individual Tapes)

Staff, faculty, or students frequently request that a disk file be written to tape. The librarian first checks to see if the individual currently owns a tape in the library of the same system, copied with the same utility and device as required to access the file(s), that has a sufficient unused portion to write the file(s) requested. If an appropriate tape is found and used, the addition of the new file(s) must be recorded into the library. If there are none, the librarian must look for an adequate scratch tape. If a new tape was used because the owner had no previous tape(s) on file, the librarian identifies it as a new set with a new sequential order. A set of individual tapes can include up to three volumes. After copying the tape, the librarian catalogs the information necessary to find which tape the new file is on and how to write it to disk or another tape.

6. <u>Administrative</u>

a. Condition of Tapes

The librarian is responsible for monitoring the condition of the tapes and to recopy or throw out tapes as required. Condition is determined in one of two ways: errors and bit age.

(1) <u>Errors</u>

Errors are unreadable bit spaces on the tape and are communicated to the librarian by the utility each time the tape is read or written. Ten to twelve errors (after head cleaning) is considered maximum life. The librarian must catalogue this information each time the physical tape is read or written. Physical tapes exceeding their maximum cycle or error life are permanently deleted from the library and thrown away.

(2) Bit Age

Bit age has to do with the information written on the tape and applies only to active tapes (vendor or backups). It is the age of the oldest bit on the tape. Bits deteriorate after five years so the tape information must be copied to another tape. The physical tape, however, may be reused. Checks for bit age must be done on all tapes on a periodic basis.

b. Retirement of Tapes

The librarian requires a flexible means of controlling the length of time all active tapes are retained by the library, other than by condition. The reasons could vary per type of active tape. When the retirement date occurs, a disposition decision will be made.

c. Lending of Tapes

Vendor, scratch, and individual tapes may be loaned. The library must maintain information regarding which tapes are on loan, and to whom.

7. Deleting Tapes

Backup tapes may be retired to scratch tapes, or any tape may be thrown away. For retiring backups, the librarian must provide a means to delete backup information but retain the physical tape information. It must allow deletion of all tape information and reflect the availability of a vacant labelled slot in the cabinet when a tape is thrown away.

B. DATA FLOW DIAGRAMS (DFD)

1. Current Library

An overview of the current TLS described in the narrative is illustrated by a DFD (Figure 1). Minispecifications and file descriptions follow the DFD.

2. Proposed Library

A general overview of proposed library procedures is reviewed in DFD(O) (Figure 2). All procedures are to be performed by a librarian. Subsequent DFDs (Figures 3-10) are more detailed. The procedures are numbered 1-9. The procedures and data input and output for each procedure varies according to the function the librarian performs. A DFD is written for each function to illustrate the specific procedures and data inputs and outputs required. Functions are assigned letters A-H. Mini-specifications of the procedures and a specification data dictionary follow each DFD. The procedure numbers of the functional DFDs match the procedure to the definition provided in the overview DFD(O) and are preceded by the appropriate function letter. For example: "A.2" should be read - Function A, Procedure 2.

For some functions, a procedure is broken down into sub-procedures. Sub-procedures retain the function and procedure number, followed by a decimal number (i.e., A.2.1 should read - Function A, Procedure 2, Sub-procedure 1).

Cabinets and the database will be the only files used to store library objects and information. Cabinets will consist of labelled slots, consecutively numbered left to right, top to bottom. The range of the numbering system will be expanded as tapes are added. Cabinets will be known by the range of numbers assigned. For example, a cabinet may be known as 100-300.

The library data will be the only source of information regarding labelled cabinet spaces, the physical condition of tapes assigned, and the data preserved on active tapes (vendor tapes and backup tapes). A backup tape is a tape with information which the library safeguards.

DATA FLOW DIAGRAM: CURRENT SYSTEM OVERVIEW

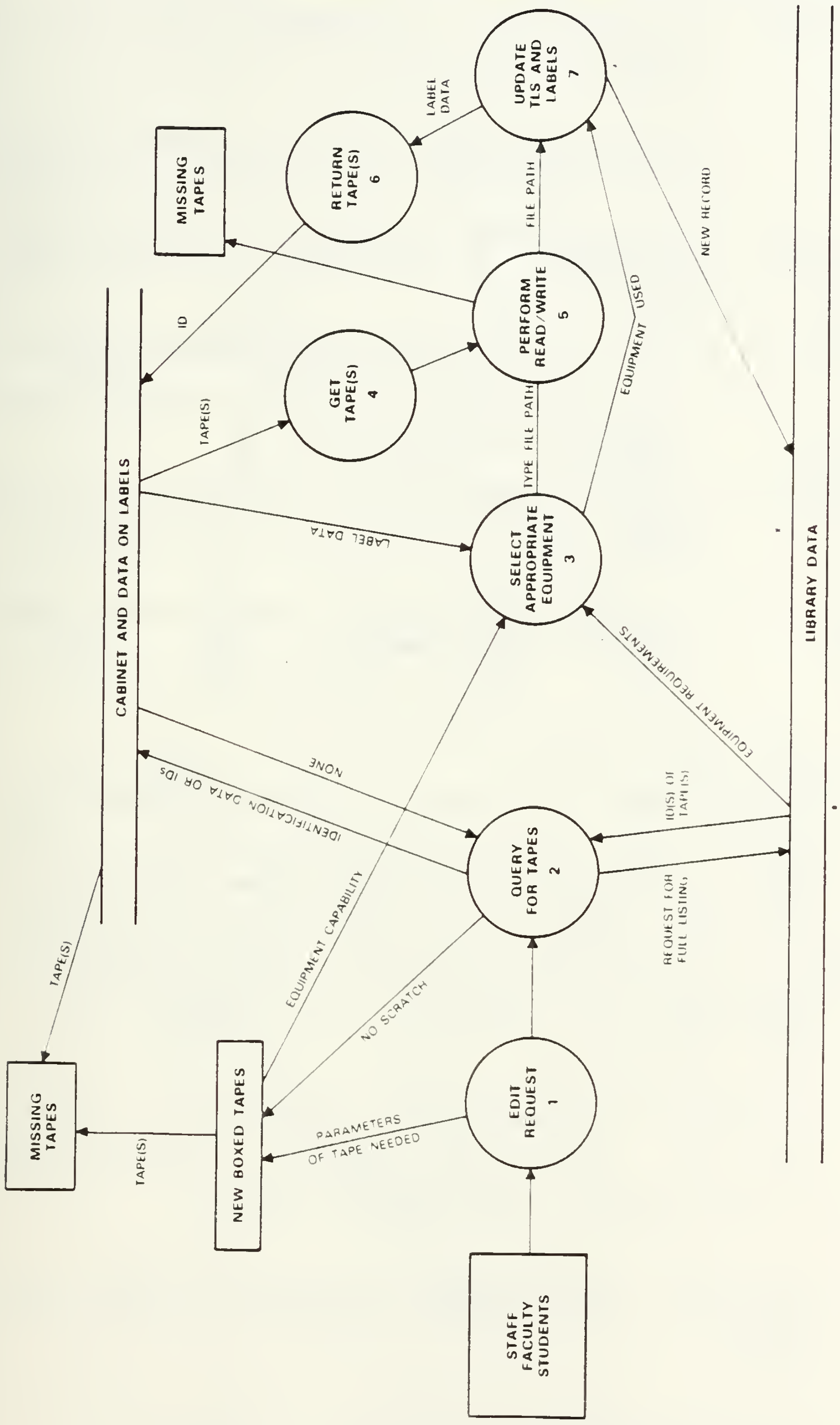


Figure 1. Current System Overview

MINI-SPECIFICATION: THE OLD SYSTEM LIBRARY - OVERVIEW

1.	Edit Request	The librarian or requestor edits the request at this time to determine what action to take.
2.	Query for Tape(s)	The requestor or librarian searches for the tapes which will satisfy the request.
3.	Select Appropriate Equipment	The librarian or requestor selects the equipment, if required, to satisfy the request.
4.	Get Tape(s)	The librarian gets appropriate tape(s).
5.	Perform Read/Write	The librarian or requestor performs the read/write, if required.
6.	Return Tape(s)	The librarian returns the tape(s) to the cabinet; the requestor might or might not.
7.	Update Database	The librarian updates the database; the requestor does not.

FILES

Four objects serve as files in the current library.

CABINETS	Cabinets have slots which hold the tapes. These slots are not numbered. The tapes which are contained in them have labels with numbers and tape information. An attempt is made to place reels of similar systems and tapes in specific sections of the cabinets. The cabinets are opened and easily accessible to anyone at NPS.
DATABASE	A vaguely defined computerized file of tape numbers with information for the librarian regarding the tape. Information required to perform subsequent functions is frequently omitted. This data is also written onto the tape label.
NEW BOXED	New tapes received are left in boxes near the cabinets. The physical parameters of the tapes appears on the labels. The box is accessible to anyone.
SINK	Tapes sometimes disappear.

a. DFD(O): Proposed System Overview

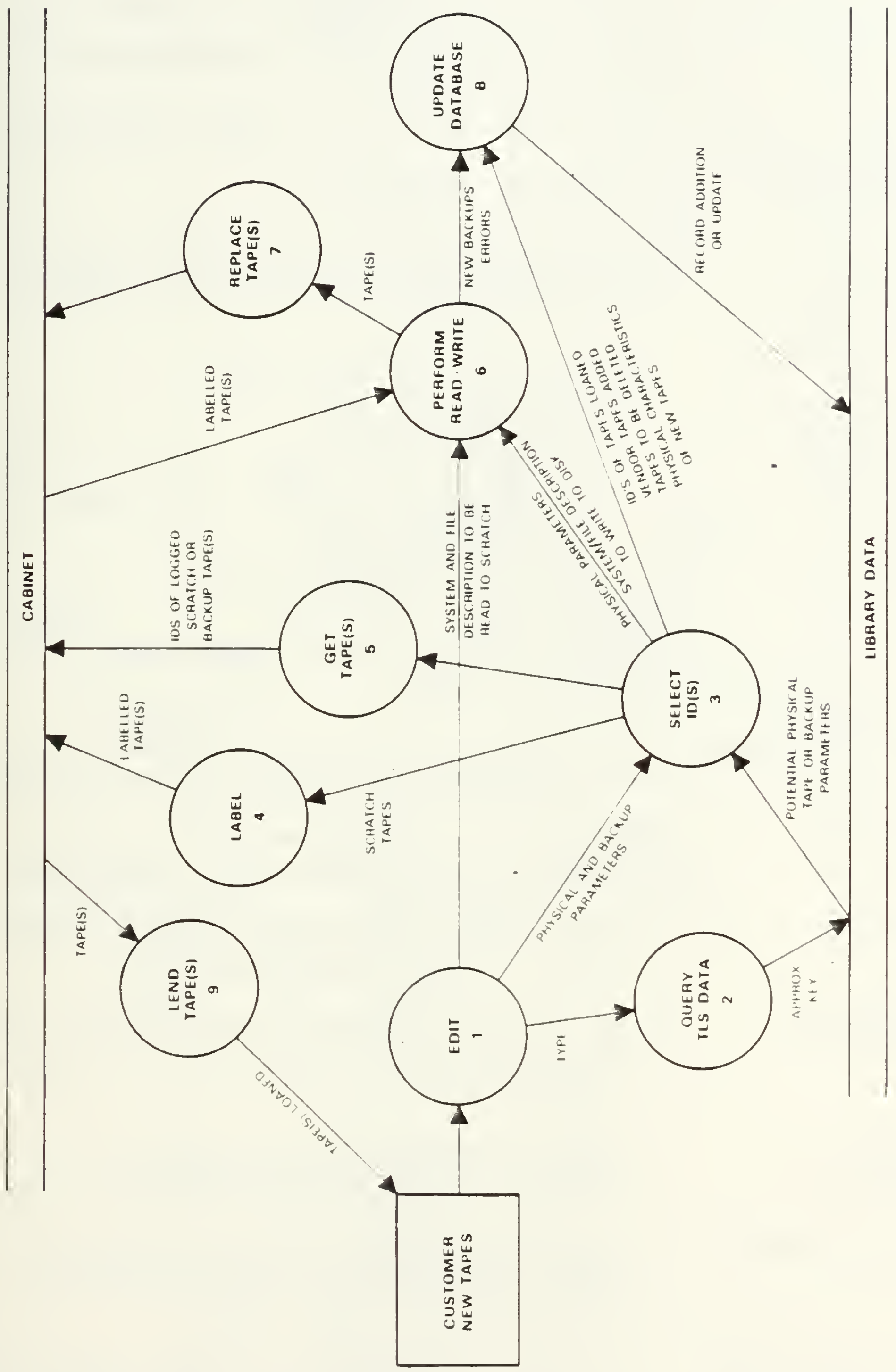

CABINET
LIBRARY DATA
CUSTOMER NEW TAPES
EDIT 1
QUERY TLS DATA 2
SELECT ID(S) 3
LABEL 4
GET TAPE(S) 5
PERFORM READ-WRITE 6
REPLACE TAPE(S) 7
UPDATE DATABASE 8
LEND TAPE(S) 9
TAPE(S) LOANED
TAPE(S)
TYPE
APPROX KEY
PHYSICAL AND BACKUP PARAMETERS
POTENTIAL PHYSICAL TAPE OR BACKUP PARAMETERS
SCRATCH TAPES
LABELLED TAPE(S)
IDS OF LOGGED SCRATCH OR BACKUP TAPE(S)
SYSTEM AND FILE DESCRIPTION TO BE READ TO SCRATCH
PHYSICAL PARAMETERS SYSTEM/FILE DESCRIPTION TO WRITE TO DISK
ID'S OF TAPES LOANED ID'S OF TAPES ADDED VENDOR TO BE DELETED TAPES CHARACTERISTICS PHYSICAL OF NEW TAPES
LABELLED TAPE(S)
TAPE(S)
NEW BACKUPS ERRORS
RECORD ADDITION OR UPDATE

Figure 2. Proposed System Overview

Function: General Procedures of the Proposed Library

<u>Mini-specifications 0</u>

The following procedures are required by most of the functions the librarian performs. The combination of procedures and the data input and output for each procedure will vary according to the function requested.

1.	Edit	An event/discussion with a requestor to determine the function required. For example: Add empty cabinet slots Add scratch tapes Add vendor tapes Write disks to tape: To partially used individual tape To backup tape To scratch tape Read active tape to disk Loan Tape Receive returned loaned tape Delete tapes
2.	Query TLS Data	Provide criteria upon which the information requested is retrieved.
3.	Select IDs	Select the id numbers which are required to perform the function.
4.	Label	Make labels for, and label objects for each id number selected.
5.	Get Tape(s)	Go to the cabinet slot labelled with the selected id number and withdraw the tape(s).
6.	Perform Read/Write	Perform write disk to tape or read tape to disk, as appropriate.
7.	Replace Tape(s)	Replace tape(s) to the cabinet slot which matches the identification number on the tape label.
8.	Update TLS Data	Add to, or update TLS data with new information regarding the operation performed.

b. DFD(A)

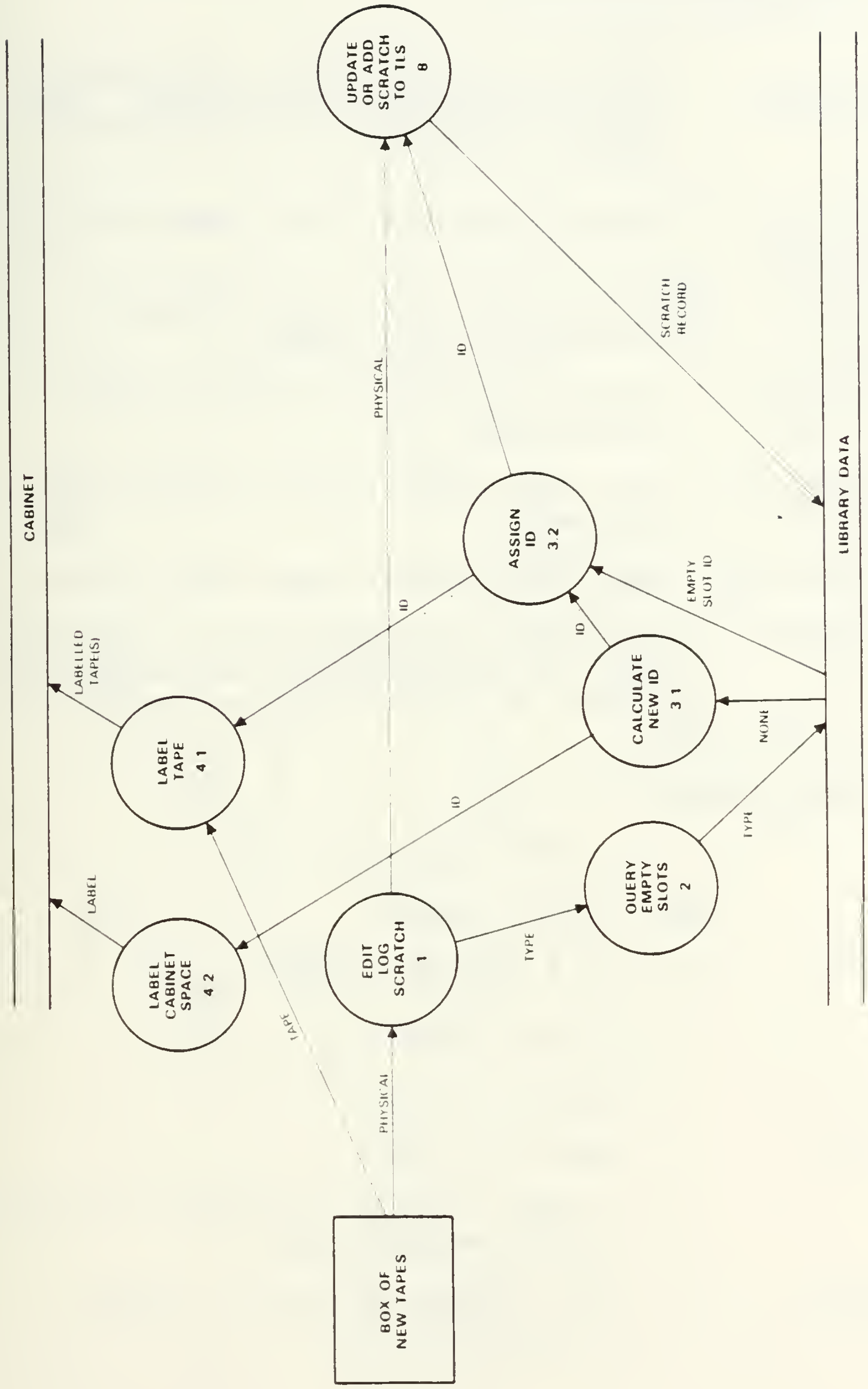


Figure 3. Add New Scratch Tapes to Library

Function: Add new scratch Tapes to Library

Mini-specifications A

1.	Edit: Log Scratch	Request to log scratch. Log new unlabelled tape(s) upon arrival. Read physical parameters from manufacrer's label.
2.	Query Empty Slots	Query library data for lowest empty id.
3.1	Calculate New id	If no empty slots are available, calculate new id number
3.2	Assign id(s)	Select the id(s) to assign to the scratch tape
4.1	Label Tape(s)	Label the tape(s) with the assigned slot id number.
4.2	Label Cabinet	Label the cabinet space with the new id
8.	Add Scratch	If id was of an empty slot, update tape physical parameters to the id record. If new id, add id and tape parameters.

Data Items A

Physical	=	length + density
Type	=	(type = empty)
Empty Slot id	=	id + (type = empty)
None	=	no empty slots
New id Selected	=	max id + 1
Labelled Tape	=	Tape + id
Label	=	a label with an id number to be put on the cabinet space
Scratch Record	=	id + (Type = scratch) + length + density

c. DFD(B)

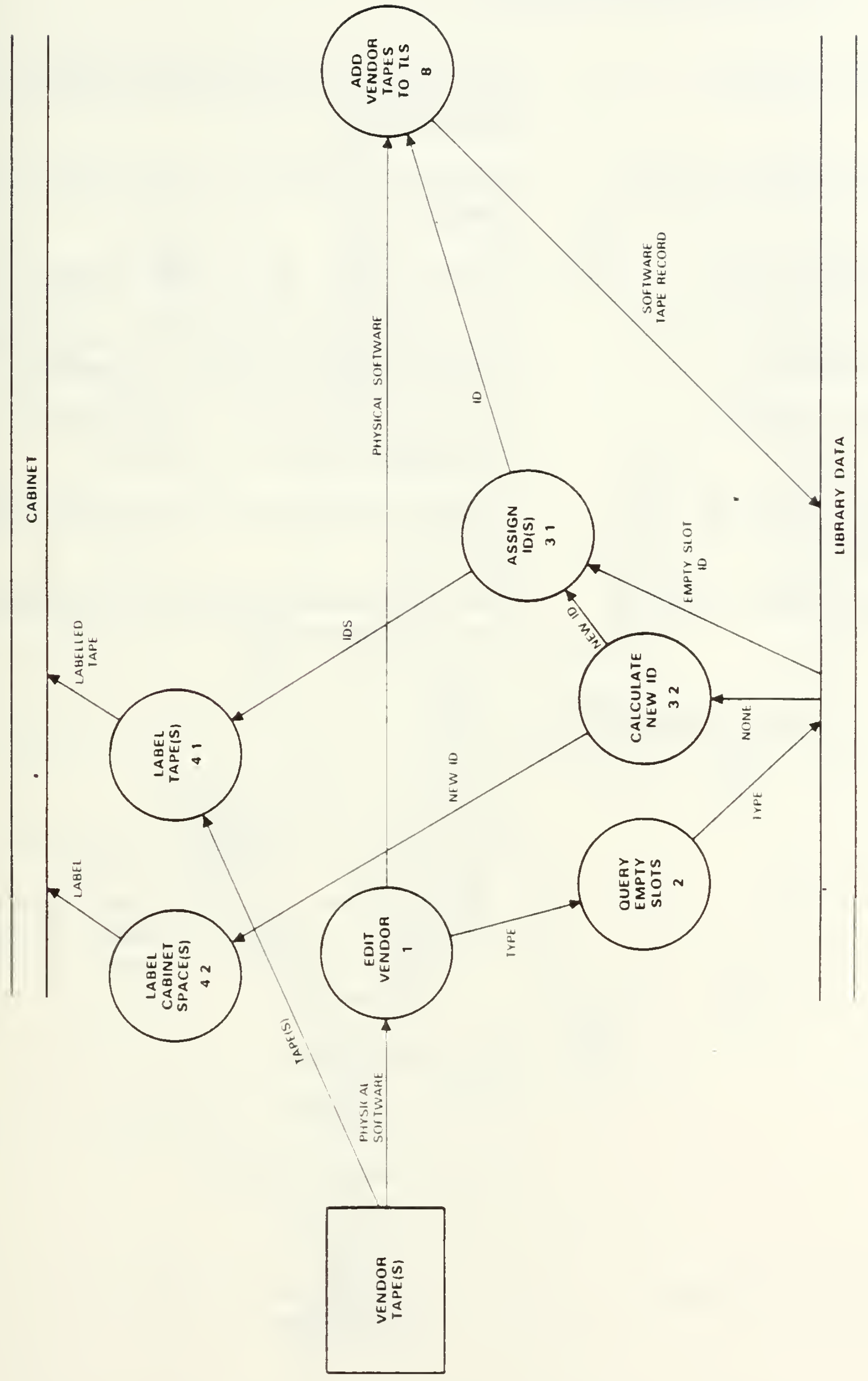


Figure 4. Add Vendor Tapes

Function: Add Vendor Tapes

<u>Mini-specification B</u>

1.	Edit: Vendor	The tape is software from a vendor for the library to log and maintain
2.	Query Empty Slots	Query the library data for lowest empty id
3.1	Assign id(s)	Assign the id to the vendor tape
3.2	Calculate New id Numbers	If no empty slots are available, calculate the new id number
4.1	Label Tape(s)	Label the tape(s) with the assigned slot id number
4.2	Label Cabinet Space(s)	Label the cabinet space with the new id
8.	Add Vendor Tape(s) to TLS	Add software tape(s) record to the library data

<u>Data Items B</u>

Physical	= bpi
Software	= (type = vendor) + system + utility + label + date received + retirement date + owner + contact phone + software + vendor + description + sequence
Type	= Type empty
Slot id	= id + (type = empty)
None	= No empty slots
New id	= max id + 1
Software Tape Record	= id + (type = vendor) + bpi + software

d. DFD(C)

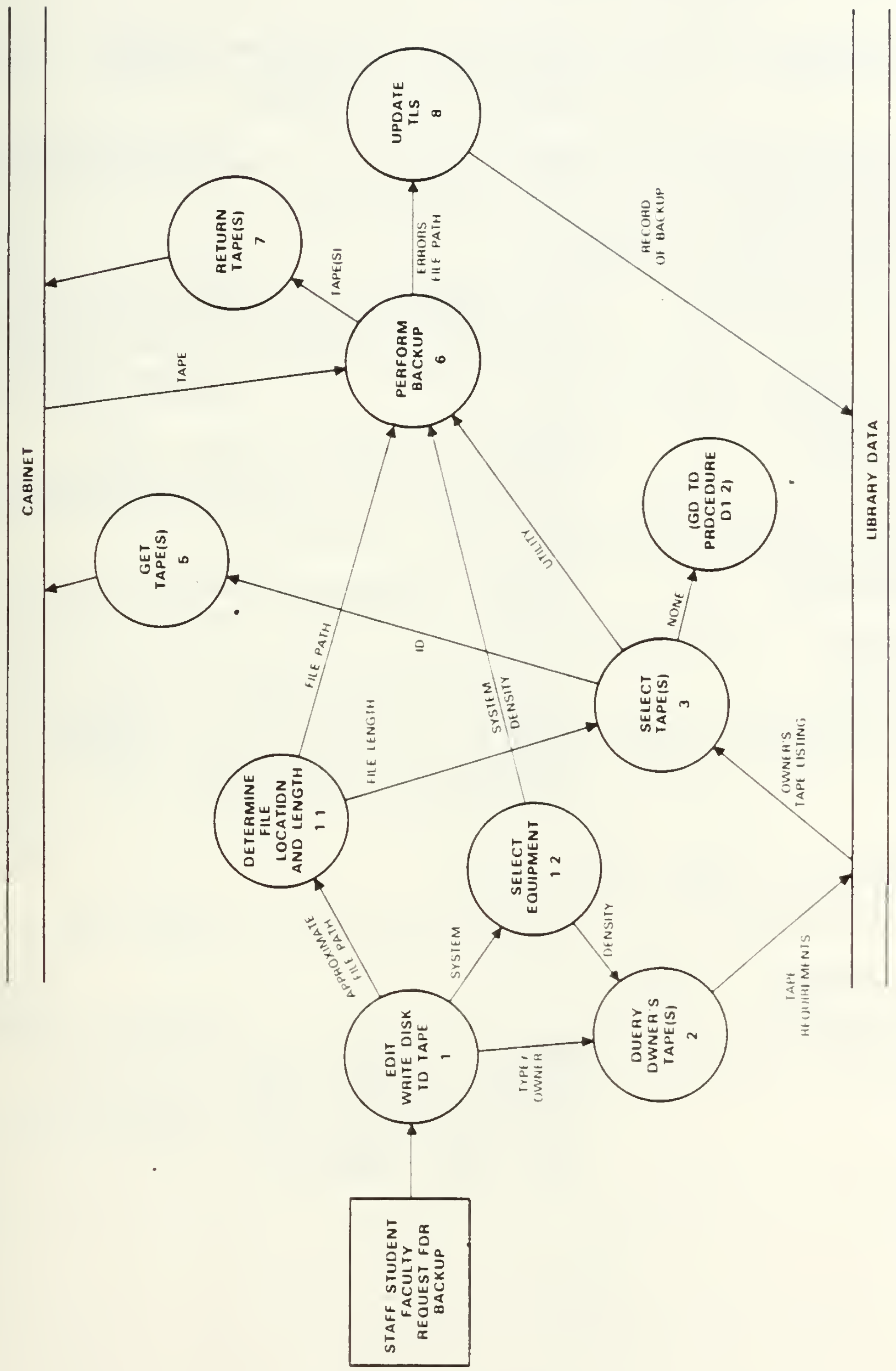


Figure 5. Write Individual's Disk to Tape

Function: Write Individual's Disk to Tape

Mini-specification C

1.	Edit: Write Disk to Tape	If type = individual staff, faculty, or student backup. If not go to DFD(E).
1.1	Determine File Location and Length of Files	By discussion or may require searching files.
1.2	Select Equipment	The computer and operating system combination determines the device. The device determines the density which can be used.
2.	Query Owner's Tape(s)	Query the library data for a listing of the owner's backup tapes which may be used to copy the files requested.
3.	Select Tape	Select a backup tape which may be used to write the requested file.
D.1.2	Go to Procedure D.1.2	If none of the owner's tapes are appropriate to use, go to Procedure 1.2 of DFD(D) and resume looking for a scratch tape to perform the write.
5.	Get Tape	
6.	Perform Backup	Write disk files to the tape.
7.	Return Tape	
8.	Update TLS	Add the record of the new files to the library data. If the utility reads error information different than that recorded for the tape, update.

Data Items C

Type	=	individual, staff, faculty, or student backup
Owner	=	the owner of the individual backup tape
Approximate File Path	=	(some or all) (system + owner + partition/disk)
Type	=	individual, staff, faculty, or student
Density	=	bits per inch (bpi)
Tape Requirements	=	(some or all) (type = individual) + owner + system + density
Owner Tape Listing	=	id + length + density + loaned + system + utility + label, percent used + retirement date
File Path	=	Partition/disk + saveset/ source
File Lengths	=	Approximate length of magnetic tape needed to copy the files
Utility	=	Utility
None	=	No tapes meet the requirements for the backup
Errors	=	Error, date of errors
Record of Backup	=	id + errors + date of errors + partition/disk + saveset/ source

e. DFD(D)

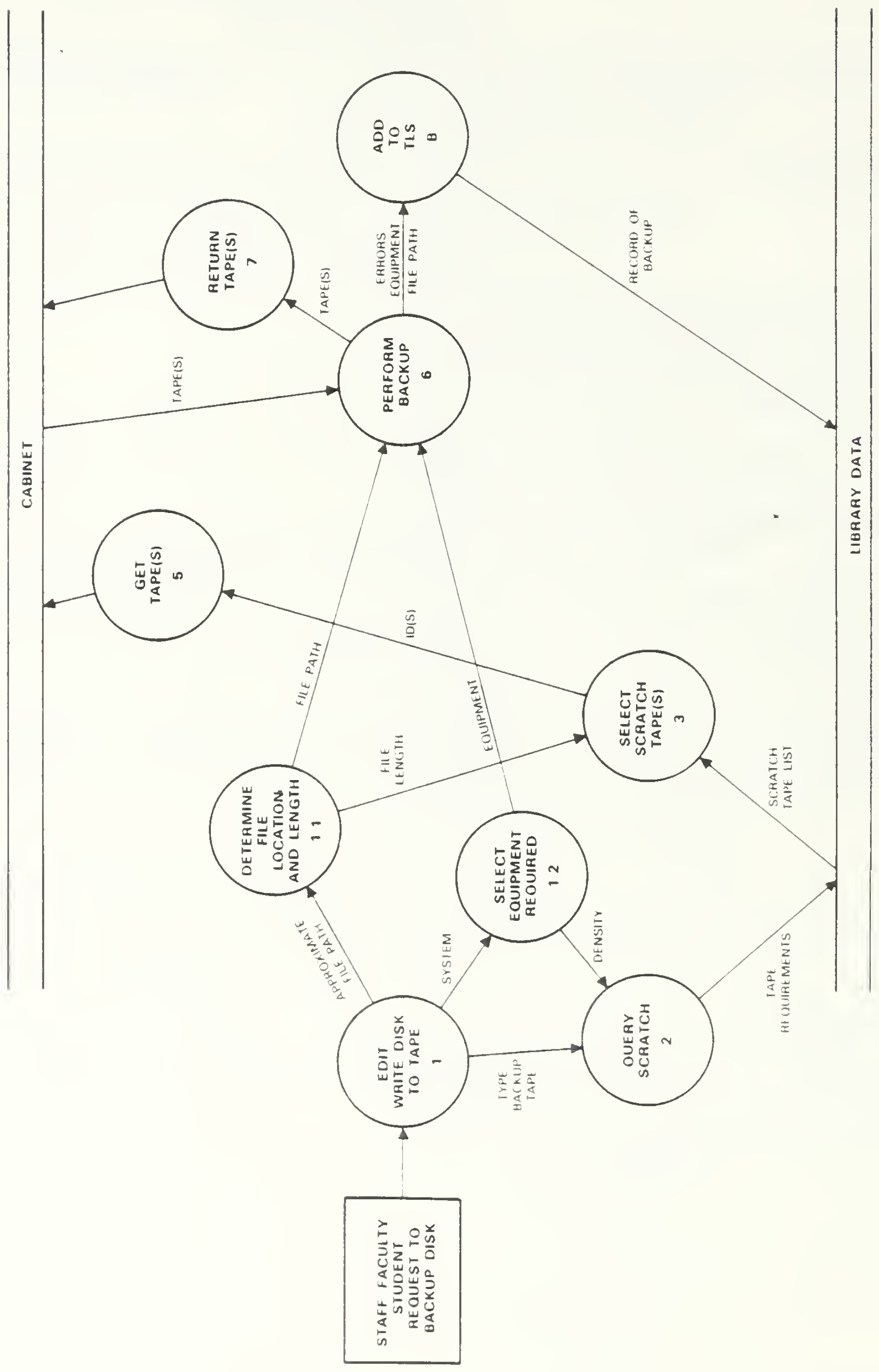


Figure 6. Write Disk to Scratch Tape

Function: Write Disk to Scratch Tape

Mini-specification D

1.	Edit: Write Disk to Scratch	If type = daily, weekly, monthly, or graduation backup
1.1	Determine File Location and Length	By discussion or may require searching files
1.2	Select Equipment Required	The computer and operating system combination determines the device. The device determines the density which can be used. The utility is determined by the system and the density
2.	Query Scratch	Query the database for scratch tapes which have the physical parameters required by the edit procedure
3.	Select Scratch Tape(s)	Select ids from the listing of scratch tape(s) provided by the database
5.	Get Tape(s)	Retrieve tape(s) with the id(s) selected
6.	Perform Backup	Write disk files requested to the tape
7.	Return Tape(s)	
8.	Add to TLS	Enter the information regarding the backup to the library data

Data Items D

Function: Write Disk to Scratch

Approximate File Path	=	(some or all) (system + part/disk + saveset/source)
Type Backup	=	daily, weekly, monthly, graduation, or individual backup
Type Tape	=	Type = scratch

Density	=	bits per inch (bpi)
Tape Requirements	=	(Type = scratch) + length + density + errors + date of errors
Scratch Tape List	=	id + length + density + errors + date of errors + loan
File Length	=	Approximate length of magnetic tape needed to copy the file(s)
File Path	=	Partition/disk + saveset/source
Equipment	=	System + utility + density
Errors	=	Error + date of errors
Equipment	=	System + utility + density
Record of Backup	=	Various according to type:
Daily	=	id(s) + type + errors + date of errors + label + sequence + date created + retirement date + system + utility + description + partition/ disk + saveset(s)/source(s)
Weekly/Monthly	=	id(s) + type + errors + date of errors + label + sequence + date created + retirement date + system + utility + description + partition/ disk + saveset/source
Graduation	=	id(s) + type + errors + date of errors + label + sequence + date of graduation + retirement date + system + utility + description + partition/ disk(s) + saveset(s)/source(s)
Individual	=	id(s) + type + errors + date of errors + label + sequence + percent used + date created + retirement date + system + utility + description + owner + advisor + contact phone + partition/disk(s) + saveset(s)/ source(s)

f. DFD(E)

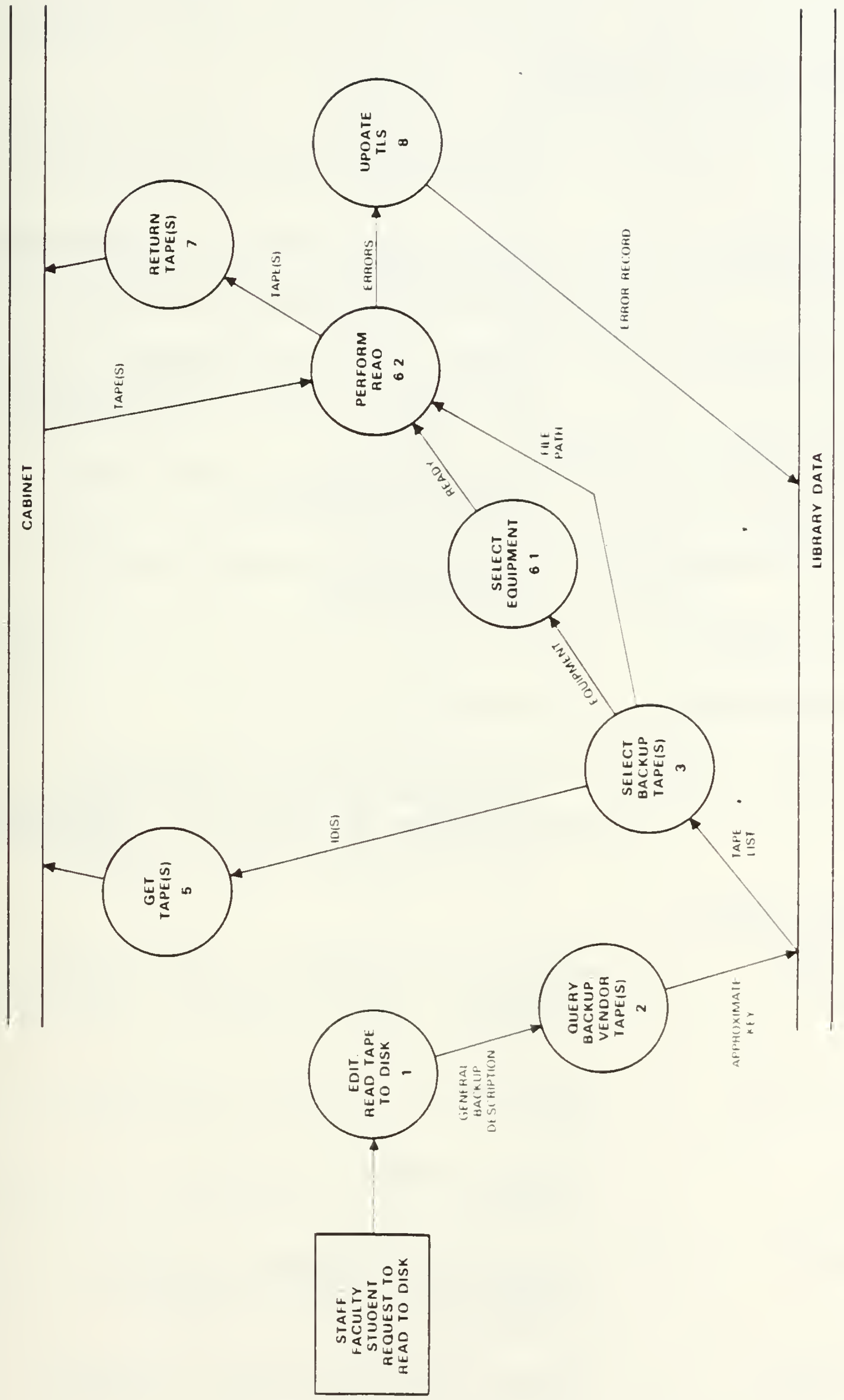


Figure 7. Read Tape to Disk

Function: Read Tape to Disk

Mini-specification E

1.	Edit: Read Tape to Disk	Requestor would like a vendor or backup tape read to disk.
2.	Query Backup/ Vendor Tapes	Query the library with as much information as the requestor can provide concerning the tape.
3.	Select Backup Tape(s)	Select from the database listing the exact tapes needed to perform the read.
5.	Get Tape(s)	Retrieve tape(s) with the id numbers selected.
6.1	Select Equipment	Select the equipment defined in the database.
6.2	Perform Read	
7.	Return Tape(s)	
8.	Update TLS	If the physical information provided by the utility regarding the tape is different than that in the library data, update.

Data Items E

General Backup Description	=	(Some or all) type + date created + system + name of owner + software + vendor
Approximate Tape	=	(Some or all) Varies according to type:
Vendor	=	(Type = vendor) + system + software + vendor
Periodic	=	(Type = daily, weekly, monthly, or graduation) + date + system + partition/disk
Individual	=	(Type = staff, faculty, student) + system + owner

Tape List	=	Varies according to type backup:
Vendor	=	id(s) + density + loan + errors + label(s) + sequence + date received + system + utility + description + owner + contact phone + software + vendor + partition/disk + saveset(s)/ source(s)
Periodic	=	id(s) + (type = daily, weekly, monthly, or graduation) + errors + label + sequence + date + system + utility + description + partition/ disk + saveset(s)/source(s)
Individual	=	id(s) + (type = staff, faculty, student) + density + loan, errors + label(s) + sequence + percent used + date + system + utility + owner + advisor (if student) + contact phone + description + partition(s)/disk(s) + saveset(s)/source(s)
Equipment	=	system + utility + density .
File Path	=	partition/disk + saveset/source
Errors	=	errors + date of errors
Error Record	=	id + error update

g. DFD(F)

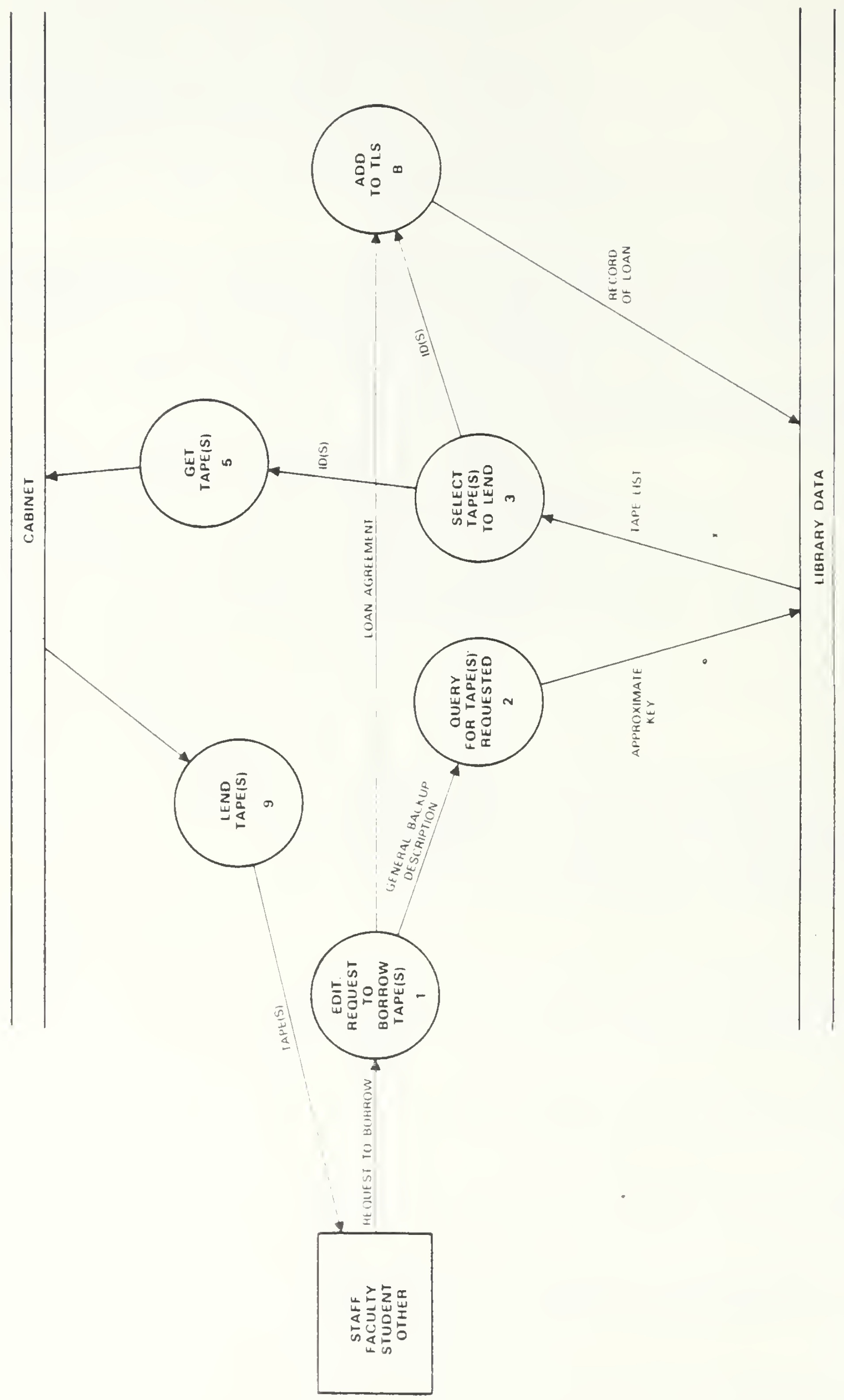


Figure 8. Loan Tapes

Function: Loan Tapes

<u>Mini-specification F</u>

1.	Edit: Request to Borrow Tape(s)	If requested by anyone but a librarian to remove a tape(s) from the library. Type tape must be vendor, individual, staff, student, faculty. Permission from the owner should be obtained before lending.
2.	Query for Tape(s) Requested	Query database for id of tape(s) requested.
3.	Select Tape(s) to Lend	Select ids of requested tape(s) from database listing.
5.	Get Tape(s)	Retrieve tape(s) with the id numbers selected.
8.	Add (Loan Information) to TLS	Update the library to contain information regarding the loan.

<u>Data Items F</u>

General Backup Description	=	(Some or all) type + date created + system + name of owner + software + vendor
Approximate Key	=	(Some or all) Varies according to type:
Vendor	=	(Type = vendor) + system + software + vendor
Periodic	=	(Type = daily, weekly, monthly, or graduation) + date + system + partition/disk
Individual	=	(Type = staff, faculty, student) + system + owner
Tape List	=	Varies according to type backup:
Vendor	=	id(s) + density + loan + errors + label(s) + sequence + date received + system + utility + description + owner

		+ contact phone + software + vendor + partition/disk + saveset(s)/source(s)
Periodic	=	id(s) + (type = daily, weekly, monthly, or graduation) + errors + label(s) + sequence + date + system + utility + description + partition/disk + saveset(s)/source(s)
Individual	=	id(s) + (type = staff, faculty, student) + density + loan, errors + label(s) + sequence + percent used + date + system + utility + owner + advisor (if student) + contact phone + description + partitiion(s)/disk(s) + saveset(s)/source(s)
Loan Agreement	=	loan + date loaned + estimated date of return + loanee + loanee's contact phone
Record of Loan	=	id + loan agreement

h. DFD(G)

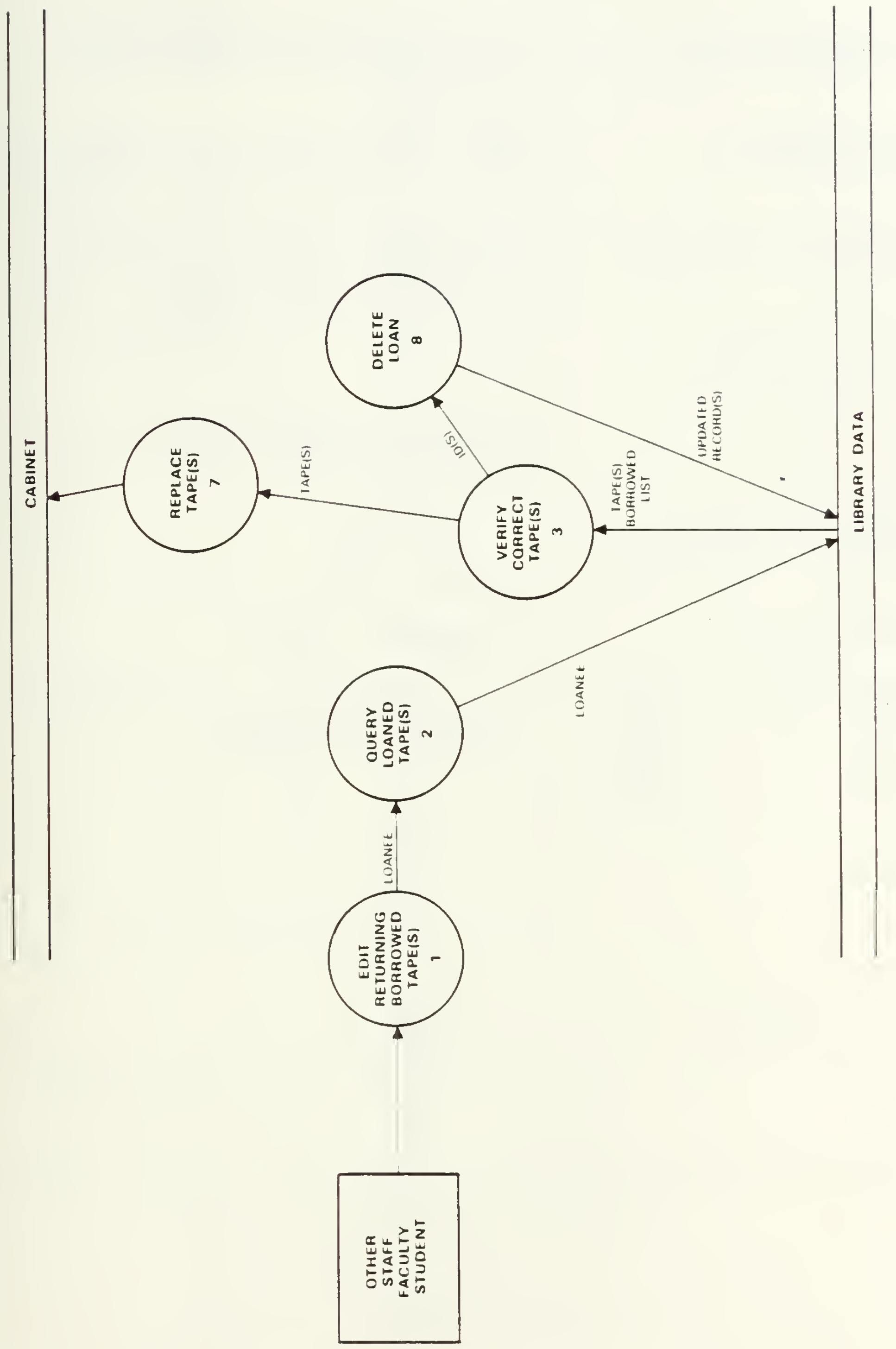


Figure 9. Return Loaned Tapes

Function: Returned Loaned Tapes

Mini-specification G

1.	Edit: Returning Borrowed Tape(s)	The requestor (loanee) returns tape(s) borrowed from the library.
2.	Query Loaned Tape(s)	Query the library to find which tapes are held by the loanee.
3.	Verify Correct Tape(s)	Verify the returned tape(s) belonging to the library, that all are present, etc.
5.	Replace Tape(s)	
8.	Delete Loan	Delete the record(s) of the loan from the library data.

Data Items G

Tape	= tape + id
Loaned Tape List	= loanee + id(s)
Updated Record(s)	= id + (loan = no) + (delete record of loan)

I. DFD(H)

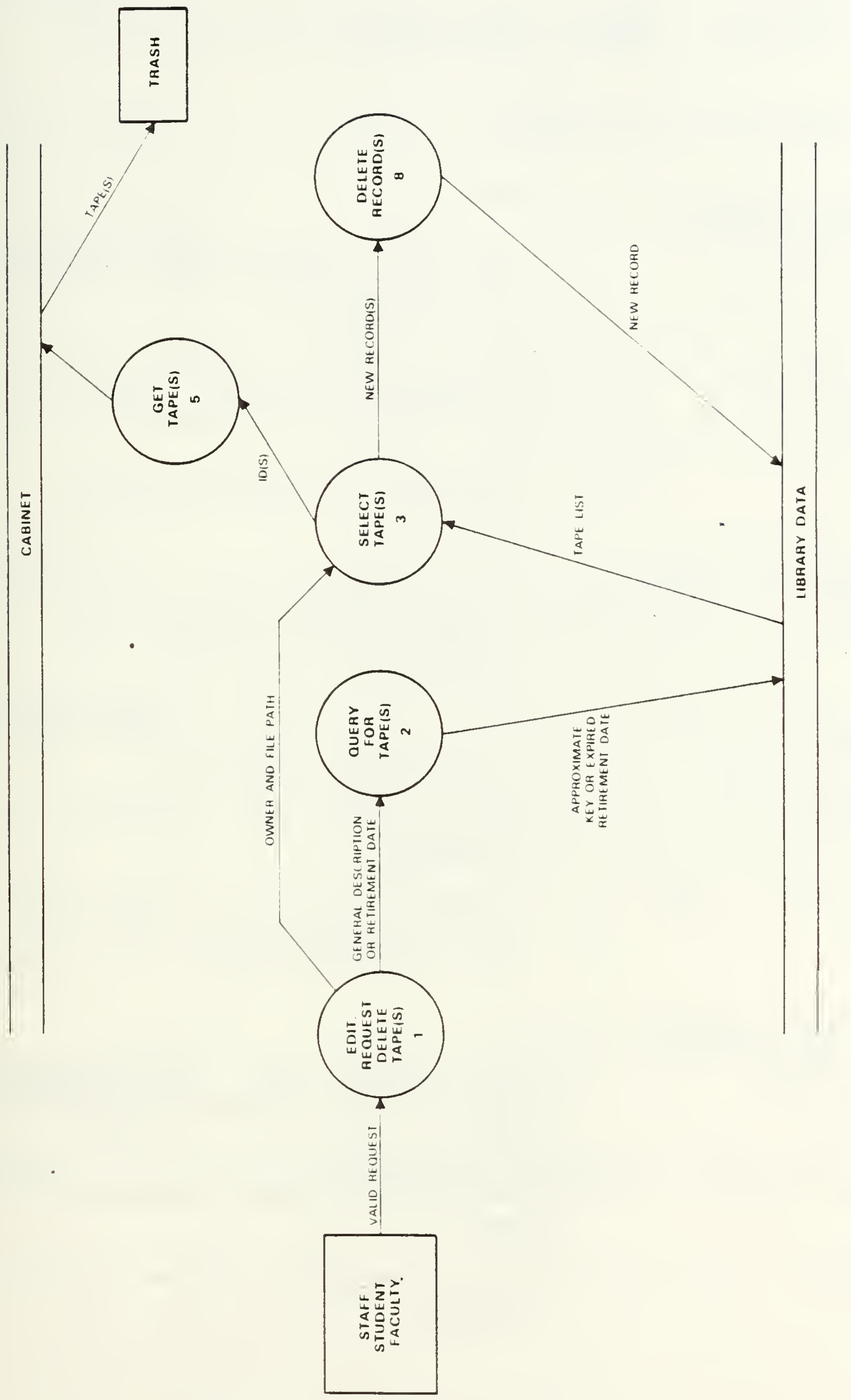


Figure 10. Delete Tapes

Function: Delete Tapes (erase backup or trash physical tape)

Mini-specification H

1.	Edit: Request to Delete Backup or Tape(s)	Valid if: Any type tape is first copied to another tape and then logged into the library. Daily, weekly, monthly, or graduation backup = retirement date <= current date. Vendor or individual staff, faculty, student = owner = requestor.
2.	Query for Tape(s)	Query library for a listing of tape(s) which satisfy the requestors description of the tape(s).
3.	Select Tape(s)	Select the id(s) of the tape(s) desired for deletion.
4.	Get Tape(s)	Included only if trashing tape(s).
8.	Delete Records	Delete records to reflect the erasure of backup or the trashing of a physical tape

Data Items H

General Backup Description	=	(Some or all) type + date created + system + name of owner + software + vendor
Approximate Key	=	(Some or all) Varies according to type:
Vendor	=	(Type = vendor) + system + software + vendor

Periodic	=	(Type = daily, weekly, monthly, or graduation) + date + system + partition/disk
Individual	=	(Type = staff, faculty, student) + system + owner
Expired Retirement Date	=	Requirement date <= today's date
Tape List	=	Varies according to type backup:
Vendor	=	id(s) + density + loan + errors + label(s) + sequence + date received + system + utility + description + owner + contact phone + software + vendor + partitioin(s)/disk(s) + saveset(s)/source(s)
Periodic	=	id(s) + (type = daily, weekly, monthly, or graduation) + errors + label(s) + sequence + date + system + utility + description + partition/disk + saveset(s)/source(s)
Individual	=	id(s) + (type = staff, faculty, student) + density + loan, errors + label(s) + sequence + percent used + date + system + utility + owner + advisor (if student) + contact phone + description + partition(s)/disk(s) + saveset(s)/source(s)
ids	=	ids
New Record:		
If to Delete Tape Data	=	(type = scratch) delete backup record
If to Trash Tape	=	(type = empty) delete physical and backup record

C. DFD DATA DICTIONARY

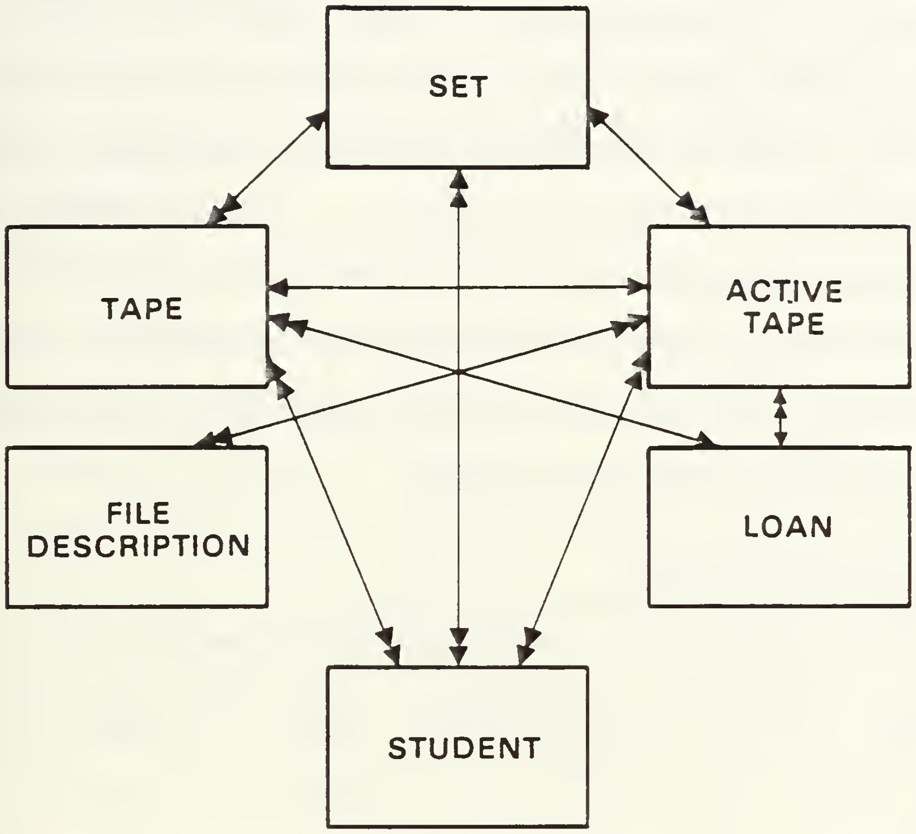
Advisor	=	The name of a student's advisor
Contact Phone	=	The phone number of the owner
Date of Errors	=	The date the errors were provided by the utility
Date of Graduation	=	The graduation date of the students whose files are contained in a graduation backup tape
Date Loaned	=	The date a tape is loaned
Date Received	=	The date the tape was received by the library
Density	=	The bits per inch which may be read to tape
Description	=	Notes by the librarian regarding the software
Disk	=	Used with the VMS operating system to define a major subdivision of files of disk
Errors	=	The number of "bad bits" on a physical tape. The figure is provided by the utility when a tape is read or written
Estimated Date of Return	=	The date the librarian assigns as the date the tape should be returned to the library
Identification Number (id)	=	A unique number assigned a slot in the cabinet. Slots are numbered consecutively left to right, top to bottom, so the number may be used to locate the slot. When a tape is logged into the library it is assigned the identification number of the slot in which it is stored.

Label	=	The VMS operating system requires that each tape be assigned a "label" by the librarian. When the tape is mounted, the label verifies to the system that the tape is VMS. If the "copy" utility is used to read or write the tape, the label must be entered by the librarian when mounting the tape.
Length	=	The total length of a reel of tape
Loaned	=	Whether or not the tape is loaned from the library
Loanee	=	The person to whom the tape is loaned
Loanee Contact Phone	=	The phone number of the loanee
Owner	=	The name of the owner of the tape
Partition	=	Used by the UNIX operating systems to define a major subdivision of files
Percent Used	=	The percent of the tape containing backup data
Retirement Date	=	The date the librarian assigns to the tape for retirement
Saveset	=	Used with the VMS "backup" utility to mark the beginning of each file or set of files
Software	=	The name of the software on the tape
Source	=	Used with other than VMS operating system, and "backup" or "copy" utilities to describe a file or set of files on a tape
System	=	The combination hardware and operating system by which the tape data may be read or written
Type	=	The use of a tape maintained by the library

Type Daily	= Disks are read to tape each day. These daily tapes are written to one set of tapes each week and identified as a daily backup
Type Graduation	= A backup tape containing disk files of students of a graduated class
Type Individual	= The type tape is an individual staff, faculty, or student's backup tape
Type Monthly	= Each month, all disks are read to tape and identified as monthly backups
Type Scratch	= Blank tapes assigned to slots in the library available to read data from disk
Type Vendor	= Tapes containing vendor software
Type Weekly	= Each week, all disks are read to tape and identified as a weekly backup
Utility	= The software utility which must be used to read or write the tape data
Vendor	= The name of the company which produced the software

III. LOGICAL SCHEMA

A. BACHMAN DIAGRAM



B. SPECIFICATION DATA DICTIONARY

The Specification Data Dictionary defines the entities and data elements in the tape library. It groups each of the data elements under the entity to which it is logically related. The definitions describe the meaning and use of the entities and elements in the library. The Specification Data Dictionary is provided in Appendix A.

IV. DESIGN

In this chapter the design of the TLS is described in detail. First the logic of the design is discussed. It is followed by a description of the tables. A chart of the application module and a description of each frame is provided. The modules association with the functions described by the DFDs are provided by charts. Integrity, security, the database administrator, and validation and test are discussed. The design data dictionary and frames definition are derived from this chapter and are provided in Appendices C and D for ease in future reference.

A. LOGIC OF THE DESIGN

Several factors were considered before mapping the logical schema to the Ingres database. They were prioritized as follows:

- User
    - Interactive standardization and validation of input
    - Simplicity
    - Librarian functions
- Ingres
    - Database requirements
    - Query by forms (QBF) limitations
    - Report by forms (RBF) flexibility
    - Applications by forms (ABF) capabilities
- Logical Schema

1. User

Non-standard record input, incorrect data input, and an inability to control the sequential consecutive assignment of identification (id) numbers are the major problems in the current library system. An interactive forms-based input method with validation capabilities can correct these problems. The TLS must be kept simple and quick to facilitate the task of logging tapes and retrieving data.

2. Ingres

Ingres is a relational DBMS. It provides a means to store data in tables. The basic method for entering, retrieving, and manipulating data is by use of its data manipulation language, "QUEL". The user of QUEL must understand the language commands and syntax, and the structure and content of each table. Data manipulation with QUEL is too time consuming and distracting for the daily functions of the librarians. However, Ingres provides Query By Forms (QBF), an automated forms processing interface, which simplifies data manipulation. The user may interactively append, delete, replace, or retrieve data from one table at a time in the database without knowledge of QUEL.

QBF provides an automated method for the functions of append, query, update, and deletion of values of records of one table at a time. It presents forms to the user which contain fields from the appropriate table. By default, QBF presents all of the table's fields to the user. The forms may be edited by the application developer by using the Visual Forms Editor (VIFRED) to limit the data the user can input or manipulate. The editing capabilities include the deletion or addition of fields in a form, definition of field parameters, and definition of field attributes for validation, mandatory input, error messages, automatic repetition of field values, and special display modes.

QBF requires different user procedures for each of its functions. The Append function must be used for initial entry of a record. The Append function has an automated facility which simplifies the sequential input of records which have many fields of equal value. After the first record is entered, the user may duplicate the field values in subsequent records by pressing a function key. This simplifies input and can be used to remind the user of the previous value input.

The Update function must be used in QBF to replace or delete values in existing records. When using Update, the first display of fields is presented for the user to query the record or records desired. After the query is entered, Ingres retrieves the records and displays them on the terminal, one at a time, for update. Update does not have the duplication facility.

Four limitations of QBF greatly affected design: 1) The automatic field value duplication facility may be used in Append only; 2) When more than one record is retrieved by a query in the Update function, they appear on the terminal one at a time. This format is not useful for reports or records; 3) QBF only allows manipulation of data on one table at a time; and 4) QBF does not provide for any internal manipulation and assignment of data values.

Report By Forms (RBF) is a forms-based record specifier available for retrieving reports without knowledge of QUEL. RBF allows the developer to select, design, and present multiple record reports from one table at a time, through the use of views. RBF's facility to present reports with data from multiple tables greatly influenced the design of the application.

The developer may integrate the functions and forms of QBF and RBF into a single application by using Ingres Applications By Forms (ABF). The application uses forms as the vehicle for interaction between the application and user. The user uses menus to select operations desired and forms to append, query, or update, or to retrieve reports from the database. Knowledge of QUEL is not required. ABF allows the developer greater flexibility than the functions available in QBF and RBF. RBF, however, was sufficient for the library in certain cases, whereas QBF was not. In most cases, it was impossible to match a task with only one table. In QBF, one task would have required the librarian to remember to add new values to several tables. To maintain data integrity, it would be unwise to leave this responsibility to the librarian. Additionally, the assignment of id numbers required control to prevent duplication and ensure consecutive, sequential numbers. This required a search of current id numbers, a possible arithmetic operation on the maximum id and assignment of the result to the form. ABF enables such flexibility by allowing development and integration of procedures into the application by the developer. The procedures may be used with forms developed by the use of VIFRED. The forms have the same capabilities as those described above. ABF will be discussed further in Section D.

3. Logical Schema

a. Data Class TAPE

The data class TAPE was mapped intact to table TAPE in the logical schema. The fields values in table TAPE provide physical tape characteristics which retain their importance throughout the life of the tape's existence in the library whether or not any data on the tape is of importance to the library.

The TAPE table stores all of the id numbers assigned in the library. As discussed above, assignment of tapes to id numbers, and the addition of new ids to the library are controlled by the application. The librarian will typically enter the values (id, length, and bpi) of the TAPE record when adding new blank tapes or new vendor tapes to the library. Such tapes will have the same values for length and bpi. Therefore, the task is greatly facilitated by providing the field attribute of repeating values to the form definition. Data integrity and future retrieval is assured by writing validation criteria into the form definition. Two fields, errors and date of errors, are entered and maintained by the librarian at other times - after the tape is read or written. As a separate task, the record is updated. After an active tape is retired, but the physical tape is retained by the library for rewrite, all of the values in TAPE are retained. If the tape is thrown away, all of the values of TAPE are updated to null except the id number which is retained. Such an id number is known as an empty slot and will be used by the application for future assignment to a new scratch or vendor tape.

b. Data Class SET

Data class SET must be related to TAPE so that the id number in SET allows the librarian to access the physical characteristics in TAPE when a tape is to be read or written.

The key to data class SET is long and variable according to its type active tape. Therefore, field id was added to SET. Since data class ACTIVE TAPE values are entered and deleted at the same time as SET, and since ACTIVE TAPE is logically closely related to SET, it is contained with data class SET, in a table named SETS. New records are added to SETS at the time a backup is read, or when a new vendor tape is input. As discussed above, when new vendor tapes are added, SETS values are appended to the SETS table by the application.

The timing for entering backup tape data is different. In order to read a backup to tape, the librarian must first find an appropriate scratch tape in table TAPE. New records are added to table SETS using QBF-append. As each record is added for each id, common values (those of data class SET) are automatically called to the form after the first entry. Only the fields expected to be different for each id (those of data class ACTIVE TAPES) will be entered by the librarian. If the librarian loses track of which id was previously entered, (s)he may use the function key to recall it.

c. Data Class FILE PATH

Data class FILE PATH is contained in table FILE. There are many file paths on one active tape or one file path on several tapes of one set.

Each FILE PATH record must be linked to an id number so the librarian can locate the particular tape id(s) on which the file path was located. Therefore, id was added to table TAPE. File records are usually added at the same time new SETS records are added, i.e., when a new vendor tape is received or when a new backup is read. In this case, new FILE records are added using the QBF-append function. Since one id may have many FILE records, or one FILE record may have many ids, no repeating values are defined in the form field attributes. The librarian may repeat values by using the function key. The only exception to the time of entry of FILE PATH discussed above is on the infrequent occassion when the librarian adds a new file path to an individual's old tape. This task requires fields from two tables to be changed - FILE and SETS. In this case, a separate procedure is provided in the application to ensure the integrity of both tables.

d. Data Class STUDENT

Data class STUDENT is contained in table STUDENT. New records are added to STUDENT each time a new graduation backup is added to SETS. Eventually, the librarian must find on which tapes student files are contained by student name. Therefore, id was added to STUDENT. New records are added using QBF-append. A student may have files on many tapes. Data class STUDENT fields are, therefore, repeated by the form after the first entry.

e. Data Class LOAN

Data class LOAN is included in table LOAN. Since tapes are added to table LOAN on an individual basis as frequently as they are by SETS, id was added to the table. LOAN records are added when a tape is loaned using the QBF-append. They are deleted when a tape is returned.

B. TABLES AND VIEWS

Initial input of library data are stored in the hash storage structure in which records are stored randomly; new rows are added to the bottom. Therefore, the entire table must be scanned for retrieval. The majority of retrievals and comparisons in the library involve selection of records based on id number. Hash tables store each row at an address determined by a column or columns in a record. Records may be located directly by the value of the address, without scanning the entire table. The hash storage structure can greatly excellerate queries run on tables with a large number of rows. After the current library is input, it is recommended that all tables be converted to the "hash" Storage Structure [Refs. 1 and 2], using the id number as the "address" column. Thereafter, the hash structure should be modified when 20 percent of the table is changed. The tables and views are contained in Appendix B.

TABLES	VIEWS
file	vind
loan	vloan
sets	vpe
student	vretire
tape	vscloan
	vscratch
	vstudent

C. DESIGN DATA DICTIONARY

The Specification Data Dictionary (Appendix A) defines fields in terms of how they are used in the library. The Design Data Dictionary (Appendix C) defines fields in terms of how they are used in the TLS application. Both should be read for full comprehension of the field's use.

D. THE APPLICATION

The tape library was developed using Ingres Application By Forms (ABF), and is an interactive application with a forms interface. The application user interacts with the form by filling in fields with data. The application interfaces with the form by getting values from it, and sometimes placing values on it.

ABF applications are built by the use of frames. The frame is the principal vehicle for interaction between the user and the forms application. Each full screen the user sees is a frame. A frame is composed of a form and a procedure. The upper portion of the frame is the form.

The developer designs the form. It may include trim and/or fields. When defining a form the developer only includes the fields which may be manipulated by the frame. Fields and their attributes are included in the form definition. Attributes include validation criteria, validation failure messages, and value "display only", and mandatory input requirements.

Below the form on the lower left of the frame is a command menu defined and placed on the frame by the procedure. The menu includes a list of operations which the user may select. By selecting a command, another procedure is called to execute the operation. The library application commands include procedures for movement to other frames, input or retrieval of data, and a few arithmetic operations.

Frames may be designated as one of four types or usages by the developer. Three are used in the library - QBF, RBF, and User Specified (User).

(1) QBF - When encountering a QBF-defined frame, ABF calls on the QBF subsystem. QBF allows append, query, and update operations to be performed on one table of the database. The developer may limit which of these operations may be performed by the frame when defining the frame. A form and one table are associated with a QBF frame. The procedure is automatically provided by QBF.

(2) Report Writer - When encountering a "report"-defined frame in the library, ABF calls on the Ingres RBF subsystem. Reports and optional forms are associated with report frames. By defining the report, the developer controls the output mode format and report inclusion and layout. The developer may provide a form in the frame definition. The form allows the user flexibility in report retrieval. The user may select records by field values or ranges.

(3) User - When ABF encounters a user-specified frame, it calls on procedures written by the developer. Procedures and forms are associated with user frames. In the library, procedures are written in either the ABF Operations Specification Language Code (OSL) or in a combination "C" language and QUEL. OSL simply calls other frames or procedures, or exits from the application. "C" and QUEL were used where such a program would be more efficient, simpler, or more secure than QBF, or where an operation required data input to or retrieval from more than one table.

E. FRAMES

The frames of the tape library are provided as charts in Appendix D. They are defined briefly in the frame description section below. The frames, and their associated forms, procedures, reports, and tables are listed and defined in greater detail in Appendix E.

F. FRAMES DESCRIPTION

1. Topframe (1)

Topframe (1) is a user frame which provides a menu form. The form spells out the commands available in the operations menu. The user may select one of six operations which call the OSL procedure. The first five call another frame - TQuery/Report (1.1), PQuery/Report (1.2), Add (1.3), Update (1.4), and Delete (1.5). The last, Exit, ends the application.

2. TQuery/Report (1.1)

PQuery (1.1) and all of its subordinate frames provide either a path to retrieval, or retrieval of a report from the database to the terminal. Terminal reports were designed for use by the librarian to search the database for a particular tape or set of tapes. The fields, formats, and sorting was selected for easy reference and relevance to the librarian. PQuery is a User frame which provides a menu form. The form defines the commands available on the operations menu. The user may select one of nine operations which call the OSL procedure. The first eight call another frame - Scratch (1.1.1), Periodic (db, wb, mb, gb) (1.1.2), Individual (is, if, iu) (1.1.3), Vendor (1.1.4), Student (1.1.5), Loaned (1.1.6), and Date to Retire (1.1.7). By selecting Return, the OSL calls Topframe (1). Exit leaves the application.

(1) Scratch (1.1.1), Periodic (1.1.2), Individual (1.1.3), Vendor (1.1.4), Student (1.1.5), Backup or Vendor (1.1.6.2), and Date to Retire (1.1.7). Each of these frames are report frames which display a parameter form. The form includes fields in which the user must enter data to define records to be included in the report. RBF provides three commands at the bottom of the form, Help, Report, and End. Help provides the user with assistance. Report enters the parameters and causes the appropriate data to be listed. End returns to the previous frame. The fields and field attributes of each form and the report definitions are provided in Appendix E under the appropriate frame definition.

(2) Loaned (1.1.6) is a User frame which provides a menu form. The form lists four commands available on the operations menu. The user may select one of the operations to call an OSL procedure. The first three call another frame - Scratch (1.1.6.1) and Backup or

Vendor (1.1.6.2). Return calls the previous frame TQuery/Report (1.1). Exit terminates the application.

(3) Scratch (1.1.6.1) is a report frame which has no associated form for parameter selection. RBF presents a command menu with the commands Help, Report, and End. Help provides assistance to the user. Report causes the loaned scratch tape report to be listed on the terminal. End calls the previous frame - Loaned (1.1.6). The report fields and format are described in Appendix E under the appropriate frame definition.

3. <u>PQuery/Report (1.2)</u>

PQuery/Report (1.2), and all of its subordinate frames, provide either a path to retrieval, or retrieve a report from the database to a file in the librarian's personal UNIX workspace. The librarian may then print the file using the UNIX operating system. The primary reason for providing printed reports (except in the case of dump reports) was to provide a means for the librarian to communicate with library users. Therefore, the fields, format, and sorting was selected to group data in a format relevant to users. Dump reports were designed to provide the librarian with a printed list of each relation in a format which would facilitate comparisons between relations. All printed reports were designed to conserve paper. Ingres documentation describes a method for printing reports directly from ABF. However, this method does not work because of a coding error in all of the current Ingres versions. It will be corrected in the upcoming Version III. Therefore, this slightly more cumbersome method of "write to file" was used. TQuery is a User frame which provides a menu form. The form spells out the operations available on the command menu. The user may select one of nine operations which call the OSL procedure. The first nine call another frame - Scratch (1.2.1), Periodic (1.2.2), Individual (1.2.3), Vendor (1.2.4), Student (1.2.5), Loaned (1.2.6), Date to Retire (1.2.7), Dump (1.2.8). Return calls the previous frame. Exit terminates the application.

(1) Scratch (1.2.1), Periodic (1.2.2), Individual (1.2.3), Vendor (1.2.4), Student (1.2.5), Backups or Vendor (1.2.6.2), and Date to Retire (1.2.7). Each of these frames are report frames which display a parameter form. The form includes fields in which the user must enter data to define the records to be included in the report. It also provides the name of the output files to the user. RBF provides three commands at the bottom of the form - Help, Report, and End. Help provides assistance to the user. Report enters the parameters, selects data, and writes the report to file. End returns to the previous frame - PQuery/Report (1.2). The reports definitions are provided in Appendix E.

(2) Loaned (1.2.6) is a User frame which provides a menu form which writes out the operations available on the

command menu. The first two call the OSL to call frame Scratch (1.2.6.1) or Backup or Vendor (1.2.6.2). Return returns to the previous frame - PQuery/Report (1.2). Exit terminates the application.

(3) Scratch (1.2.6.1) is a report frame which provides a form that communicates the name of the output file of the report to the user. RBF provides three operations on the command menu - Help, Report, Return. Help provides assistance to the user. Report writes a report of scratch tapes to the output file. End returns to the previous frame - Loaned (1.2.6)

(4) Dump (1.2.8) is a User frame which provides a menu form which writes out the operations included in the command menu. The first six call frames - Tape (1.2.8.1), Sets (1.2.8.2), File (1.2.8.3), Student (1.2.8.4), and Loan (1.2.8.5). Return calls PQuery/ Report (1.2). Exit terminates the application.

(5) Tape (1.2.8.1), Sets (1.2.8.2), File (1.2.8.3), Student (1.2.8.4), and Loan (1.2.8.5) are report frames which provide the name of the output file to the user. RBF provides the operations on the command menu - Help, Report, and Return. Help provides assistance to the user. Report selects, sorts, and writes the report to file. Return calls the previous frame - Dump (1.2.8). The report definitions are included in Appendix E.

4. <u>Add (1.3)</u>

Add (1.3), and all the frames subordinate to it, provide the means or a path to a means for the librarian to add a new record to a table or to more than one table of the database. Add is a User frame which provides a menu form that defines the operations available on the command menu. The user may select one of seven operations. All call the OSL to call a frame - Scratch (1.3.1), Vendor (1.3.2), Backup (1.3.3), Files (1.3.4), Students (1.3.5), Loan (1.3.6). Return calls the previous frame - Add (1.3).

(1) Scratch (1.3.1) is a User frame with a form and three procedures. The form provides fields for data input into one table. The first procedure is an OSL procedure which provides a command menu at the bottom of the form of three operations - Id, Add, and Return. By selecting Id, the OSL calls on procedure sgetid. The purpose of sgetid (and vgetid (1.3.2)) is to ensure empty slots of the library are filled and that new ids assigned are the maximum id plus one. New ids are assigned to the database only when scratch tapes or vendor tapes are added. Sgetid id (like vgetid id in frame 1.3.2) assigns the id labels to tapes or tapes and slots and informs the librarian of the old or new id. Sgetid places the id on the form and prompts the user to enter new data and select Add. By selecting Add, the OSL calls procedure ascratchc which

updates the fields of an empty slot record in the TAPE table or appends a new record to the TAPE table if the entered data meets the procedure and form validation criteria. It informs the user of the action taken and what labelling action the librarian should take. By selecting Return, the previous frame - Add (1.3) is called.

(2) Vendor (1.3.2) is a user frame with a form and three procedures. The form provides fields for data input to two tables. The first procedure is an OSL procedure which provides a command menu at the bottom of the form which includes three operations - Id, Add, and Return. By selecting Id, the OSL calls on procedure vgetid which retrieves the lowest empty slot, or new id number from the database and puts it on the form. It tells the user if the id is that of an empty slot or a new id and prompts the user to enter the new input and select Add. By selecting Add, the OSL calls procedure avendorc which enters the new data on the two tables if it meets the field attribute and the procedure's validation criteria. The procedure prompts the user to inform him or her of action taken. By selecting Return the previous frame - Add (1.3) is called.

(3) Backup (1.3.3) is a User frame which provides a menu form which writes out the operations provided by the

OSL procedure. The OSL procedure provides a command menu of three operations on the bottom of the form which call other frames - Periodic (1.3.3.1), Individual (1.3.3.2), and Return to the previous frame - Add (1.3.3).

(4) Periodic (1.3.3.1), Sets (1.3.3.2), New Backups or Vendor (files) (1.3.4.1), Students (1.3.5), and Loan (1.3.6), are QBF-append only frames. Each frame provides a form for data input. QBF provides a command menu at the bottom of the form which includes three operations - Help, Add, and Return. Help provides assistance to the user in the use of QBF. Add appends the data to the appropriate table. Return returns to the previous frame.

(5) Files (1.3.4) is a User frame which provides a menu form which writes out the operations provided by the OSL in the command menu. The OSL provides three operations which call frames - Periodic (1.3.3.1), Individual (1.3.3.2). Return calls the previous frame - Add (1.3).

(6) Old Partial Individual (1.3.4.2) is a User frame with a form and two procedures. The form provides fields and validation criteria for data input to two tables. The OSL procedure provides three operations on the form's command menu. Add calls procedure adfold which appends a record to the file table and updates the

sets table. Return calls the previous frame - Files (1.3.4). Exit terminates the application.

5. Update (1.4)

Update (1.4), and the frames subordinate to it, allow the librarian to update selected information in the database. The data may require update because the values changed since they were entered or because they were entered incorrectly. Update is a User frame with a menu form and an OSL procedure. The form writes out the four operations provided by the OSL in the command menu. The first four call the OSL to call other frames - Errors (1.4.1), Backup (1.4.2), and Loan (1.4.3). Return calls the previous frame - Topframe (1).

Errors (1.4.1), Backup (1.4.2), and Loan (1.4.3) are QBF update only frames. Each has one form which provides selected fields which enable update of one table by the librarian. QBF provides the command line which prompts the user for query of the old record, then update.

6. Delete (1.5)

Delete (1.5) is the only frame which allows deletion of records from the database tables. Delete is a User frame with a form and four procedures; tdelete.osl, dloanc, dbackc, and dtapec. The form provides a field and lists the operations in the command menu provided by the OSL procedure. The user enters into the field the id number of the tape for which records will be deleted. The OSL procedure provides five operations on the command menu. Loan calls procedure dloanc which deletes records from the LOAN table. Backup calls procedure dbackc which deletes records from the SETS, STUDENT, and FILE tables. Tape calls procedure dtapec which deletes records from the SETS, FILE, STUDENT, and LOAN tables and all fields from the TAPE table except id. The id number in the TAPE table will then be an empty slot.

G. DFDS/FRAMES

FUNCTION	PROCESS	FRAME
A	2, 3, 8	1.3.1
B	2, 3, 8	1.3.2
C	2	1.1.3
C	8	1.3.3.2, 1.3.4.1, 1.4.1
D	2	1.1.1
D	8	1.3.3.1, 1.3.3.2, 1.3.4.1, 1.3.5, 1.4.1
E	2	1.1.2, 1.1.3, 1.1.4, 1.1.5
E	8	1.4.1
F	2	1.1.1, 1.1.3, 1.1.4
F	8	1.3.6
G	2	1.1.6.1, 1.1.6.2
G	8	1.5
H	1	1.1.7
H	2	1.1.2, 1.1.3, 1.1.4
H	4	1.5

H. INTEGRITY

Database integrity is maintained in seven ways.

(1) Permits are defined to allow only the librarians to perform any functions on the tables of the TLS.

(2) All but two fields are limited to only one table. The exceptions are id, which is used as the link between tables, and contph. Contph is in SETS and STUDENT.

The student record is created when the student graduates. At that time, all of their files are written to the graduation backup, including any individual student backups they may have owned. All SET records regarding the student are deleted at this time (including the SET contph). Therefore when the student's contph number is entered to STUDENT, it no longer is recorded in SET.

(3) Data is validated upon input. Where allowed values were limited and known not to change frequently, validations were defined in forms. They are listed in Appendix E.

(4) "Repeat previous value facility". Librarians have many other duties which may distract them while entering data on a form. To remind them of their previous entry, they may simply press a function key for a display of previous fields entered when using QBF-append.

(5) Programming. To ensure the consecutive, sequential assignment of ids to cabinet spaces and new tapes, the assignment is performed by the application.

(6) Update. Values which change in the course of time, or values which have no validation on input may be updated by the librarian by using the update frames.

(7) The dump reports provide listings of all tables sorted by id number. The librarian may use these to compare the values recorded for each id number in each table.

I. SECURITY

As discussed in Section H, only personnel to whom permits have been granted may affect the tables of the database. Only the librarians are assigned permits. Any additional security measure is login symbols. Login symbols provide access to the application.

A login symbol is defined in the login files of each of the librarians to grant them access to the TLS. No one else may gain access.

J. DATABASE ADMINISTRATOR

The database administrator will perform the following functions to maintain the TLS.

(1) Implementation - instructions for implementation are provided in Appendix F.

 (a) Load the current library records

 (b) Change the storage structure of all tables

(2) Maintenance - Instructions for maintenance of the library are included in the tutorial in Appendix G

 (a) To maintain peak performance, system modifications must be run on each table when 20 percent of the records in the table have been changed. Monthly system modifications are recommended in the tutorial.

 (b) Maintain the integrity of the library by destroying and defining permits to tables as personnel are relieved of their librarian duties and new librarians are assigned.

 (c) Define the TLS login symbol in the login files of new librarians assigned. Delete the login sign from the login files of personnel relieved of librarian duties.

K. VALIDATION AND TEST

Frames were developed in a top-down fashion, Topframe and each of the menu frames first. Append frames were then developed and tested. Records of 34 tapes, backup data, and associated data, were entered into the library using the new frames. The update frames were then tested. The librarian verified the functionality of the frames and appropriateness of the fields. The terminal report frames were then developed and tested followed by the printed report frames. The librarian verified the reports' inclusion and format. All frames were tested together using the small, representative library data and a scenario of the various processes performed in the library.

Various bugs in the Ingres DBMS were discovered during testing:

(1) The Ingres documentation does not describe how to select records from one table which are not in another table. This was resolved by an Ingres representative.

(2) The VM UNIX Version 2.1/15VE.04 interface between ABF and QBF malfunctions. A corrected version has been sent to NPGS.

  (a) In QBF-update mode, updates were not performed.

  (b) In QBF - all modes, validations could not be performed between sets.

(3) The Ingres documentation does not describe how to interface parameter forms with RBF reports. An Ingres representative explained the procedure.

(4) A known bug exists in the RBF facility. The documentation describes how to flag Ingres to output reports directly to a printer. The process is coded incorrectly and will not be corrected until the next

version of Ingres, Version III is developed. Therefore, the TLS outputs reports to be printed to the user's UNIX workspace. The report must then be printed from UNIX.

V. RECOMMENDATIONS AND CONCLUSIONS

Recommendations regarding the use of the TLS, observations regarding Ingres, and an opinion on use of the requirements analysis and design process are presented.

A. THE LIBRARY

The tape library system (TLS) is designed to enable the Computer Science Department to provide secure, dependable library services. They may do so by implementing the TLS described in this thesis, and by locking the cabinets in which they keep the tapes. There are ways, however, in which data can be entered incorrectly. They exist because: 1) either there is no means of validating the entry; 2) the likelihood of making the error can be reduced by methods other than validation; or 3) the potential error is not critical or can be easily recognized and corrected by the librarian.

(1) Ingres forms allow validations which refer to other fields in the same or other tables. However, these values are recorded for comparison at the time the form is initialized. Forms are initialized when QBF is called. Therefore, when adding records to a set, the values added after QBF was called will not be included in the validation. Tables which get new

values through QBF-append (SETS, FILE, LOAN, STUDENT) are vulnerable to this lack of validation. Double entries of id numbers or entry of id numbers which are not logged in the library can occur. This likelihood is kept low because of the two procedures provided in a. and b. below. If either error occurs, it may be caught and corrected by c.

(a) In order to write a backup the librarian must get a scratch tape and an id number. They must be found in the scratch tape list, there is no other place to find available tapes or id numbers. The scratch tape list is created and maintained by the application and has very tight integrity.

(b) If the librarian is entering records into the tables, and forgets the last id number entered, a control function may be used to see which was the last id number entered.

(c) If an id is listed in SETS, FILES, STUDENT, or LOAN which was not assigned in the library or which duplicated a previously assigned number it will be discovered when a routine backup or vendor report is run, or when the librarian runs a dump of the table.

(2) Librarians occassionally enter the wrong dates, especially the year. Ingres does not allow arithmetic operations on dates in QBF validation. Therefore, the application will not validate date input. However, current date entry errors are quickly discovered in the library. In the TLS they may be corrected using the update frames.

(3) When reports are retrieved by selecting parameters by value, the value must be exact (or * for all or any

letter or any word). Upper case letters are considered different than lower case. It is recommended that all entries be in lower case letters and that the librarians standardize input as recommended in the data dictionaries (Appendices A and C). Where likely discrepancies occur, the * symbol may be used within a word to mean "any letters," or by itself to mean "any value."

B. INGRES

Various other strengths and weaknesses of Ingres were learned during the study.

Ingres promotes the use of QBF, RBF, and ABF in its documentation. RBF and ABF are efficient and comprehensive for use for the tape library and much more complicated applications; QBF is not. When a QBF form is called from ABF, at least 15 seconds pass before a form appears on the screen. This time, and the limitations on validations available, are burdensome to the user. Input time is faster than it would be by use of a procedure, but for simple appends and updates, it dosen't save enough time to compensate. RBF takes as long to produce a report, which is acceptable considering the data retrieval, selection, and sorting process involved. ABF is a powerful and convenient tool for building and testing applications.

Ingres documentation is incomplete. As discussed in

Chapter IV, certain procedures were omitted.

Various bugs were revealed in Ingres. This was the developers first intimate experience with a commercial software product. This experience and discussions with more experienced developers indicates software, at any price, is always in the test and development stages. It is recommended that the CS staff maintain close liaison with the helpful hotline at RTI to improve the NPGS purchase and the RTI product.

C. REQUIREMENTS ANALYSIS AND DESIGN

A requirements analysis and design process similar to those described by Kroenke [Ref. 3] and DeMarco [Ref. 4] was implemented. Although there were times when it seemed the process was in the way of progress, it was followed diligently. As has been written by Boehm [Ref. 5], and many others, time consuming requirements result in fewer surprises during the design process. By the time the design stage was started, both the library and Ingres were well known. Only Ingres bugs or deficient documentation caused surprises during design and implementation.

APPENDIX A

SPECIFICATION DATA DICTIONARY

A. DATA CLASS: SLOT (included in Tapes)*

<u>Definition</u>	A slot in a tape cabinet. Only one tape is stored in a slot. Each slot in a cabinet is identified and labelled with an identification number. Slots are labelled consecutively left to right, top to bottom

<u>Elements</u>

Identification	A unique number assigned to a slot
Attribute	id**
Type	Type is an attribute of tape. Therefore, if no tape is assigned to a slot, the TAPE table will have a null entry in all tape attributes except "id".

* If all attributes other than "id" in table TAPE have null values, the record represents an empty slot. If the id and any other attributes in table TAPE have values, the slot id number is the same as the tape id number.

** Key

B. DATA CLASS: TAPE

Definition	A 7/9 track reel of magnetic tape up to 3600 feet long, to be used or being used to store data useful to the CS department. Attribute values impact the librarian's decision of whether or not to use the tape for a specific "read to tape" request.

Elements

Identification	A unique identifier of a tape. The same as the identification number of the slot to which it is assigned
Attribute	id
Constraint	unique, key, mandatory
Length	The length of the tape. Tapes are normally 300, 350, 400, 450, 550, 600, 650, 1200, 2400, or 3600 feet long
Attribute	length
Constraint	mandatory
Density	The bits per inch the tape may copy. Densities are 800, 1600, 6250,a combination 1600/6250, or 10,000. Combinations shall be recorded as 6250
Attribute	bpi
Constraint	mandatory
Errors	The number of bad bits on the tape. The utility provides the number when a tape is read or written. Used to determine the condition of the tape. If over 10, the librarian will run it again when the heads are freshly changed to get a more accurate reading. If the figure remains greater than 10, it is considered unacceptable to use to read certain types of backups.
Attribute	err
Date of Errors	The date the errors were last checked
Attribute	derr

C. DATA CLASS: SET

Definition	A conceptual entity which is one or more tapes of recorded data (may be called an active tape) which, together, are identified as a whole.

Elements

Type	The category of the active tape(s) by virtue of what is recorded on it. Types are: daily system backup (db) weekly system backup (wb) monthly system backup (mb) graduated student's backup (gb) individually owned-staff (is) individually owned-faculty backup (if) individually owned-student backup (iu) vendor software (v)
Attribute	type
Constraints	key, mandatory if w or v, value must = db, wb, mb, gb, is, if, iu, or v
System	A code representing a combination hardware and operating system from which the tape was read. Current codes are: iris1 scald2 iris2 sun1 microvax1 sun2 microvax2 unix1 pdp1 vms1 pdp2 vms2 scald1
Attribute	system
Constraints	key, mandatory
Utility	The software utility which was, and must be used to read or write the set of tapes. Utilities currently used by the department are: /ingres /usrc /nps$user1 /vlsl /nps$user2 /vmssys /npsuser3 /work /user

Attribute	utility
Constraints	mandatory
Density	See "Data Class: TAPE"
Attribute	bpi
Constraints	must be the same value as Data Class: TAPE
Date Created	The date the first bit of the active set was read to tape. Used to determine the age of the bits and the period of time the tape represents. The date to be recorded varies according to the type of the set.
Attribute	dcreate
Constraints	key, mandatory
	value for each tape is limited to:
	db, wb, mb, is, if, iu = date read to tape
	gb = the date the class graduated
	v = the first day of the month the tape was received by the owner or department
Retirement Date	The date the active set should be converted to scratch. (In the case of individual sets, the owner's permission must be obtained before conversion.) Varying according to type.
Attribute	dretire
Constraints	mandatory
	value for each type limited to:
	db = dcreate + 1 month
	wb = dcreate + 3 months
	mb = dcreate + 6 months
	gb = dcreate + 2 years
	v = dcreate + 5 years
	is = dcreate + 2 years
	if = dcreate + 2 years
	iu = dcreate + 2 years
Description	Narrative comments about the set
Attribute	descr
Constraints	none

Name of Owner	The name of the owner of the tape(s). Used only in individual or vendor set types.
Attribute	owner
Constraints	is, if, iu = mandatory and key v = optiional db, wb, mb, gb, = null
Advisor	The name of the student's advisor. Used only in individual student set type.
Attribute	advisor
Constraints	u, i = optional db, wb, mb, gb, is, if, v = null
Contact Phone	A phone number. Only used in individual or vendor set types. The person whose number is recorded varies according to set type.
Attributes	contph
Constraints	is, if, v = owner's phone number iu = advisor's phone number
Software	The name of the software on the tape, including the version number. Only used for set type vendor
Attribute	software
Constraints	v = mandatory, key db, wb, mb, gb, is, if, iu = null
Vendor	The name of the vendor company which produced the software. Used only for set type vendor
Attribute	vendor
Constraints	v = optional db, wb, mb, gb, is, if, iu = null

D. DATA CLASS: ACTIVE TAPE

Definition	A conceptual entity which is one tape which, by virtue of its recorded data is a member of a set. Attributes are dependent upon the fact it has recorded data and is a member of an active set.

Element

Identification	A unique identifier of a reel, the same as the identification number of the slot to which it is assigned
Attribute	id
Constraint	unique, key, mandatory
Label	The VMS operating system requires that each tape be assigned a label by the librarian. When the tape is mounted, the label verifies to the system that the tape is for VMS read/ write. If the "copy" utility is used to read or write the tape, a label must be entered by the librarian when mounting the tape. It is not necessary for the librarian to know the label name when using the other VMS utility-"backup". The librarian choses a label name which indicates what is recorded on the tape.
Attribute	label

Constraint	if (utility = copy) mandatory
	if (utility = backup) optional

Sequence	The sequential order of the tape in the total number of tapes which make up the set
Attribute	seq
Constraint	if number of tapes in the parent set > 1, then mandatory
Percent Used	The percent of the tape which contained. Data used for tapes which are members of set type individual only; and only tape is partially used.
Attribute	perusd
Constraint	if (is. if, iu) and partially used, then optional

E. DATA CLASS: FILE PATH

Definition	A conceptual entity which provides the path to the disk location of one or more files. Inherent in the file path name construed by the librarian as the file content.

Elements

Partition	A major subdivision of files in the UNIX operating system. The partition from which the files were read
Attribute	part

Constraint	if system = UNIX* then mandatory, key
	if system = UNIX* then null

Disk	A major subdivision by disk in the VMS operating system. The disk from which the files are read
Attribute	disk

Constraint	if system = VMS* then mandatory, key
	if system = UNIX** then null

Saveset	When using VMS "backup" utility, marks the beginning of each file or set of files. Terminates with an end-of-file marker. When using the "backup" utility for read/write, the saveset must be entered by the librarian in order to mount the tape. Librarians assign saveset names which indicate what is in the file
Attribute	sav
Constraint	if utility = backup then mandatory, key
Source	Systems other than VMS and utilities other than backup and copy do not require the assignment of label or saveset names. The librarian has contrived a "source" to be assigned to a file or set of files on a tape which is read by other than VMS backup and copy utilities. The assigned name indicates the file(s) contents
Attribute	src
Constraint	if utility = backup or copy and set type = gb, then mandatory if utility = backup or copy and set type = gb, then optional

F. DATA CLASS: STUDENT

Description	The name given to a person who has graduated from NPGS and whose files are contained in a set of graduation tapes

Elements

Name	The last and first name of the student
Attribute	name
Constraint	required key
Advisor	The name of the student's advisor
Attribute	advisor
Constraint	mandatory
Contact Phone	The telephone number of the advisor
Attribute	contph

G. DATA CLASS: LOAN

Definition	A conceptual entity which describes the agreement under which a tape is loaned.

Element

Loanee	The name of the person to whom the tape was loaned
Attribute	loanto
Constraint	mandatory key
Date Loaned	The date the tape is loaned
Attribute	dloan
Constraint	none
Estimated Date of Return	The date the tape should be returned to the the library
Attribute	edr
Constraint	edr > dloan
Contact Phone	The phone number of the loanee
Attribute	lcntph
Constraint	none

APPENDIX B

TABLES AND VIEWS

A. TABLES [Ref. 4]

File, Loan, Sets, Student, Tape

```
Table:                File
Owner:                crawford
Row Width:            26
Saved Until:           1-Jan-1999 00:00:00
Number of Rows:       21
Number of Pages:      1
Journaling:           disabled
Storage Structure:    heap
Table Type:           user table

column name   type   length      column name   type   length

id            i           2      part_disk     c           9
sav_src       c          15      err           i           1
```

```
Table:                Loan
Owner:                crawford
Row Width:            52
Saved Until:           1-Jan-1999 00:00:00
Number of Rows:       3
Number of Pages:      1
Journaling:           disabled
Storage Structure:    heap
Table Type:           user table

column name   type   length      column name   type   length

id            i           2      dloan         date       12
edr           date       12      loanto        c          15
lcntph        c          11
```

```
Table:                  Sets
Owner:                  crawford
Row Width:              126
Saved Until:             1-Jan-1999 00:00:00
Number of Rows:         12
Number of Pages:        1
Journaling:             disabled
Storage Structure:      heap
Table Type:             user table


column name    type   length        column name    type   length

id             i           2        type           c           2
dcreate        date       12        dretire        date       12
system         c           9        utility        c           6
owner          c           9        advisor        c           9
contph         c           4        software       c          15
vendor         c          20        label          c           6
descr          c          15        sequ           f           4
perusd         i           1
```

```
Table:                  Student
Owner:                  crawford
Row Width:              30
Saved Until:             1-Jan-1999 00:00:00
Number of Rows:         4
Number of Pages:        1
Journaling:             disabled
Storage Structure:      heap
Table Type:             user table

column name    type   length        column name    type   length

id             i           2        name           c          15
advisor        c           9        contph         c           4
```

```
Table:                  Tape
Owner:                  crawford
Row Width:              19
Saved Until:             1-Jan-1999 00:00:00
Number of Rows:         18
Number of Pages:        1
Journaling:             disabled
Storage Structure:      heap
Table Type:             user table


column name    type   length      column name    type   length

id             i           2      length         i           2
bpi            i           2      err            i           1
derr           date       12
```

B. VIEWS

vind, vloan, vpe, vretire, vscloan, vscratch, vstudent, vven

<u>VIEW VIND DEFINED</u>

```
range of t is tape
range of s is sets
range of f is file
define view vind (
        id = s.id,
        type = s.type,
        owner = s.owner,
        advisor = s.advisor
        contph = s.contph
        dcreate = s.dcreate,
        system = s.system,
        utility = s.utility,
        bpi = t.bpi,
        label = s.label,
        part_disk = f.part_disk,
        sav_src = f.sav_src,
        descr = s.descr,
        err = t.err,
        length = t.length,
        perusd = s.perusd,
        sequ = s.sequ)
        where (s.id = f.id)
        and   (t.id = s.id)
        and   (s.type = "i*")
```

VIEW VLOAN DEFINED

```
range of t is tape
range of s is sets
range of l is loan
define view vloan (
        id = t.id,
        type = s.type,
        system = s.system,
        eequ = s.sequ,
        owner = s.owner,
        software = s.software,
        dloan = l.dloan,
        loanto = l.loanto,
        lcntph = l.lcntph,
        edr = l.edr)
        where (t.id = s.id)
        and   (l.id = t.id)
```

VIEW VPE DEFINED

```
range of s is sets
range of t is tape
range of f is file
define view vpe (
        id = s.id,
        type = s.type,
        dcreate = s.dcreate,
        system = s.system,
        utility = s.utility,
        bpi = t.bpi,
        label = s.label,
        part_disk = f.part_disk,
        sav_src = f.sav_src,
        descr = s.descr,
        err = t.err,
        sequ = s.sequ)
        where (t.id = s.id)
        and   (s.id = f.id)
        and   (s.type = "*b")
```

VIEW VRETIRE DEFINED

```
range of t is tape
range of s is sets
define view vretire (
        dretire = s.dretire,
        type = s.type,
        dcreate = s.dcreate,
        system = s.system,
        sequ = s.sequ,
        id = s.id,
        length = t.length,
        bpi = t.bpi,
        err = t.err)
        where (t.id = s.id)
```

VIEW VSCLOAN DEFINED

```
range of t is tape
range of s is sets
range of l is loan
define view vscloan (
        id = lid,
        edr = l.edr,
        dloan = l.dloan,
        loanto = l.loanto,
        lcntph = l.lcntph)
        where (t.bpi != 0)
        and   (t.length != 0)
        and   (any(s.id by t.id
        where (t.id = s.id)) = 0)
        and   (t.id = l.id)
```

VIEW VSCRATCH DEFINED

```
range of t is tape
range of s is sets
define view vscratch (
        id = t.id,
        length = t.length,
        bpi = t.bpi,
        err = t.err,
        derr = t.derr)
        where (t.length != 0)
        and   (any(s.id by t.id
        where (t.id = s.id)) = 0)
        and   (t.bpi != 0)
```

VIEW VSTUDENT DEFINED

```
range of t is tape
range of s is sets
range of rv2 is student
range of f is file
define view vstudent (
        name = rv2.name,
        dcreate = s.dcreate,
        id = s.id,
        system = s.system,
        utility = s.utility,
        bpi = t.bpi,
        label = s.label,
        part_disk = f.part_disk,
        sav_src = f.sav_src,
        descr = s.descr,
        sequ = s.sequ,
        advisor = rv2.advisor,
        contph = rv2.contph)
        where (f.id = rv2.id)
        and   (s.id = f.id)
        and   (t.id = s.id)
        and   (s.type = "gb")
```

VIEW VVEN DEFINED

```
range of t is tape
range of s is sets
range of f is file
define view vven (
        id = s.id,
        software = s.software,
        vendor = s.vendor,
        system = s.system,
        utility = s.utility,
        bpi = t.bpi,
        label = s.label,
        part_disk = f.part_disk,
        sav_src = f.sav_src,
        descr = s.descr,
        owner = s.owner,
        advisor = s.advisor
        contph = s.contph
        err = t.err,
        sequ = s.sequ)
        where (s.id = f.id)
        and   (t.id = s.id)
        and   (s.type = "v")
```

APPENDIX C

DESIGN DATA DICTIONARY

In Appendix A, the parameters of integers and floating point numbers were expressed in bytes, as Ingres does. The parameters of these data types are expressed in bits. Read "i2" as "integer, two digits." Read f1.1 as "one digit decimal one digit."

Field:	advisor			
Name:	advisor			
Data Class:	set			
Definition:	The name of the student's advisor.			
Parameters:	c9			
Constraints:	First eight letters of last name, first initial of first name; no spaces or commas.			
Where Used:	Source Input:	<u>Table</u>	<u>Frame</u>	
		sets	1.3.2, 1.3.3.2	
		student	1.3.5	
	Maintenance:	<u>Table</u>	<u>Frame</u>	<u>Action</u>
		sets	1.4.2	update
		student	1.5	delete
		sets	1.5	delete
	Other:	1.1.3, 1.1.4, 1.1.5, 1.2.3, 1.2.4, 1.2.5, 1.2.8.2, 1.2.8.4		
Storage:	sets, student			
View:	vstudent			

Field:	bpi
Name:	density
Data Class:	tape
Definition:	The bits per inch capacity of the tape.
Parameters:	i4
Constraints:	Valid values: [800, 1600, 6250, 10,000] Mandatory

Where Used:	Source Input:	Table	Frame	
		tape	1.3.1, 1.3.2	
	Maintenance:	Table	Frame	Action
		tape	1.5	delete
	Other:	1.1.1, 1.1.2, 1.1:3, 1.1.4, 1.1.5, 1.1.7, 1.2.1, 1.2.2, 1.2.3, 1.2.4, 1.2.5, 1.2.7, 1.2.8.1		

Storage:	tape
View:	vind, vpe, vretire, vscratch, vstudent, vven

Field:	contph
Name:	contact phone
Data Class:	set
Definition:	If field advisor = null then the phone number of the owner If field advisor != null then the phone number of the advisor
Parameters:	i4
Constraints:	Last four digits of the telephone number Mandatory

Where Used:

Source Input:

Table	Frame
sets	1.3.2, 1.3.3.2
student	1.3.5

Maintenance:

Table	Frame	Action
sets	1.4.2	update
student	1.5	delete

Other: 1.1.3, 1.1.4, 1.1.5, 1.2.3, 1.2.4, 1.2.5, 1.2.8.2, 1.2.8.4

Storage:	sets
View:	vstudent, vind

Field:	dcreate
Name:	date created
Data Class:	set

Definition: If type = db, wb, mb, is, if, iu then the date the backup was created
If type = gb then the date the class graduated
If type = v then the date the tape was received by NPGS

Parameters: c8

Constraints: mm/dd/yr (last two digits)
Mandatory

Where Used:

Source Input:	Table	Frame	
	sets	1.3.2, 1.3.3.1, 1.3.3.2	
Maintenance:	Table	Frame	Action
		1.4.2	update
		1.5	delete
Other:	1.1.2, 1.1.3, 1.1.5, 1.1.7, 1.2.2, 1.2.3, 1.2.5, 1.2.8.2		

Storage:	sets
View:	vind, vpe, vretire, vstudent

Field:	derr			
Name:	date of errors			
Data Class:	tape			
Definition:	The date the errors were last checked.			
Parameters:	c8			
Constraints:	mm/dd/yr (last two digits)			
Where Used:	Source Input:	Table	Frame	
		tape	1.4.1	
	Maintenance:	Table	Frame	Action
		tape	1.4.1	update
		tape	1.5	delete
	Other:	1.1.1, 1.2.1, 1.2.8.1		
Storage:	tape			
View:	vscratch			

Field:	desc			
Name:	description			
Data Class:	set			
Definition:	Librarian's notes about the set.			
Parameters:	c15			
Constraints:	none			
Where Used:	Source Input:	<u>Table</u>	<u>Frame</u>	
		sets	1.3.2, 1.3.3.1, 1.3.3.2	
	Maintenance:	<u>Table</u>	<u>Frame</u>	<u>Action</u>
			1.4.2	update
			1.5	delete
	Other:	1.1.2, 1.1.3, 1.1.4, 1.1.5, 1.2.2, 1.2.3, 1.2.4, 1.2.5, 1.2.8.2		
Storage:	sets			
View:	vind, vpe, vstudent, vven			

Field:	dloan			
Name:	date loaned			
Data Class:	loan			
Definition:	The date a tape is loaned.			
Parameters:	c8			
Constraints:	mm/dd/yr (last two digits) Mandatory			
Where Used:	Source Input:	Table	Frame	
		loan	1.3.6	
	Maintenance:	Table	Frame	Action
			1.5	delete
	Other:	1.1.6.1, 1.1.6.2, 1.2.6.1, 1.2.6.2, 1.2.8.5		
Storage:	loan			
Views:	vloan, vscloan			

Field:	dretire
Name:	retirement date
Data Class:	set

Definition: If type = db, wb, mb, gb, is, if, iu, then the date the backup should be converted to scratch
If type = v then the date the vendor tape should be replaced due to old bit age
If type = is, if, iu, then permission must be obtained from the owner before conversion
Recommended values by type:

db = dcreate + 1 month
wb = dcreate + 3 months
mb = dcreate + 6 months
gb = dcreate + 2 years
v = dcreate + 5 years
is, if, iu = dcreate + 2 years

Parameters: c8

Constraints: Mandatory

Where Used:

Source Input:	Table	Module
	sets	1.3.2, 1.3.3.1, 1.3.3.2

Maintenance:	Table	Module	Action
		1.4.2	update
		1.5	delete

Other: 1.1.7, 1.2.7, 1.2.8.2

Storage: sets
View: vretire

Field:	edr			
Name:	estimated date of return			
Data Class:	loan			
Definition:	The date the librarian assigns for a loaned tape to be returned to the library.			
Parameters:	c8			
Constraints:	mm/dd/yr (last two digits)			
Where Used:	Source Input:	<u>Table</u>	<u>Frame</u>	
		loan	1.3.6	
	Maintenance:	<u>Table</u>	<u>Frame</u>	<u>Action</u>
			1.4.3	update
			1.5	delete
	Other:	1.1.6.1, 1.1.6.2, 1.2.6.1, 1.2.8.5		
Storage:	loan			
View:	vloan, vscloan			

Field:	err			
Name:	errors			
Data Class:	tape			
Definition:	The number of bad bits on a tape.			
Parameters:	i2			
Constraints:	1-99			
Where Used:	Source Input:	Table	Frame	
		tape	1.4.1	
	Maintenance:	Table	Frame	Action
		tape	1.4.1	update
		tape	1.5	delete
	Other:	1.1.1, 1.1.2, 1.1.3, 1.1.4, 1.1.7, 1.2.1, 1.2.2, 1.2.3, 1.2.4, 1.2.7, 1.2.8.1		
Storage:	tape			
View:	vind, vpe, vretire, vscratch, vven			

Field:	id
Name:	identification
Data Class:	tape

Definition: The unique number assigned to all cabinet slots and tapes; key to all tables.

Parameters: i5

Constraints: 1-10000
Mandatory input in all tables

Where Used: Source Input:

Table	Frame
tape	1.3.1, 1.3.2,
sets	1.3.3.1, 1.3.3.2
file	1.3.4.1, 1.3.4.2
student	1.3.5
loan	1.3.6 .

Maintenance:

Table	Frame	Action
sets	1.5	delete
file	1.5	delete
student	1.5	delete
loan	1.5	delete

Other: 1.1.1, 1.1.2, 1.1.3, 1.1.4, 1.1.5, 1.1.6.1, 1.1.6.2, 1.1.7, 1.2.1, 1.2.2, 1.2.3, 1.2.4, 1.2.5, 1.2.6.1, 1.2.6.2, 1.2.7, 1.2.8.1, 1.2.8.2, 1.2.8.3, 1.2.8.4, 1.2.8.5, 1.4.1, 1.4.2, 1.4.3

Storage: tape, sets, file, student, loan
Views: vind, vloan, vpe, vretire, vscloan, vscratch, vstudent, vven

Field:	label			
Name:	label			
Data Class:	active tape			
Definition:	Required by the VMS operating system. The name of a tape, assigned by the librarian.			
Parameters:	c6			
Constraints:	if system = VMS then Mandatory			
Where Used:	Source Input:	<u>Table</u>	<u>Frame</u>	
		sets	1.3.2, 1.3.3.1, 1.3.3.2	
	Maintenance:	<u>Table</u>	<u>Frame</u>	<u>Action</u>
			1.5	delete
	Other:	1.1.2, 1.1.3, 1.1.4, 1.1.5, 1.2.2, 1.2.4, 1.2.5, 1.2.8.2		
Storage:	sets			
Views:	vind, vpe, vstudent, vven			

Field:	lcntph			
Name:	loanee contact phone			
Data Class:	loan			
Description:	Phone number of the person to whom a tape is loaned.			
Parameters:	c11			
Constraints:	Three numbers of area code, space, seven numbers of phone number (no hyphen) Mandatory			
Where Used:	Source Input:	Table	Frame	
		loan	1.3.6	
	Maintenance:	Table	Frame	Action
			1.4.3	update
			1.5	delete
	Other:	1.1.6.1, 1.1.6.2, 1.2.6.1, 1.2.6.2, 1.2.8.5		
Storage:	loan			
Views:	vloan, vscloan			

Field:	length
Name:	length
Data Class:	tape
Definition:	The length in feet of a magnetic tape.
Parameters:	i4
Constraints:	Valid Value: [300, 400, 450, 555, 600, 650, 1200, 2400, or 3600] Mandatory

Where Used:			
Source Input:	Table	Frame	
	tape	1.3.1	
Maintenance:	Table	Frame	Action
		1.5	delete
Other:	1.1.1, 1.1.3, 1.1.7, 1.2.1, 1.2.3, 1.2.7, 1.2.8.1		

Storage:	tape
Views:	vind, vretire, vscratch

Field:	loanto
Name	loanee
Data Class:	loan
Definition:	The name of the person to whom a tape is loaned.
Parameters:	c15
Constraints:	Last name, comma, as much of first name that will fit Mandatory

Where Used:	Source Input:	Table	Frame	
		loan	1.3.6	
	Maintenance:	Table	Frame	Action
			1.4.3	update
			1.5	delete
	Other:	1.1.6.1, 1.1.6.2, 1.2.6.1, 1.2.6.2, 1.2.8.5		

Storage:	loan
Views:	vloan, vscloan

Field:	name			
Name:	name			
Data Class:	student			
Definition:	The name of a student who has graduated from NPGS and whose files are maintained on a graduation backup tape.			
Parameters:	c15			
Constraints:	Last name, comma, as much of first name that will fit Mandatory			
Where Used:	Source Input:	<u>Table</u>	<u>Frame</u>	
		student	1.3.5	
	Maintenance:	<u>Table</u>	<u>Frame</u>	<u>Action</u>
			1.5	delete
	Other:	1.1.5, 1.2.5, 1.2.8.4		
Storage:	student			
View:	vstudent			

Field:	owner
Name:	owner
Data Class:	set
Definition:	The name of owner of the data on the tape.
Parameters:	c9
Constraints:	First eight letters of first name, first initial of last name; no spaces or commas Mandatory if type = individual

Where Used:

	Table	Frame	Action
Source Input:	sets	1.3.2, 1.3.3.2	
Maintenance:		1.4.2	update
		1.5	delete
Other:	1.1.3, 1.1.4, 1.1.6.2, 1.2.3, 1.2.4, 1.2.6.2, 1.2.8.2		

Storage:	sets
Views:	vind, vloan, vven

Field:	part_disk			
Name:	partition or disk			
Data Class:	file path			
Description:	Partition is a major subdivision of files in the UNIX operating system. Disk is a major subdivision of files by disk in the VMS operating system.			
Parameters:	c9			
Constraints:	Mandatory			
Where Used:	Source Input:	Table	Frame	
		file	1.3.2, 1.3.4.1, 1.3.4.2	
	Maintenance	Table	Frame	Action
			1.5	delete
	Other:	1.1.2, 1.1.3, 1.1.4, 1.1.5, 1.2.2, 1.2.3, 1.2.4, 1.2.5, 1.2.8.3		
Storage:	file			
Views:	vind, vpe, vstudent, vven			

Field:	perusd			
Name:	percent used			
Data Class:	active tape			
Definition:	If type = is, if, or iu then the percent used of the last tape of a sequence of tapes which make up a set. Use only if percent is less than 100.			
Parameters:	i2			
Constraints:	1-99			
Where Used	Source Input:	<u>Table</u>	<u>Frame</u>	
		sets	1.3.3.2, 1.3.4.2	
	Maintenance:	<u>Table</u>	<u>Frame</u>	<u>Action</u>
			1.5	delete
	Other:	1.1.3, 1.2.3, 1.2.8.2		
Storage:	sets			
View:	vind			

Field:	sav_src			
Name:	savset or source			
Data Class:	file path			
Definition:	If VMS backup utility then marker for beginning of each set of files. If not VMS backup utility, then librarians name of a set of files			
Parameters:	c15			
Constraints:	Mandatory			
Where Used:	Source Input:	Table	Frame	
		file	1.3.2, 1.3.4.1, 1.3.4.2	
	Maintenance:	Table	Frame	Action
			1.5	delete
	Other:	1.1.2, 1.1.3, 1.1.4, 1.1.5, 1.2.2, 1.2.3, 1.2.4, 1.2.5, 1.2.8.3		
Storage:	file			
Views:	vind, vpe, vstudent, vven			

Field:	sequ
Name:	sequence
Data Class:	active tape
Definition:	The sequential number of a tape in the total number of tapes which make up a set (read "1.4" as "one of four").
Parameters:	f1.1
Constraints:	Express as sequence number decimal point total number (e.g. 1.4)

Where Used:

Source Input:	Table	Frame
	sets	1.3.3.1, 1.3.3.2

Maintenance:	Table	Frame	Action
		1.4.2	update
		1.5	delete

Other: 1.1.2, 1.1.3, 1.1.4, 1.1.5, 1.1.6.2, 1.1.7, 1.2.2, 1.2.3, 1.2.4, 1.2.5, 1.2.6.2, 1.2.7, 1.2.8.2, 1.3.2

Storage:	sets
Views:	vind, vloan, vpe, vretire, vstudent, vven

Field:	software			
Name:	software			
Data Class:	sets			
Definition:	The name of the software of a vendor tape.			
Parameters:	c15			
Constraints:	Include version number after name. If the name is too long leave off enough letters to include version number as last entry. Mandatory			
Where Used:	Source Input:	Table	Frame	
		sets	1.3.2	
	Maintenance:	Table	Frame	Action
			1.5	delete
	Other:	1.1.4, 1.1.6.2, 1.2.4, 1.2.6.2, 1.2.8.2		
Storage:	sets			
Views:	vloan, vven			

Field:	system			
Name:	system			
Data Class:	set			
Definition:	Combination hardware and operating system code.			
Parameters:	c9			
Constraints:	Valid Values:	iris1	nbi3	pdp1
		iris2	nbi4	pdp2
		microvax1	nbi5	unix1
		microvax2	nbi6	vms1
		nbi1	nbi7	vms2
		nbi2	nbi8	
	Mandatory			
Where Used:	Source Input:	<u>Table</u>	<u>Frame</u>	
		sets	1.3.2, 1.3.3.1, 1.3.3.2	
	Maintenance:	<u>Table</u>	<u>Frame</u>	<u>Action</u>
			1.5	delete
	Other:	1.1.2, 1.1.3, 1.1.4, 1.1.5, 1.1.6.2, 1.1.7, 1.2.2, 1.2.3, 1.2.4, 1.2.5, 1.2.6.2, 1.2.7, 1.2.8.2		
Storage:	sets			
Views:	vind, vloan, vpe, vretire, vstudent, vven			

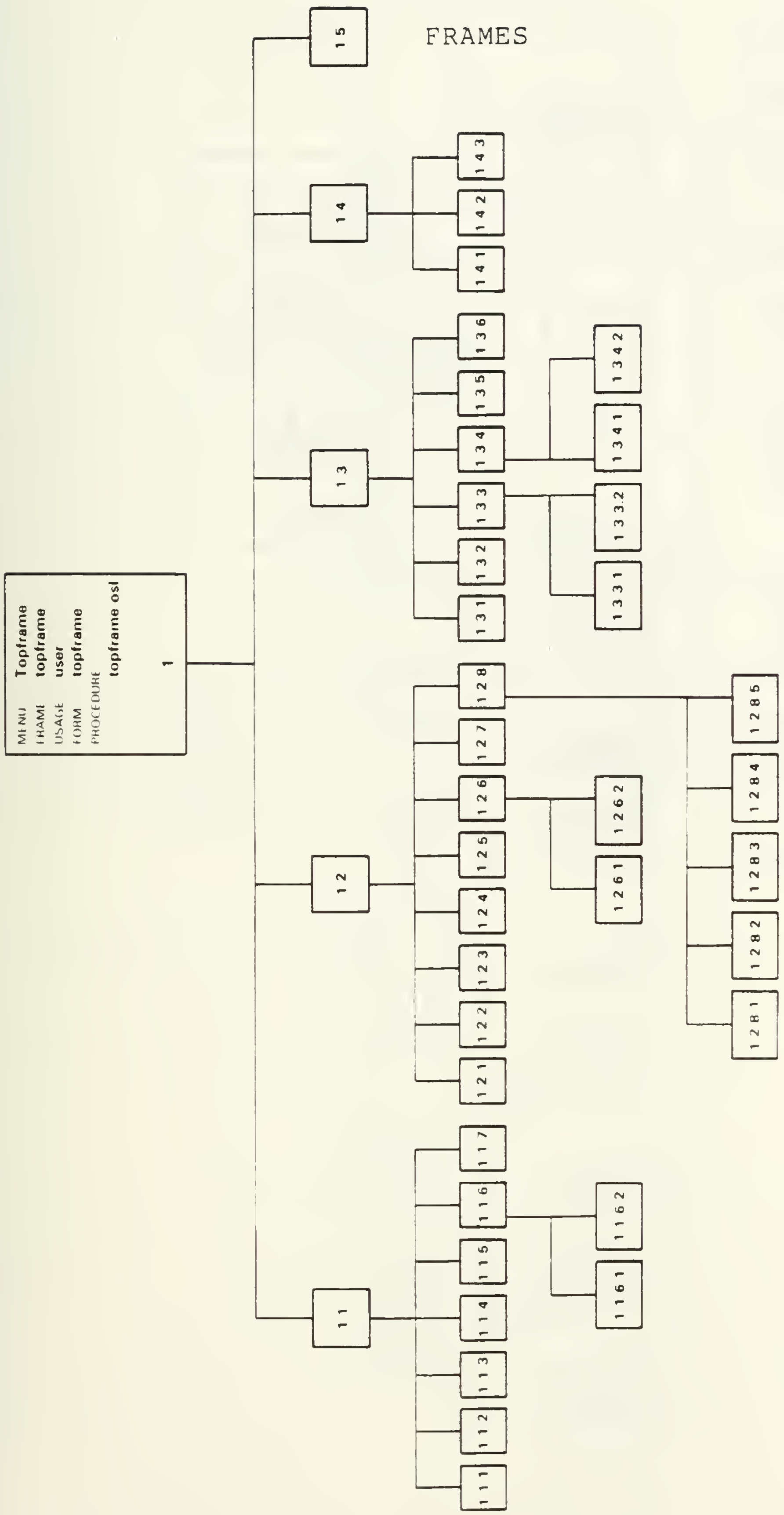
Field:	type			
Name:	type			
Data Class:	set			
Definition:	The category of active tape by virtue of what is recorded on it.			
Parameter:	c2			
Constraints:	db, wb, mb, gb, is, if, iu, v Mandatory			
Where Used:	Source Input:	Table	Frame	
		sets	1.3.2, 1.3.3.1, 1.3.3.2	
	Maintenance:	Table	Frame	Action
			1.5	delete
	Other:	1.1.2, 1.1.3, 1.1.6.2, 1.1.7, 1.2.2, 1.2.3, 1.2.6.2, 1.2.7, 1.2.8.2		
Storage:	sets			
Views:	vind, vloan, vpe, vretire			

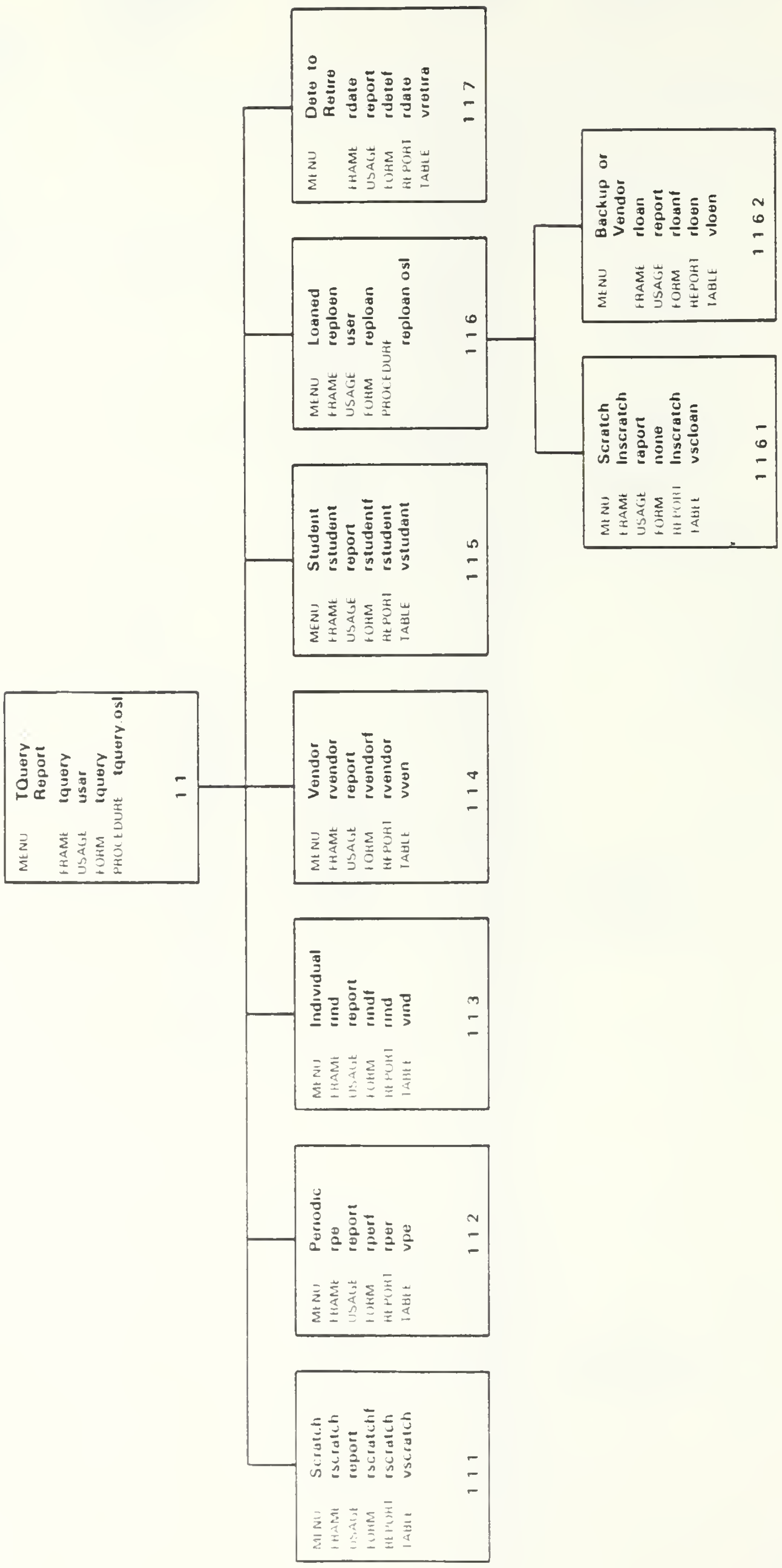
Field:	utility			
Name:	utility			
Data Class:	set			
Definition:	The software utility used to read or write the backup or vendor tape.			
	/ingres			
	/nps$user1	/nps$user3	/usrc	/vms$sys
	/nps$user2	/user	/vlsl	/work
Parameters:	c6			
Constraints:	Mandatory			
Where Used:	Source Input:	Table	Frame	
		sets	1.3.2, 1.3.3.1, 1.3.3.2	
	Maintenance:	Table	Frame	Action
			1.5	delete
	Other:	1.1.2, 1.1.3, 1.1.4, 1.1.5, 1.2.2, 1.2.3, 1.2.4, 1.2.5, 1.2.8.2		
Storage:	sets			
Views:	vind, vpe, vstudent, vven			

Field:	vendor			
Name:	vendor			
Data Class:	set			
Definition:	The name of the vendor of a vendor tape.			
Parameter:	c20			
Constraints:	Mandatory			
Where Used:	Source Input:	Table	Frame	
		sets	1.3.2	
	Maintenance:	Table	Frame	Action
			1.5	delete
	Other:	1.1.4, 1.2.4, 1.2.8.2		
Storage:	sets			
View:	vven			

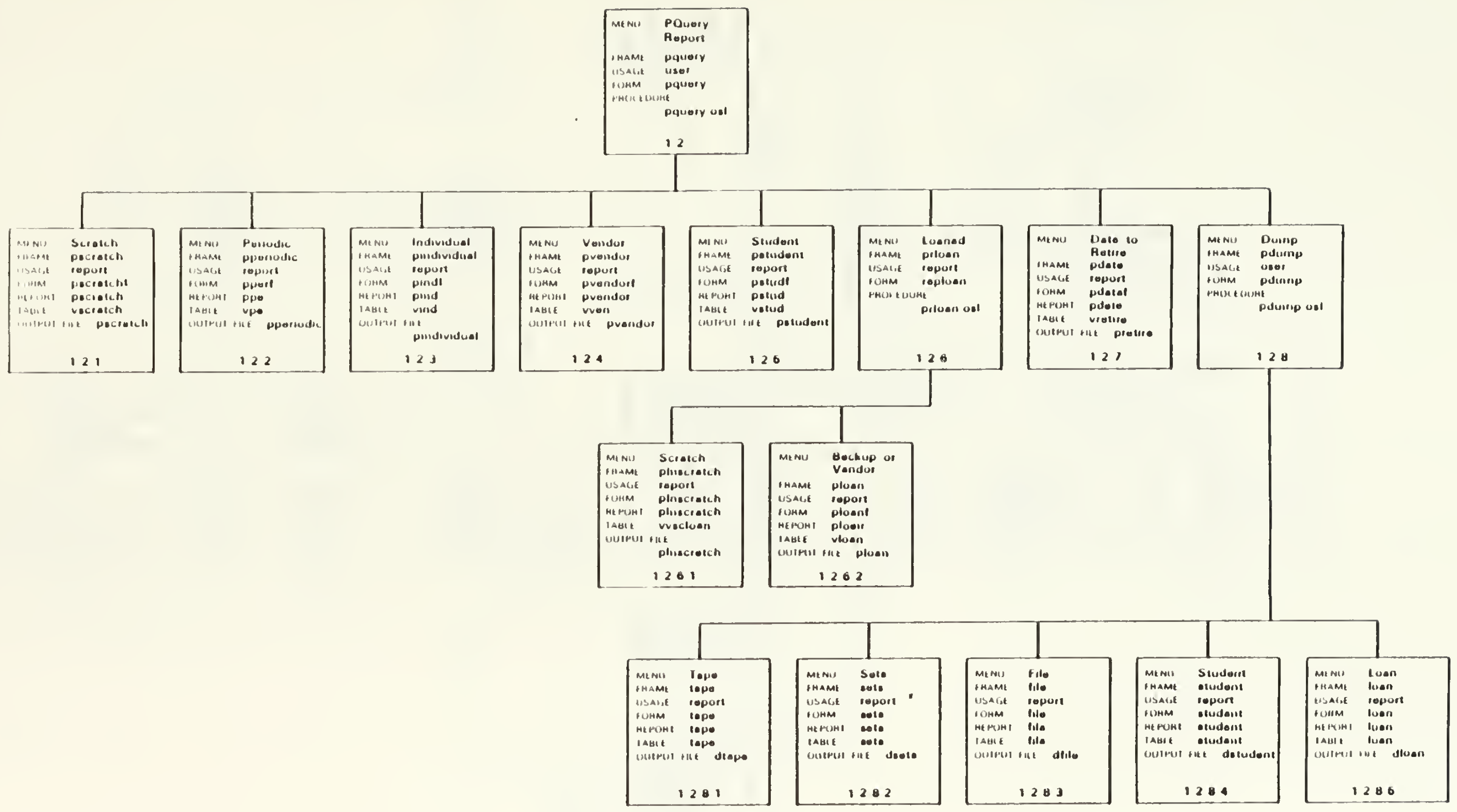
APPENDIX D

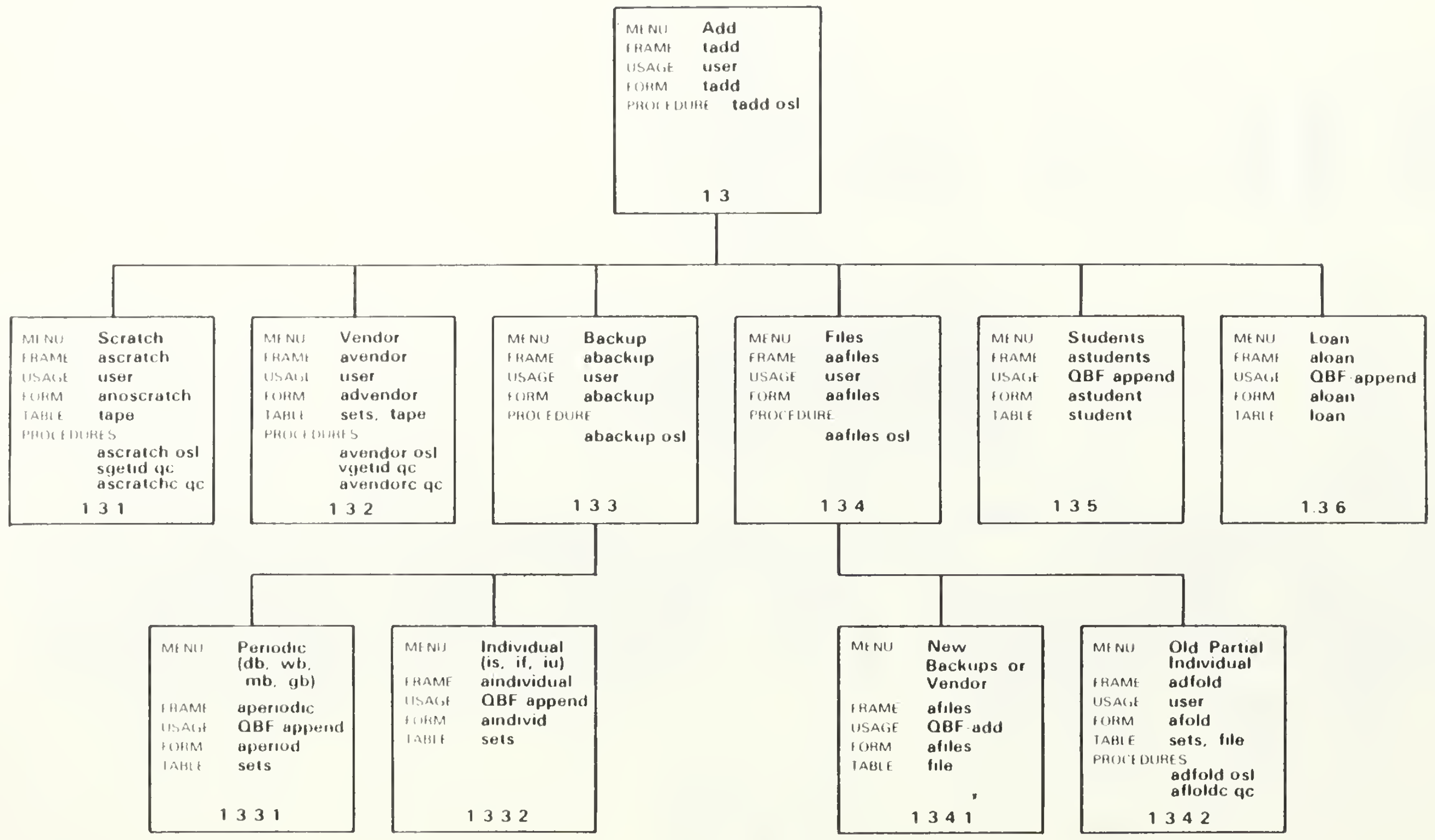
FRAMES

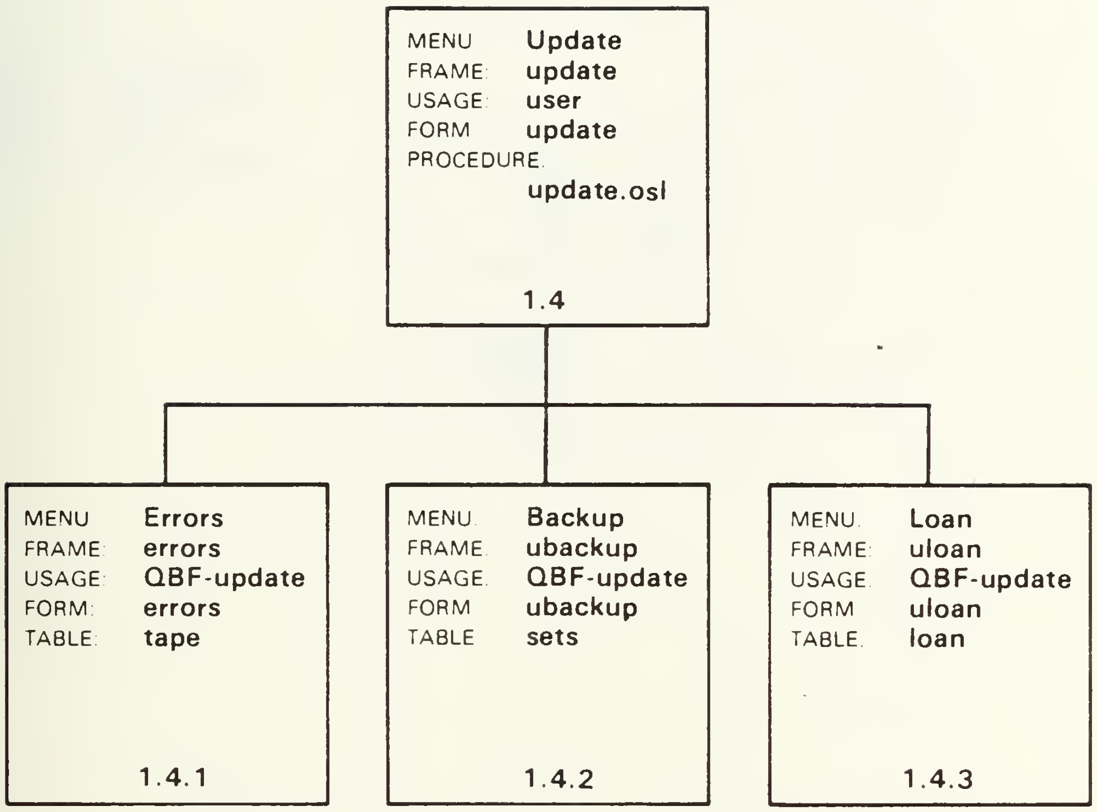




MENU TQuery Report
FRAME tquery
USAGE user
FORM tquery
PROCEDURE tquery.osl
1 1
MENU Scratch
FRAME rscratch
USAGE report
FORM rscratchf
REPORT rscratch
TABLE vscratch
1 1 1
MENU Periodic
FRAME rpe
USAGE report
FORM rperf
REPORT rper
TABLE vpe
1 1 2
MENU Individual
FRAME rind
USAGE report
FORM rindf
REPORT rind
TABLE vind
1 1 3
MENU Vendor
FRAME rvendor
USAGE report
FORM rvendorf
REPORT rvendor
TABLE vven
1 1 4
MENU Student
FRAME rstudent
USAGE report
FORM rstudentf
REPORT rstudent
TABLE vstudent
1 1 5
MENU Loaned
FRAME reploan
USAGE user
FORM reploan
PROCEDURE reploan.osl
1 1 6
MENU Date to Retire
FRAME rdate
USAGE report
FORM rdatef
REPORT rdate
TABLE vretira
1 1 7
MENU Scratch
FRAME lnscratch
USAGE raport
FORM none
REPORT lnscratch
TABLE vscloan
1 1 6 1
MENU Backup or Vendor
FRAME rloan
USAGE report
FORM rloanf
REPORT rloan
TABLE vloan
1 1 6 2


MENU PQuery Report
FRAME pquery
USAGE user
FORM pquery
PROCEDURE pquery osl
1 2
MENU Scratch
FRAME pscratch
USAGE report
FORM pscratcht
REPORT pscratch
TABLE vscratch
OUTPUT FILE pscratch
1 2 1
MENU Periodic
FRAME pperiodic
USAGE report
FORM pperf
REPORT ppe
TABLE vpe
OUTPUT FILE pperiodic
1 2 2
MENU Individual
FRAME pindividual
USAGE report
FORM pindl
REPORT pind
TABLE vind
OUTPUT FILE pindividual
1 2 3
MENU Vendor
FRAME pvendor
USAGE report
FORM pvendorf
REPORT pvendor
TABLE vven
OUTPUT FILE pvandor
1 2 4
MENU Student
FRAME pstudent
USAGE report
FORM pstudf
REPORT pstud
TABLE vstud
OUTPUT FILE pstudent
1 2 6
MENU Loaned
FRAME prloan
USAGE report
FORM reploan
PROCEDURE prloan osl
1 2 6
MENU Date to Retire
FRAME pdate
USAGE report
FORM pdatef
REPORT pdate
TABLE vretire
OUTPUT FILE pretire
1 2 7
MENU Dump
FRAME pdump
USAGE user
FORM pdump
PROCEDURE pdump osl
1 2 8
MENU Scratch
FRAME plnscratch
USAGE report
FORM plnscratch
REPORT plnscratch
TABLE vvscloan
OUTPUT FILE plnscratch
1 2 6 1
MENU Backup or Vendor
FRAME ploan
USAGE report
FORM ploanf
REPORT ploanr
TABLE vloan
OUTPUT FILE ploan
1 2 6 2
MENU Tape
FRAME tape
USAGE report
FORM tape
REPORT tape
TABLE tape
OUTPUT FILE dtape
1 2 8 1
MENU Sets
FRAME sets
USAGE report
FORM sets
REPORT sets
TABLE sets
OUTPUT FILE dsets
1 2 8 2
MENU File
FRAME file
USAGE report
FORM file
REPORT file
TABLE file
OUTPUT FILE dfile
1 2 8 3
MENU Student
FRAME student
USAGE report
FORM student
REPORT student
TABLE student
OUTPUT FILE dstudent
1 2 8 4
MENU Loan
FRAME loan
USAGE report
FORM loan
REPORT loan
TABLE loan
OUTPUT FILE dloan
1 2 8 6


MENU Add
FRAME tadd
USAGE user
FORM tadd
PROCEDURE tadd osl
1 3
MENU Scratch
FRAME ascratch
USAGE user
FORM anoscratch
TABLE tape
PROCEDURES ascratch osl sgetid qc ascratchc qc
1 3 1
MENU Vendor
FRAME avendor
USAGE user
FORM advendor
TABLE sets, tape
PROCEDURES avendor osl vgetid qc avendorc qc
1 3 2
MENU Backup
FRAME abackup
USAGE user
FORM abackup
PROCEDURE abackup osl
1 3 3
MENU Files
FRAME aafiles
USAGE user
FORM aafiles
PROCEDURE aafiles osl
1 3 4
MENU Students
FRAME astudents
USAGE QBF append
FORM astudent
TABLE student
1 3 5
MENU Loan
FRAME aloan
USAGE QBF-append
FORM aloan
TABLE loan
1.3 6
MENU Periodic (db, wb, mb, gb)
FRAME aperiodic
USAGE QBF append
FORM aperiod
TABLE sets
1 3 3 1
MENU Individual (is, if, iu)
FRAME aindividual
USAGE QBF append
FORM aindivid
TABLE sets
1 3 3 2
MENU New Backups or Vendor
FRAME afiles
USAGE QBF-add
FORM afiles
TABLE file
1 3 4 1
MENU Old Partial Individual
FRAME adfold
USAGE user
FORM afold
TABLE sets, file
PROCEDURES adfold osl afloldc qc
1 3 4 2


MENU Update
FRAME: update
USAGE: user
FORM update
PROCEDURE.
update.osl
1.4
MENU Errors
FRAME: errors
USAGE: QBF-update
FORM: errors
TABLE: tape
1.4.1
MENU. Backup
FRAME. ubackup
USAGE. QBF-update
FORM ubackup
TABLE sets
1.4.2
MENU. Loan
FRAME: uloan
USAGE. QBF-update
FORM uloan
TABLE. loan
1.4.3

MENU:	**Delete**
FRAME:	**tdelete**
USAGE:	**user**
FORM:	**tdelete**
PROCEDURES:	
	tdelete.osl
	dloanc.qc
	dbackc.qc
	dtapec.qc
	1.5

APPENDIX E

FRAME DEFINITIONS

1.
MENU: Topframe
FRAME: topframe
USAGE: user
FORM: topframe

```
                         SELECT FUNCTION


                 t  query/report (terminal)

                 p  query/report (printer)

                 add

                 update

                 delete

                 exit


   T   P   ADD   UPDATE   DELETE   EXIT:
```

```
PROCEDURE:  topframe.osl

            t = {callframe tquery;}
            p = {callframe pquery;}
            add = {callframe tadd;}
            update = {callframe update;}
            delete = {callframe tdelete;}
            exit = {exit;}
```

```
1.1
MENU:   TQuery/Report (terminal)
FRAME:  tquery
USAGE:  user
FORM:   tquery
```

```
                    QUERY/REPORT TO TERMINAL

                  scratch

                  periodic (db, wb, mb, gb)

                  individual (is, if, iu)

                  vendor

                  student

                  loaned

                  date to retire

                  return

                  exit

   SCRATCH  PER  IND  VEN  STU  LOAN  DATE  RETURN  EXIT:
```

```
PROCEDURE:  tquery.osl

            scratch = {callframe rscratch;}
            per = {callframe rpe;}
            ind = {callframe rind;}
            ven = {callframe rvendor;}
            stu = {callframe rstudent;}
            loan = {callframe reploan;}
            date = {callframe rdate;}
            return = {return;}
            exit = {exit;}
```

1.1.1
MENU: scratch
FRAME: rscratch
USAGE: report
FORM: rscratchf

```
              ENTER SCRATCH REPORT PARAMETERS



          min_length:                      max_length:




  HELP  REPORT  RETURN:
```

Field Attributes: Mandatory - max_length

REPORT: rscratch

```
  11-SEP-1985                                             10:33:04
                      REPORT ON SCRATCH TAPES
                        Table is: vscratch


   id          length          bpi          err            derr
  (i5)          (i4)          (i5)      (mm/dd/yr)      (mm/dd/yr)
```

Report Definition: Sort by length, bpi, errors, id
                   Select length by range at runtime
                   Output to terminal
                   Command flag - L80

TABLE: rscratch

```
1.1.2
MENU:    Periodic
FRAME:   rpe
USAGE:   report
FORM:    rperf
```

```
         ENTER PARAMATERS FOR PERIODIC BACKUP REPORT
                        Table is: vpe

         type:                              system:

         min_dcreate:                       max_dcreate:

         part_disk:                         sav_src:


   HELP  REPORT  RETURN:
```

```
         Field Attributes:
               Mandatory - type, system, max_dcreat, part_
                           dick, sav_src (type = db, wb, mb,
                           gb, or * only)
               Validation- type in [db, wb, mb, gb, *]
                           system (refer to Appendix C)
                           * selects "all"

REPORT:  rper
```

```
  11-SEP-1985                                            10:34:12
                      REPORT ON PERIODIC BACKUPS
                         Report on Table: vpe

  type:                 dcreate: mm/dd/yy      utility:
  id: i5                system:                bpi: i5
  sequ: p4.1            part_disk:             label:
                        sav_src:               descr:
                                               err: i2
```

```
        Report Definition: Sort by type, dcreate, system,
                             part_disk, sequ, sav_src
                           Select dcreate by range at runtime
                           Select type, system, part_disk,
                             sav_src by value at runtime
                           Output to terminal
                           Command Flag - L80

TABLE:  vpe
```

1.1.3
MENU: Individual
FRAME: rind
USAGE: report
FORM: rindf

```
          ENTER INDIVIDUAL BACKUP REPORT PARAMETERS


   type:                                           owner:
                         system:
   part_disk:                                      sav_src:


 HELP  REPORT  RETURN:
```

```
         Field Attributes:
               Mandatory - type, owner, system, part_disk,
                             sav_src
                            (type = is, if, iu, or * only)
               Validation- type in ["is", "if", "iu", "*"]
                            System (refer to Appendix C)
                            "*" selects all
```

REPORT: rind

```
 11-SEP-1985                                          10:36:04
                REPORT ON INDIVIDUAL BACKUPS
                       Table is:  vind


 type:                   system:      utility:   length: (i4)
 owner:                  part_disk:   label:     perusd: (i2)
 dcreate: (mm/dd/yy)     sav_src:     bpi: (i5)  err: (i2)
 id: (i5)                             descr:
                         sequ: (f4.1) advisor:   contph:
```

```
         Report Definition: Sort by type, owner, dcreate,
                              system, sequence
                            Select type, owner, system,
                              part_disk, sav_src by value at
                              runtime
                            Output to terminal
                            Command flag - L80
```

TABLE: vind

1.1.4
MENU: Vendor
FRAME: rvendor
USAGE: report
FORM: rvendorf

```
            ENTER PARAMETERS FOR VENDOR REPORT
                     Table is: vven



    software:                              vendor:

                                system:


 HELP   REPORT   RETURN
```

Field Attributes:
    Mandatory - software, vendor, system
    Validation- system (Appendix C) * selects all

REPORT: rvendor

```
 11-SEP-1985                                           12:48:35
                   REPORT ON VENDOR TAPES
                   Report on Table:  vven


 software:             part_disk:             utility:
 vendor:               sav_src:               label:
 system:               descr:                 bpi: (i5)
 owner:                advisor:               err: (i2)
 id: (i5)              contph:
 sequ: (f4.1)
```

Report Definition: Sort by software, vendor, system,
  owner, sequ
Select software, vendor, system,
  by value by runtime
Output to terminal
Command flag - L80

TABLE: vven

```
1.1.5
MENU:    Student
FRAME:   rstudent
USAGE:   report
FORM:    rstudentf
```

```
             ENTER PARAMETERS FOR STUDENT REPORT
                    Table is: vstudent


   min_dcreate:                        max_dcreate:


   name:                               system:


 HELP  REPORT  RETURN:
```

```
         Field Attributes:
              Mandatory - max_dcreate, name, system
              Validation- System (Appendix C) *selects all
```

```
REPORT:  rstudent
```

```
 11-SEP-1985                                          10:37:26
                  REPORT ON STUDENT TAPES
                Report on Table:  vstudent

 dcreate: (mm/dd/yr)       system:           .label:
 name:                     part_disk:         utility:
 sequ: (f1.1)              sav_src:           bpi: (i5)
 id: (i5)                  descr:             advisor:
                                              contph:
```

```
         Report Definition: Sort by dcreate, name, system,
                              part_disk, sequ, sav_src
                            Select dcreate by range at
                              runtime
                            Select name, system by value at
                              runtime
                            Output to terminal
                            Command flag - L80
```

```
TABLE:  vstudent
1.1.6
```

```
1.1.6
MENU:    Loaned
FRAME:   reploan
USAGE:   user
FORM:    reploan
```

```
            SELECT TYPE OF LOANED TAPES TO REPORT


                      scratch

                      backup or vendor

                      return

                      exit


 SCRATCH  BACKUP  RETURN  EXIT
```

```
PROCEDURE:  reploan.osl

            scratch = {callframe lnscratch;}
            backup = {callframe rloan;}
            return = {return;}
            exit - {exit;}
```

```
1.1.6.1
MENU:    Scratch
FRAME:   lnscratch
USAGE:   report
FORM:    default blank
```

```




  HELP  REPORT  RUN:
```

```
REPORT:  lnscratch
```

```
  11-SEP-1985                                           10:39:14
                    REPORT ON LOANED SCRATCH TAPES
                       Report on Table:  vscloan


   id: (i5)          edr: (mm/dd/yr)           loanto:
                     dloan: (mm/dd/yr)         lcntph:
```

```
         Report Definition: Sort by id
                            Output to terminal
                            Command flag - L80
```

```
TABLE:  vscloan
```

1.1.6.2
MENU: Backup or Vendor
FRAME: rloan
USAGE: report
FORM: rloanf

```
            ENTER PARAMETERS FOR LOANED TAPES REPORT
                        Table is:  vloan


      min_edr:                                  max_edr:

                 loanto:


  HELP  REPORT  RETURN:
```

Field Attributes:
      Mandatory - max_edr, loanto

REPORT: rloan

```
  11-SEP-1985                                          10:40:26

            REPORT ON LOANED BACKUP AND VENDOR TAPES
                     Report on Table:  vloan


  id: (i5)            loanto:             type:
  edr: (mm/dd/yr)     lcntph:             software:
  dloan: (mm/dd/yr)   owner:              system:
                                          sequ: (f4.1)
```

```
Report Definition: Sort by id
                   Select - edr by range at runtime
                            loanto by value at
                              runtime
                   Output file - terminal
                   Command flag - L80
```

TABLE: vloan

1.1.7
MENU: Date to Retire
FRAME: rdate
USAGE: report
FORM: rdatef

```
         ENTER PARAMETERS FOR RETIREMENT DATE REPORT
                     Table is:  vretire


     min_dretire:                          max_dretire:


                          type:


  HELP  REPORT  RETURN:
```

Field Attributes:
    Mandatory - max_dretire, type

REPORT: rdatef

```
 11-SEP-1985                                        12:50:55
                 REPORT ON RETIREMENT DATES
                  Report on Table:  vretire


 dretire: (mm/dd/yr)     type:             length: (i4)
 dcreate: (mm/dd/yr)     system:           bpi: (i5)
 id: (i5)                sequ: (f4.1)      err: (i2)
```

```
Report Definition: Sort by dretire, type, dcreate,
                     system, sequ
                   Select dretire by range at
                     runtime
                   Select type by value at runtime
                   Output file - terminal
                   Command flag - L80
```

TABLE: vloan

```
1.2
MENU:    PQuery/Report
FRAME:   pquery
USAGE:   user
FORM:    pquery
```

```
                         SELECT PRINTED REPORT

              scratch

              periodic backups (db, wb, mb, gb)

              individual backups (is, if, iu)

              vendor

              student

              loaned

              date to retire

              dump

              return

              exit

 SCRATCH  PER  IND  VEN  STU  LOAN  DATE  DUMP  RETURN  EXIT:
```

```
PROCEDURE:  pquery.osl

            scratch = {callframe pscratch;}
            per = {callframe pperiodic;}
            ind = {callframe pindividual;}
            ven = {callframe pvendor;}
            stu = {callframe pstudent;}
            loan = {callframe prloan;}
            date = {callframe pdate;}
            dump = {callframe pdump;}
            return = {return;}
            exit = {exit;}
```

1.2.1
MENU: Scratch
FRAME: pscratch
USAGE: report
FORM: pscratchf

```
                ENTER SCRATCH REPORT PARAMETERS



   min_length:                           max_length:


                   outputs to file:  pscratch


 HELP  REPORT  RETURN:
```

Field Attributes: Mandatory - max_length

REPORT: pscratch

```
 11-SEP-1985                                            10:41:35
                     REPORT ON SCRATCH TAPES
                   Report on Table:  vscratch


  id            length          bpi          err        derr
 (i5)            (i4)          (i5)          (i2)    (mm/dd/yr)
```

```
Report Definition: Sort by length, bpi, err, id
                   Select length by range at runtime
                   Output file - pscratch
                   command flag - L132
```

TABLE: vscratch

1.2.2
MENU: Periodic
FRAME: pperiodic
USAGE: report
FORM: pperf

```
         ENTER PARAMETERS FOR PERIODIC BACKUP REPORT
                      Table is:  vpe

         type:                           system:
         min_dcreate:                    max_dcreate:
         part_disk:                      sav_src:

                 outputs to file:  periodic


  HELP  REPORT  RETURN:
```

```
         Field Attributes:
               Mandatory - type, system, max_dcreate,
                           part_disk, sav_src
               Validation- type = db, wb, mb, gb, *
                           System (Appendix C) * selects all
```

REPORT: ppe

```
  11-SEP-1985                                         10:42:53
               REPORT ON PERIODIC BACKUP TAPES
                    Report on Table:  vpe


    id: (i5)      type:     dcreate: (mm/dd/yr)    system:
    utility:      label:    bpi: (i5)              err: (i2)

              part_disk:          sav_src:
              sequ: (f4.1)        descr:
```

```
        Report Definition: Sort by type, dcreate, system,
                             part_disk, sequ, sav_src
                           Select type, system, part_disk,
                             sav_src by value at runtime
                           Select dcreate by range at runtime
                           Output to file - pperiodic
                           Command Flag - L132
```

TABLE: vpe

1.2.3
MENU: Individual
FRAME: pindividual
USAGE: report
FORM: pindf

```
          ENTER INDIVIDUAL BACKUP REPORT PARAMETERS


  type:                                           owner:
                             system:

  part_disk:                                      sav_src:

                outputs to file: pindividual


 HELP  REPORT  RETURN:
```

Field Attributes: Mandatory - type, owner, system,
                   part_disk, sav-src
                   (type = is, if, iu, or * only)

REPORT: pind

```
 11-SEP-1985                                           12:45:08
                REPORT ON INDIVIDUAL BACKUPS
                       Table is:  vind

 type:                  system:               part_disk:
 owner:                 sequ: (f1.1)          sav_src:
 dcreate: (mm/dd/yr)    id: (i5)              descr:


 utility:               length: (i4)          advisor:
 label:                 perusd: (i2)          contph:
 bpi: (i5)              err: (i2)
```

Report Definition: Sort by type, owner, dcreate,
                     system, sequ
                   Select type, owner, system,
                     part_disk, sav_src by value
                   Output to file - pindividual
                   Command flag - L132

TABLE: vind

```
1.2.4
MENU:    Vendor
FRAME:   pvendor
USAGE:   report
FORM:    pvendorf
```

```
             ENTER PARAMETERS FOR VENDOR REPORT
                     Table is:  vven

   software:                                   vendor:

                         system:

                 outputs to file:  pvendor


 HELP  REPORT  RETURN:
```

```
         Field Attributes:
               Mandatory - software, vendor, system
               Validation- system (Appendix C) * selects all
```

```
REPORT:  pvendor
```

```
 11-SEP-1985                                           11:53:22
                   REPORT ON VENDOR TAPES
                   Report on Table:  vven


 software:  system:       part_disk:  utility:    err: (i2)
 vendor:    sequ: (f4.1)  sav_src:    label:      advisor:
 owner:     id: (i5)      descr:      bpi: (i5)   contph:
```

```
         Report Definition: Sort by software, vendor, system,
                              owner, sequ
                            Select software, vendor, system
                              by value at runtime
                            Output to file - pvendor
                            Command flag - L132
```

```
TABLE:  vven
```

1.2.5
MENU: Student
FRAME: pstudent
USAGE: report
FORM: pstudf

```
            ENTER PARAMETERS FOR STUDENT REPORT
                   Table is:  vstudent


    min_dcreate:                                max_dcreate:
    name:                                       system:

                 outputs to file:  pstudent


  HELP  REPORT  RUN
```

Field Attributes:
Mandatory - max_dcreate, name, system
Validation- system (Appendix C) * selects all

REPORT: pstud

```
  11-SEP-1985                                      11:55:35
                  REPORT ON STUDENT TAPES
                Report on Table:  vstudent


  dcreate: (mm/dd/yr)       name:          system:
  id: (i5)                  advisor:       part_disk:
  sequ: (f4.1)              contph:        sav_src:

             utility:              descr:
             label:
             bpi: (i5)
```

```
Report Definition: Sort by dcreate, name, system,
                     part_disk, sequ, sav_src
                   Select dcreate by range at
                     runtime
                   Select name, system by value at
                     runtime
                   Output to file - pstudent
                   Command flag - L132
```

TABLE: vstud

```
1.2.6
MENU:   Loaned
FRAME:  prloan
USAGE:  user
FORM:   reploan
```

```
           SELECT TYPE OF LOANED TAPES TO REPORT


                    scratch

                    backup or vendor

                    return

                    exit


 SCRATCH  BACK  RETURN  EXIT:
```

```
PROCEDURE:  prloan.osl

            scratch = {callframe plnscratch;}
            back = {callframe ploan;}
            return = {return;}
            exit = {exit;}
```

```
1.2.6.1
MENU:    Scratch
FRAME:   plnscratch
USAGE:   report
FORM:    pinscratch
```

```
+------------------------------------------------------------+
|                                                            |
|                                                            |
|                                                            |
|               outputs to file:  pinscratch                 |
|                                                            |
|                                                            |
|                                                            |
|  HELP   REPORT   RETURN:                                   |
|                                                            |
+------------------------------------------------------------+
```

```
REPORT:  plnscratch
```

```
+------------------------------------------------------------+
|  11-SEP-1985                                    11:08:19   |
|              REPORT ON LOANED SCRATCH TAPES                |
|                 Report on Table:  vscloan                  |
|                                                            |
|                                                            |
|  loanto:    lcntph:      edr:          dloan:     id: (i5) |
|                      (mm/dd/yr)    (mm/dd/yr)             |
|                                                            |
+------------------------------------------------------------+
```

```
         Report Definition: Sort by name, id, edr
                            Output to file - plnscratch
                            Command flag - L132
```

```
TABLE:  vscloan
```

```
1.2.6.2
MENU:    Backup or Vendor
FRAME:   ploan
USAGE:   report
FORM:    ploanf
```

```
+--------------------------------------------------------------+
|          ENTER PARAMETERS FOR LOANED TAPES REPORT            |
|                     Table is:  vloan                         |
|                                                              |
|                                                              |
|   min_edr:                                     max_edr:      |
|                           loanto:                            |
|                                                              |
|                   outputs to file:  ploan                    |
|                                                              |
|                                                              |
|  HELP   REPORT   RETURN:                                     |
+--------------------------------------------------------------+
```

Field Attributes: Mandatory - max_edr, loanto

REPORT: ploan

```
+--------------------------------------------------------------+
|  11-SEP-1985                                       11:10:04  |
|          REPORT ON LOANED BACKUP AND VENDOR TAPES            |
|                  Report on Table:  vloan                     |
|                                                              |
|                                                              |
|  loanto:        edr: (mm/dd/yr)       dloan: (mm/dd/yr)      |
|  id: (i5)       lcntph:                                      |
|                                                              |
|  type:          owner:                system:                |
|  software:                            sequ: (f4.1)           |
+--------------------------------------------------------------+
```

```
Report Definition: Sort by name, edr, id
                   Select edr by range at runtime
                   Select loanto by value at runtime
                   Output to file - ploan
                   Command flag - L132
```

TABLE: vloan

1.2.7
MENU: Date to Retire
FRAME: pdate
USAGE: report
FORM: pdatef

```
          ENTER PARAMETERS FOR RETIREMENT DATE REPORT
                      Table is:  vretire


     min_dretire:                          max_dretire:
                         type:

               outputs to file:  pretire


  HELP  REPORT  RETURN:
```

Field Attributes: Mandatory - max_dretire, type

REPORT: pdate

```
  11-SEP-1985                                          10:47:51
                 REPORT ON RETIREMENT DATES
                  Report on Table:  vretire


  dretire: (mm/dd/yr)      type:        dcreate: (mm/dd/yr)
  id: (i5)

  system:        sequ:         length:      bpi:       err:
                (f4.1)          (i4)        (i5)       (12)
```

Report Definition: Sort by dretire, type, dcreate,
                     system, sequ
                   Select dretire by range at
                     runtime
                   Select type by value at runtime
                   Output to file - pretire
                   Command flag - L132

TABLE: vretire

```
1.2.8
MENU:    Dump
FRAME:   pdump
USAGE:   user
FORM:    pdump
```

```
                SELECT TABLE FOR DUMP REPORT


                           tape

                           sets

                           file

                           student

                           loan

                           return

                           exit


 TAPE   SETS   FILE   STU   LOAN   RETURN   EXIT:
```

```
PROCEDURE:  pdump.osl

            tape = {callframe tape;}
            sets = {callframe sets;}
            file = {callframe file;}
            stu = {callframe student;}
            loan = {callframe loan;}
            return = {return;}
            exit = {exit;}
```

```
1.2.8.1
MENU:    Tape
FRAME:   tape
USAGE:   report
FORM:    tape
```

```
                    outputs to file:  dtape




  HELP  REPORT  RETURN:
```

```
REPORT:  tape
```

```
  11-SEP-1985                                              10:49:57
                              DUMP
                      Report on Table:  tape


      id        length        bpi        err          derr
     (i5)        (i4)        (i5)        (i2)      (mm/dd/yr)
```

```
            Report Definition: Sort by id
                               Output to file - dtape
                               Command flag - L80
```

```
TABLE:  tape
```

```
1.2.8.2
MENU:   Sets
FRAME:  sets
USAGE:  report
FORM:   sets
```

```
                    outputs to file:  dsets



 HELP  REPORT  RETURN:
```

REPORT: sets

```
 11-SEP-1985                                              10:50:29
                               DUMP
                      Report on Table:  sets


       id: (i5)    type: (c2)    dcreate: (mm/dd/yr)
       software:                 vendor:
       descr:

       dretire: (mm/dd/yr)     system:      utility:
                               owner:       advisor:

       label:       perusd: (i2)
       contph:      sequ: (f4.1)
```

```
        Report Definition: Sort by id
                           Output to file - dsets
                           Command flag - L132
```

TABLE: sets

```
1.2.8.3
MENU:    File
FRAME:   file
USAGE:   report
FORM:    file
```

```
                    outputs to file:  dfile



  HELP  REPORT  RETURN:
```

```
REPORT:  file
```

```
 11-SEP-1985                                            11:10:49
                               DUMP
                      Report on Table:  file


      id        part_disk        sav_src
     (i5)
```

```
         Report Definition: Sort by id
                            Output to file - dfile
                            Command flag - L132
```

```
TABLE:  file
```

```
1.2.8.4
MENU:   Student
FRAME:  student
USAGE:  report
FORM:   student
```

```
+----------------------------------------------------------+
|                                                          |
|                                                          |
|               outputs to file:  dstudent                 |
|                                                          |
|                                                          |
|  HELP  REPORT  RETURN:                                   |
+----------------------------------------------------------+
```

```
REPORT:  student
```

```
+----------------------------------------------------------+
|  11-SEP-1985                                   10:51:51  |
|                          DUMP                            |
|                Report on Table:  student                 |
|                                                          |
|     id       name        advisor          contph         |
|    (i5)                                                  |
+----------------------------------------------------------+
```

```
        Report Definition: Sort by id
                           Output to file - dstudent
                           Command flag - L80
```

```
TABLE:  student
```

1.2.8.5
MENU: Loan
FRAME: loan
USAGE: report
FORM: loan

```
                    outputs to file:  dloan



  HELP  REPORT  RETURN:
```

REPORT: loan

```
  11-SEP-1985                                          10:52:33
                              DUMP
                     Report on Table:  loan


    id      dloan        edr          loanto         lcntph
   (i5)   (mm/dd/yr)  (mm/dd/yr)
```

```
         Report Definition: Sort by id
                            Output to file - dloan
                            Command flag - L132
```

TABLE: loan

1.3
MENU: Add
FRAME: tadd
USAGE: user
FORM: tadd

```
                         SELECT ADD


                         scratch

                         vendor

                         backup

                         files

                         students

                         loan

                         return


 SCRATCH  VENDOR  BACKUP  FILES  STUDENTS  LOAN  RETURN:
```

PROCEDURE: tadd.osl

```
            scratch = {callframe ascratch;}
            vendor = {callframe avendor;}
            backup = {callframe abackup;}
            files = {callframe aafiles;}
            students = {callframe astudents;}
            loan = {callframe aloan;}
            return = {return;}
```

1.3.1
MENU: Scratch
FRAME: ascratch
USAGE: user
FORM: anoscratch

```
                          ADD SCRATCH TAPES
                           Table is:  tape


  id:                     length:                 bpi:

                     select id; enter data, add


 ID  ADD  RETURN:
```

```
        Field Attributes:
              Display only - id
              Repeat values for - length, bpi
              Mandatory - length, bpi
              Validation - length, bpi (Appendix C)
```

```
PROCEDURE (1):  ascratch.osl

                id = {callproc sgetid;}
                add = {callproc ascratchc;}
                return = {return;}
```

1.3.1 (Continued)

PROCEDURE (2): sgetid.qc

```
     sgetid()
{
##   int   low;
##   int   top;
##   int   new;

     int   i;

     i=low=0;
##   retrieve (low=min(tape.id          *retrieve an empty slot
##   where (tape.length=0 and            with the lowest id
##   (tape.bpi=0)))
##   {
        i=i+1;
##   }
     if (low!=0)                        *if an empty slot exists
     {
##      putform anoscratch (id=low)     *put the id on the form
##      message "OLD ID"
##      sleep 2
##      return;
     }
     else                               *if no empth slots exist
     {
##        retrieve (top=max(tape.id))
##        {
            i++;
##        }
          if (top>0)
          {
            new=1+top;                  *increment the max id
                                         assigned by one
##          putform anoscratch          *put the new id number
##          (id=new)                     on the form
##          message "NEW ID"
##          sleep 2
            return;
          }
     }
}
```

TABLE: tape

1.3.1 (Continued)

PROCEDURE (3): ascratchc.qc

```
     ascratchc()

{
##   int   nu;
##   int   le;
##   int   b;
##   int   check;

     int   i;

     i=check=0;
##   getform anoscratch(nu=id,         *get the values from the
##   le=length, b=bpi)                  form
     if (nu==0)
     {
##      message "Select ID before      *if the user tried to
##      entering data"                  enter data prior to
                                        getting an id
##      sleep 2
           return;
     }
##   retrieve (check=tape.id)          *find a record of an
##   where tape.id=nu                   empty slot with the id
##   {                                  number
          i++;
##   }
          if (check>0)                 *if an empty slot of the
          {                             id number exists
##           replace tape              *update the record of
##           (length=le, bpi=b)         the empty slot in the
##           where tape.id=nu           tape table
##           message "Added to old
##           ID; label tape only"
##           sleep 2
##           putform anoscratch
##           (id=0)
             return;
          }
          else                         *if the id is not in the
          {                             tape table
##        append to tape               *append to tape table
##        (
##        id=nu,
##        length=le,
##        bpi=b
##        )
```

1.3.1 (Concluded)

```
##          message "Added to new
##          ID, label slot and tape"
##          sleep 2
##          putform anoscratch           *to prevent double entry
##          (id=0)
     }
}
```

TABLE: tape, sets

1.3.2
MENU: Vendor
FRAME: avendor
USAGE: user
FORM: advendor

```
                         ADD VENDOR TAPES
                    Tables are Tape and Sets


  id:        type:         dcreate:          dretire:

  system:              utility:        label:           bpi:

  owner:                    advisor:          contph:

     software:                  vendor:

          descr:                          sequ:


                   select id; enter data; add



  ID  ADD  RETURN:
```

```
         Field Attributes:
              Display only - id, type
              Repeat values for all except id
              Mandatory - dcreate, dretire, system, utility,
                          bpi, software, vendor
              Validation - system (Appendix C) * selects all,
                           bpi (Appendix C)
```

```
PROCEDURE (1):  avendor.osl

                id = {callproc vgetid;}
                add = {callproc avendorc;}
                return = {return;}
```

1.3.2 (Continued)

PROCEDURE (2): vgetid.qc

```
    vgetid()

{
##   int   low;
##   int   top;
##   int   new;

     int   i;

     i=low=0;
##   retrieve (low=min(tape.id          *retrieve an empty slot
##   where (tape.length=0) and            with the lowest id
##   (tape.bpi=0)))
##   {
        i=i+1;
##   }
     if (low!=0)                        *if an empty slot exists
     {
##      putform advendor                *put id and type on the
##      (id=low, type ="v")              form
##      message "OLD ID"
##      sleep 2
        return;
     }
     else                               *if no empty slots exist
     {
##       retrieve (top=max
##       (tape.id))
##       {
            i++;
##       }
         if (top>0)
         {                              *increment the max id by
            new=1+top;                   one
##          putform advendor            *put the new id and type
##          (id=new, type="v")           on the form
##          message "NEW ID"
##          sleep 2
            return;
         }
     }
}
```

TABLE: tape, vendor

1.3.2 (Continued)

PROCEDURE (3): avendorc.qc

```
     avendorc.qc()
{
##   int     nu;
##   char    ty[2];
##   char    da[26];
##   char    dr[26];
##   char    sy[9];
##   char    ut[6[;
##   char    ow[9];
##   char    co[4];
##   char    so[15];
##   char    ve[20];
##   char    la[6];
##   float   se;
##   int     b;
##   char    de[15];
##   char    ad[9];
##   int     check;

     int     i;

##   getform advendor (nu=id,          *get the values from the
##   ty=type, da=dcreate,               form
##   dr=dretire, sy=system,
##   ut=utility, ow=owner,
##   co=contph, so=software,
##   ve=vendor, la=label, se=sequ,
##   b=bpi, de=descr, ad=advisor)

     if (nu==0)
     {
##   message "Select id before          *if the user tried to
##   selecting add"                      enter data prior to
##   sleep 2                             getting an id
     return;
     }
     i=check=0;
##   retrieve (check=tape.id)           *find a record of the
##   where tape.id=nu                    empty slot with the id
##   {                                   number
         i++;
##   }
     if(check > 0)                      *if an empth slot of the
     {                                   id number exists
##    replace tape (bpi=b) where        *update the record of
##    tape.id=nu                         the empty slot in the
                                         tape table
```

1.3.2 (Concluded)

```
##    append to sets(id=nu, type=ty,
##    dcreate=da, dretire=dr,
##    system=sy, utility=ut, owner=ow,
##    contph=co, software=so, vendor=ve,
##    label=la, sequ=se, descr=de,
##    advisor=ad)
##    message "Added to old ID;
##    label tape only"
##    sleep 2
##    putform advendor (id=0)
      return;
      }
      else                           *if the id is not in the
      {                               tape table
##    append to tape(                *append to the tape
##     id=nu, bpi=b                   table
##     )
##    append to sets(id=nu, type=ty,
##    dcreate=da, dretire=dr,
##    system=sy, utility=ut, owner=ow,
##    contph=co, software, so, vendor=ve,
##    label=la, sequ=se, descr=de,
##    advisor=ad)
##    message "Added to new ID;
##    label slot and tape"
##    sleep 2
##    putform advendor(id=0)         *to prevent double entry
      }
}
```

TABLE: tape, set

1.3.3
MENU: Backup
FRAME: abackup
USAGE: user
FORM: abackup

```
                    ADD BACKUP TO SCRATCH


                 periodic (db, wb, mb, gb)

                 individual (is, if, iu)

                 return


 PERIODIC  INDIVIDUAL  RETURN:
```

```
PROCEDURE:  abackup.osl

            periodic = {callframe aperiodic;}
            individual = {callframe aindividual;}
            return = {return;}
```

```
1.3.3.1
MENU:    Periodic (db,  wb, mb, gb)
FRAME:   aperiodic
USAGE:   QBF-append
FORM:    aperiod
```

```
+------------------------------------------------------------+
|        ADD ID AND BACKUP DATA FOR EACH ID IN SET            |
|                     Table is:  sets                        |
|                                                            |
|                                                            |
|    id:                  label:                sequ:        |
|                                                            |
|    type:                dcreate:              dretire:     |
|                                                            |
|              system:              utility:                 |
|                                                            |
|                          descr:                            |
|                                                            |
|                                                            |
|  APPEND #1 (CONTROL F TO ADD, <MENU KEY> TO RETURN:         |
|                                                            |
+------------------------------------------------------------+
```

```
        Field Attributes:
              Mandatory - id, type, dcreate, dretire,
                             system, utility
              Repeat values for - type, dcreate, dretire,
                             system, utility, label, descr
              Validation- Not id in sets
                          Type in [db, wb, mb, gb]
                          System (Appendix C) * selects all
```

TABLE: sets

```
1.3.3.2
MENU:   Individual (is, if, iu)
FRAME:  aindividual
USAGE:  QBF-append
FORM:   aindivid
```

```
            ADD ID AND BACKUP DATA FOR EACH ID IN SET
                          Table is:  sets


 id:            label:            sequ:            perusd:

    type:              dcreate:          dretire:

          system:                    utility:

 owner:                advisor:                    contph:

                       descr:


 APPEND #1 (CONTROL F TO ADD, <MENU KEY> TO RETURN)
```

```
        Field Attributes:
              Mandatory - id, type, dcreate, dretire,
                            system, utility, owner
              Repeat values for type, dcreate, system,
                            utility, owner, advisor,
                            contph, descr
                            System (Appendix C) * selects all
              Validation- Not id in sets
                          Type in [is, if iu]
```

```
TABLE:  sets
```

1.3.4
MENU: Files
FRAME: aafiles
USAGE: user
FORM: aafiles

```
                        ADD FILE PATHS TO


      new backups or vendor

      old partial individual tape (and change perusd)

      return

      exit


  NEW  OLD  RETURN  EXIT:
```

PROCEDURE: aafiles.osl

```
new = {callframe afiles;}
old = {callframe adfold;}
return = {return;}
exit = {exit;}
```

```
1.3.4.1
MENU:   New Backups or Vendor
FRAME:  afiles
USAGE:  QBF-append
FORM:   afiles
```

```
                        ADD FILE PATH
                       Table is:  file


   id:                                 part_disk:

                 sav_src:



 IF ONE FILE PATH HAS MORE THAN ONE ID, ADD ID AND FILE
 PATH FOR EACH ID.  IF ID CONTAINS MORE THAN ONE FILE PATH,
 ADD ID AND FILE PATH FOR EACH FILE PATH



 APPEND #1 (CONTROL F TO ADD, <MENU KEY> TO RETURN
```

```
         Field Attributes:
               Mandatory - id, part_disk, sav_src
```

TABLE: file

```
1.3.4.2
MENU:    Old Partial Individual
FRAME:   aafold
USAGE:   user
FORM:    afold
```

```
+-------------------------------------------------------------+
|          ADD FILE PATH TO OLD PARTIAL INDIVIDUAL            |
|                Tables are:  file and sets                   |
|                                                             |
|                                                             |
|  id:                 part_disk:              sav_src:       |
|                                                             |
|                         perusd:                             |
|                                                             |
|                                                             |
|                  enter data and select add                  |
|                                                             |
|                                                             |
| ADD  RETURN  EXIT:                                          |
+-------------------------------------------------------------+
```

```
        Field Attributes:  Mandatory - id, part_disk,
                                       sav_src
```

```
PROCEDURE (1):  adfold.osl

                add = {callproc afloldc;}
                return = {return;}
                exit = {exit;}


PROCEDURE (2):  afoldc.qc

     afloldc()
{
##   int   nu;
##   char  pd[9];
##   char  ss[15];
##   int   pr;

##   getform afold (nu=id, pd=part_disk, ss=sav_src,
##   pr=perusd)
     if( nu == 0)
     {
##   message "You must ender an ID before selecting
##   add"
         return;
     }
```

1.3.4.2 (Continued)

```
##   append to file (id=nu, part_disk=pd, sav_src=ss)
##   replace sets (
##       perusd=pr
##       )
}
```

TABLE: file, sets

```
1.3.5
MENU:   Students
FRAME:  astudents
USAGE:  QBF-append
FORM:   astudent
```

```
                ADD ID AND STUDENT DATA
        FOR EACH ID THE STUDENT'S FILES ARE IN
                   Table is:  student


    id:                         name:

    advisor:                    contph:



 APPEND #1 (CONTROL F TO ADD, <MENU KEY> TO RETURN):
```

```
        Field Attributes:
              Mandatory - id, name, advisor, contph
              id in sets
              Repeat values for name, advisor, contph


TABLE:  student
```

1.3.6
MENU: Loan
FRAME: aloan
USAGE: QBF-append
FORM: aloan

```
             ADD ID AND LOAN DATA FOR EACH ID LOANED
                         Table is:  loan


 id:                        dloan:                   edr:

      loanto:                        lcntph:



 APPEND #1 (CONTROL F TO ADD, <MENU KEY> TO RETURN)
```

Field Attributes: Mandatory - id, dloan, edr, loanto

TABLE: loan

```
1.4
MENU:    Update
FRAME:   update
USAGE:   user
FORM:    update
```

```
                         SELECT UPDATE

                          errors

                          backup

                          loan

                          return

                          exit

 ERRORS  BACKUPS  LOAN  RETURN  EXIT:
```

```
PROCEDURE:  update.osl

            errors = {callframe errors;}
            backups = {callframe ubackup;}
            loan = {callframe uloan;}
            return = {return;}
            exit = {exit;}
```

1.4.1
MENU: Errors
FRAME: error
USAGE: QBF-update
FORM: errors

```
                         UPDATE ERRORS
                        Table is:  tape


    id:                                          err:

                        derr:


  ENTER QUERY (<MENU KEY> TO RETURN OR TO RUN):
```

Field Attributes: Mandatory - id

TABLE: tape

1.4.2
MENU: Backup
FRAME: ubackup
USAGE: QBF-update
FORM: ubackup

```
                 UPDATE BACKUP OR VENDOR DATA
                       Table is:  sets


  id:                   dcreate:                dretire:

  owner:                advisor:                contph:

          descr:                    sequ:


 ENTER QUERY (<MENU KEY> TO RETURN OR TO RUN):_
```

Field Attributes: Mandatory - id

TABLE: sets

```
1.4.3
MENU:    Loan
FRAME:   uloan
USAGE:   QBF-update
FORM:    uloan
```

```
                         UPDATE LOAN DATA
                          Table is:  loan


      id:                                    edr:

      loanto:                                lcntph:


 ENTER QUERY (<MENU KEY> TO RETURN OR TO RUN):
```

```
         Field Attributes:  Mandatory - id
```

```
TABLE:  loan
```

1.5
MENU: Delete
FRAME: tdelete
USAGE: user
FORM: tdelete

```
                            DELETE


              id:

              loan record (return tapes)

              backup (retire)

              tape (permanently remove)

              return

              exit


  LOAN  BACKUP  TAPE  RETURN  EXIT:
```

```
PROCEDURE (1):  tdelete.osl

                loan = {callproc dloanc;}
                backup = {callproc dbackc;}
                tape = {callproc dtapec;}
                return = {return;}
                exit = {exit;}
```

1.5 (Continued)

PROCEDURE (2): dloanc.qc

```
     dloanc()
{
##   int      nu;

##   getform tdelete (nu=id)
##   delete loan where loan.id = nu
}
```

TABLE: loan

PROCEDURE (3): dbackc.qc

```
     dbackc()
{
##   int      nu;

##   getform tdelete (nu=id)
##   delete file where file.id = nu
##   delete sets where sets.id = nu
##   delete student where student.id = nu
}
```

TABLE: file, sets, student

PROCEDURE (4): dtapec.qc

```
     dtapec()
{
##   int      nu;

##   getform tdelete (nu=id)
##   delete file where file.id = nu
##   delete loan where loan.id = nu
##   delete sets where sets.id = nu
##   delete student where student.id = nu
##   replace tape (length=0, bpi=0, derr=" ")
     where tape.id = nu
```

TAPE: file, loan, sets, student, tape

APPENDIX F

IMPLEMENTATION OF THE COMPUTER SCIENCE DEPARTMENT TAPE LIBRARY SYSTEM

The tape library system (TLS) application for the Computer Science (CD) Department was created using the 4.2 BSD UNIX operating system and the Ingres VMUNIX Version 2.1/15 VE.04 DBMS. It was created under the name talibrary cs using login name crawfordb. It was developed using Ingres Applications by Forms (ABF). An executable image [Ref. 4] has been built in the file /work/crawford talibrary.exe topframe. It is defined with the symbol cslib.

A. LOADING

The current library may be loaded into the tables at this time. Chapters III and IV of this thesis should be read before loading the library data. Data may be loaded into each table from a file using the copy command. The Ingres Self Instruction Guide [Ref. 5, p. 8-1] provides a comprehensive explanation of the procedure. The following comments regarding the initial loading of data to file, or directly via the application will ensure the system operates as designed.

1. General

Ingres reads the value of an upper case letter differently than the value of a lower case letter. Many of the reports are called by value. It is recommended that fields be reviewed prior to loading, and where applicable, values be standardized where validation criteria is not written into the application. Computer records may not be duplicated in the library.

2. Tables

a. TAPE. All tapes in the library must have at least an id number and either length or bpi. If both length and bpi are null, the application will report the id number as one of a scratch tape.

b. SETS. The table SETS fields - owner, software, and vendor - are the most vulnerable to inconsistent value entry. When a record is loaded into table SETS, a record of the same id must also be entered in table TAPE. The same id should never appear twice within SETS.

c. FILE. More than one record in table FILE may have the same id. If an id is included in TAPE, however, it must also be in SETS.

d. STUDENT. Student records are kept only for students whose files are contained in graduation backup tapes created for a graduating class. All of the graduated students files should be recorded on the graduation backup tape. The id numbers refer to the graduation backup tapes on which their files are recorded. An id may appear more than once in the table STUDENT. All ids in the STUDENT table should be included in tables FILE, SETS, and TAPE.

e. LOAN. An id should not appear more than once in table LOAN. All id(s) in LOAN should appear at least in table TAPE, and may appear in table SETS and FILE.

B. TABLE STORAGE STRUCTURE

The storage structure of all tables is defined as a "heap". When the current library is loaded, the storage structure should be converted to "hash" to accelerate performance of queries. This may be done by the following procedure. (%UNIX,*Ingres)

```
%ingres talibrary
*modify <tablename> to hash
```

The Ingres User's Guide [Ref. 5] and the Ingres Reference Manual [Ref. 4] provide further information on the structure and procedure.

C. ACCESS

Only the librarians should have access to the library. To provide access, the library symbol "cslib" (no quotes) must be defined in the login files of each of the librarians. When the librarian retrieves a printed report from the library, it is written to their UNIX work spaces. Therefore the librarians need work spaces large enough to store all reports defined in the application. Reports are described in Chapter IV, Section G. Documentation regarding the provision of access to users is in Applications by Forms [Ref. 6, pp.6-3].

APPENDIX G

TUTORIAL: OPERATION OF THE COMPUTER SCIENCE DEPARTMENT TAPE LIBRARY SYSTEM (TLS)

TABLE OF CONTENTS

This Appendix describes the operation of the TLS. Before using the TLS, librarians must read Chapter III, Section B, Specification Data Dictionary; Chapter IV, Section B, Tables and Views; and Section C, Design Data Dictionary, of the thesis in order to understand the tables, data fields, and codes used in the application. Thorough descriptions and illustrations of the frames, forms, procedures, and reports are provided in Chapter IV, Section E, Frames; and Section G, Frames Definition. These should be referred to by frame number. Numbers are provided below.

This tutorial is in three parts. The first part describes the types of forms used in the library and their use. The second part describes the application and which tasks or functions the various frames perform. The third part describes procedures for the database administrator to maintain the library.

The TLS is a frame-based application. Each frame has a form and/or report and a procedure associated with it. The librarian communicates with the application by the use of forms. The application responds to the user by the use of forms and reports.

The user may perform the functions of add, update, delete and retrieve. Data input is standardized and controlled through validation criteria. Operations are selected through the use of menus. The structure of the library and forms match the functions the librarian performs.

A. FORMS

There are six types of forms associated with frames in the library:

(1) Menu,
(2) Parameter Entry for Reports,
(3) Run Report Forms,
(4) Append,
(5) QBF-update, and
(6) Delete.

Each form has an operations menu at the bottom starting at the left hand corner. Many of the operation menus include the operations RETURN and/or EXIT. By selecting RETURN, the form of the previous frame will be displayed. EXIT will terminate the application and return to UNIX. To execute an operation, the user must enter a unique letter, or set of letters, which represent the operation to the right of the menu. For example, if the command menu is:

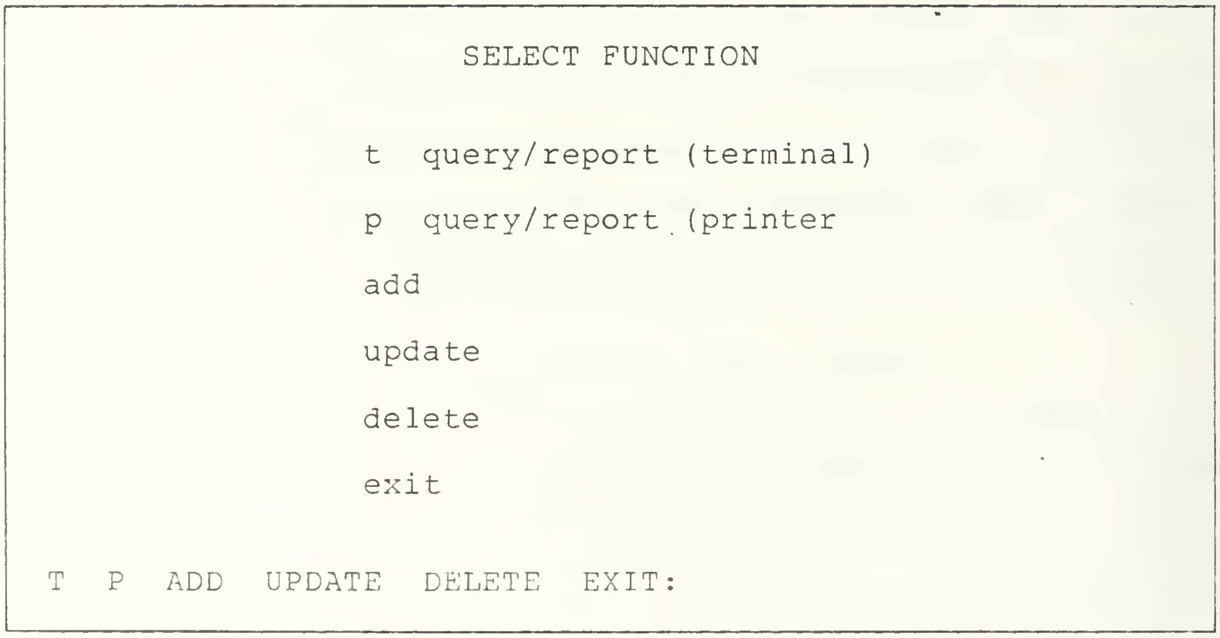
```
SCRATCH  STUD  RETURN  EXIT:
```

enter sc to select SCRATCH or enter r to select RETURN. Depress <ESC> to execute the operation. Operations are executed on all forms by using this procedure. The term "select" hereafter will be defined to mean "type" the representative command symbol, then depress <ESC>.

When using forms which include fields for data input, the cursor must be moved from field to field. It will move to the next field automatically if the number of characters or integers of the value entered fills the field. If it does not, however, the cursor can be moved to the next field by depressing the TAB key. To move it back to the previous field, depress control P.

1. Menu Forms

Menu forms provide movement to other frames. They contain the list of other frames on the upper portion and a matching list on the command menu line. A menu form is shown in Figure 11.



```
                         SELECT FUNCTION

                   t  query/report (terminal)
                   p  query/report (printer
                   add
                   update
                   delete
                   exit

  T  P  ADD  UPDATE  DELETE  EXIT:
```

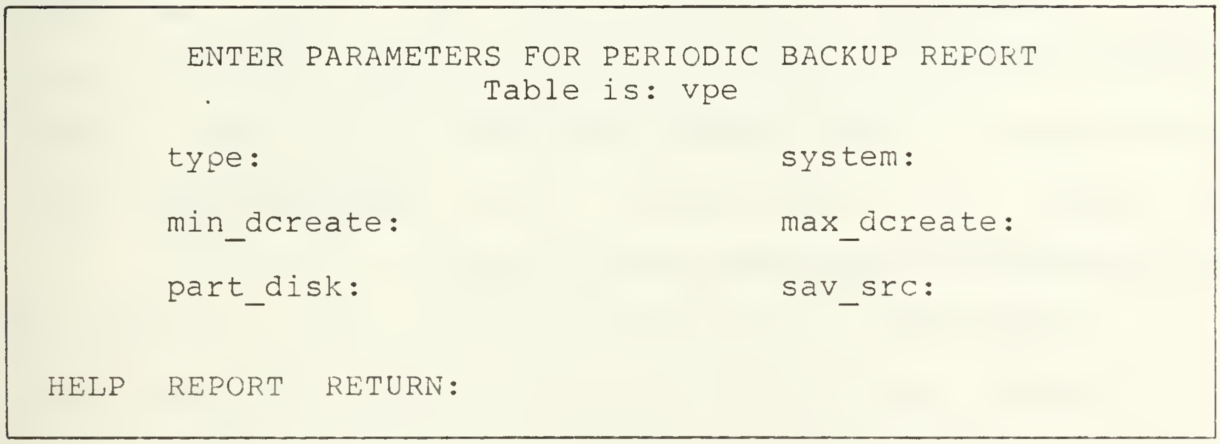
Figure 11. Menu Form

When the menu form is displayed, the cursor will always appear to the right of the operations menu. To move to another frame, select the frame desired. The new form will be displayed.

Frames 1, 1.1, 1.1.6, 1.2, 1.2.6, 1.2.8, 1.3, 1.3.3, 1.3.4, and 1.4 display menu forms.

2. Parameter Entry (for reports) Forms

Parameter Entry Forms call reports based on values or ranges entered by the user. The forms contain fields in the upper portion and a command menu at the bottom. A parameter entry form is shown in Figure 12.



```
        ENTER PARAMETERS FOR PERIODIC BACKUP REPORT
                       Table is: vpe

      type:                           system:

      min_dcreate:                    max_dcreate:

      part_disk:                      sav_src:


  HELP  REPORT  RETURN:
```

Figure 12. Parameter Entry Form

"Range" fields always include the prefix min_ or max_. Those that do not are "value" fields. Input to value fields is mandatory. If a report is desired of all values of the value field, an "*" may be entered. Only the max_ range field is mandatory. If all range values are desired, do not enter a value in the min_ range field and enter the largest value allowed in the max_ range field. Parameter Entry Forms always include the operations HELP, REPORT, and END. HELP will provide assistance to the user, REPORT will run the report, and END returns to the previous frame. When a Parameter Entry Form is first displayed, the cursor is always on the first field.

HELP may be selected at any time during parameter entry. The form will be redisplayed intact after HELP has been obtained. Select REPORT to run the report after ranges and values have been entered. Select END to return to the previous frame without running the report.

Frames 1.1.1, 1.1.2, 1.1.3, 1.1.4, 1.1.5, 1.1.6.2, 1.1.7, 1.2.1, 1.2.2, 1.2.3, 1.2.4, 1.2.5, 1.2.6.2, and 1.2.7, display Parameter Entry Forms.

3. Run Report Forms

Run Report Forms are used to run reports. No selection of values is available to the user. A Run Report Form is shown in Figure 13.

```
               outputs to file:  plnscratch


 HELP  REPORT  RETURN:
```

Figure 13. Run Report Form

The upper portion is either blank or displays a line informing the user to which file the report will be written. The command menu line is the same as report parameter entry forms.

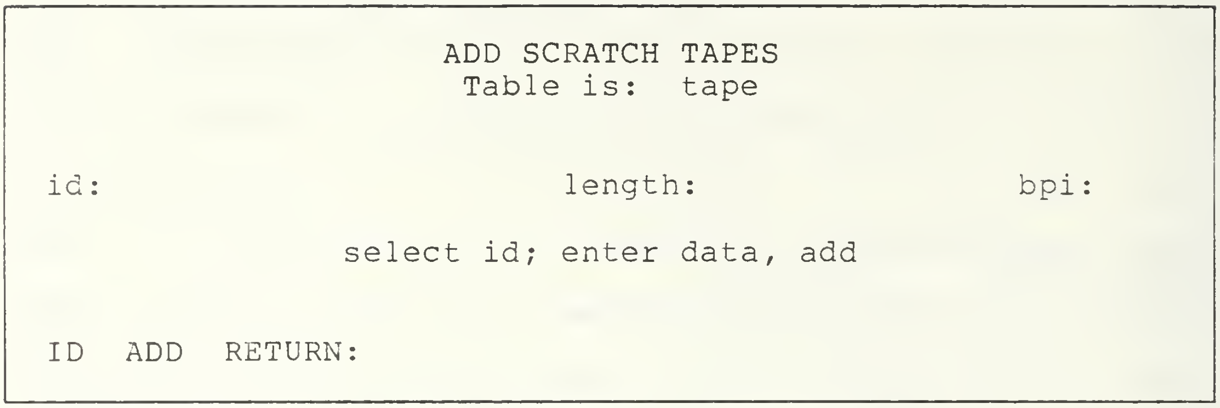
Frames 1.1.6.1, 1.2.6.1, 1.2.8.1, 1.2.8.2, 1.2.8.3, 1.2.8.4, and 1.2.8.5 display Run Report Forms.

4. Append Forms

Append forms allow the user to append new records to the library. They display fields on the upper portion for data entry and a command menu line at the bottom. Append is simple; there are three variations of Append Forms.

a. The first type of Append Form provides programmed id assignment, and is shown in Figure 14.

This type of form is associated with a procedure which automatically assigns an id number and perhaps another value to the form. The fields which are automatically assigned are for display only. The cursor will always appear



```
                    ADD SCRATCH TAPES
                    Table is:  tape


id:                       length:                  bpi:

              select id; enter data, add


ID  ADD  RETURN:
```

Figure 14. Append Form (Programmed id Assignment)

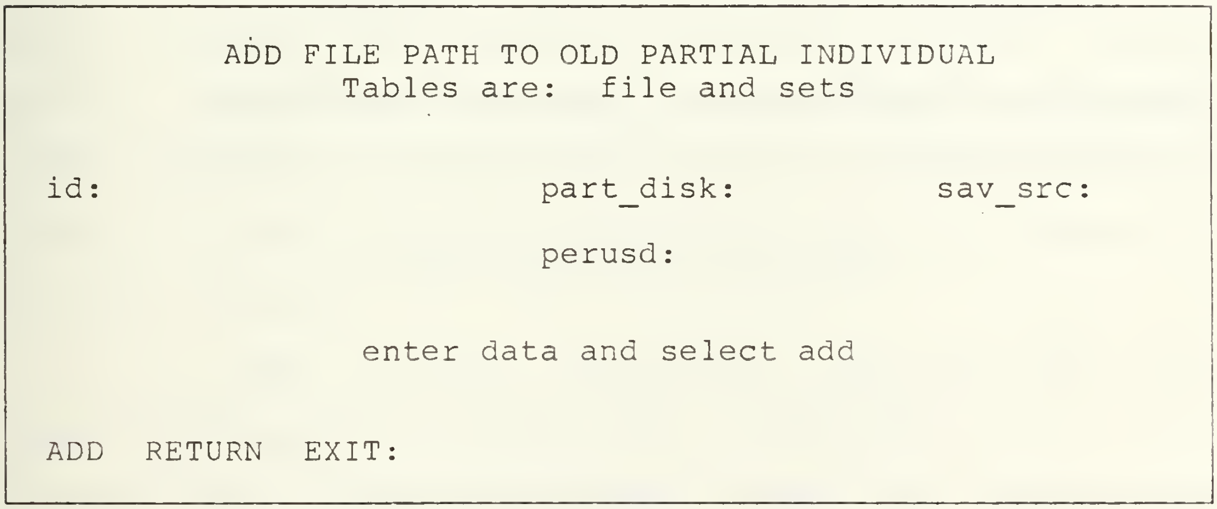
on the first field in which data may be entered when the form is initially displayed. Before entering new data, the id number must be assigned. The command line at the bottom displays operations ID, ADD, and RETURN.

To get an id assigned, first move the cursor to the command line by depressing <ESC>, then enter i RETURN. An id and perhaps other data will be assigned. Relevant notes regarding the labelling of cabinet spaces and/or tapes will be displayed next to the command line for two seconds. If the display says OLD ID, it means the cabinet space has been previously labelled, and the space is empty. Therefore, only the tape need be labelled. If the display states NEW ID, no spaces were empty and a new id has been added to the library. Both the space and the tape must be labelled.

After the id is obtained, the cursor will appear on the first field in which data may be entered. To add the data, move the cursor to the command menu line and select ADD. After the append has been completed the form will be redisplayed. All fields will contain the old values except id. A new record may be entered which has all the same values, except id, by simply selecting ID, then ADD, or a new record may be entered by selecting ID and typing over the displayed values, then select ADD. When all new records have been added, select RETURN to display the previous frame.

Frames which display programmed id assignment forms are 1.3.1 and 1.3.2.

b. Another type of append frame is the programmed append, shown in Figure 15.



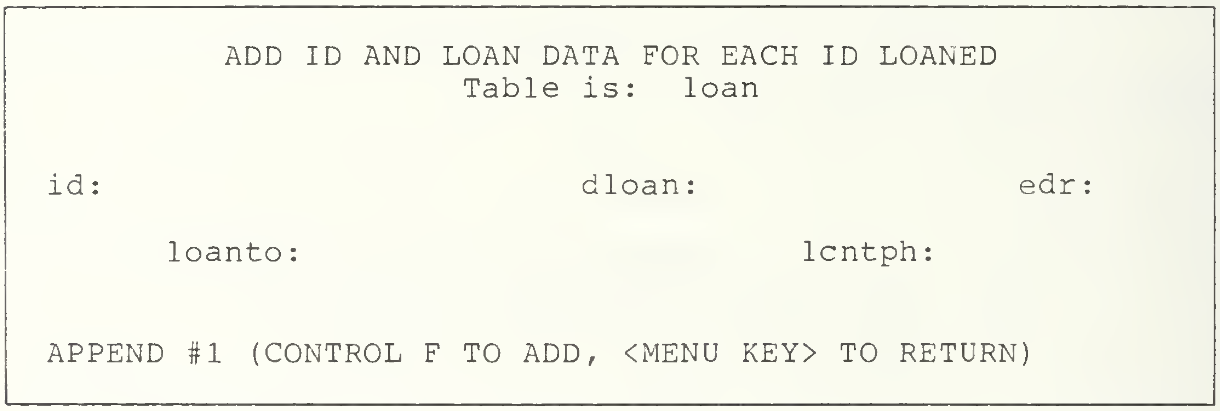
```
            ADD FILE PATH TO OLD PARTIAL INDIVIDUAL
                  Tables are:  file and sets


id:                         part_disk:          sav_src:

                            perusd:


                   enter data and select add


ADD  RETURN  EXIT:
```

Figure 15. Append Form (Programmed Append)

In the upper portion, the form displays fields into which data may be entered. It displays the command menu ADD, RETURN, and EXIT at the bottom. When the form is initially displayed, the cursor will always appear on id. To add new records, simply enter new data and select ADD.

Frame 1.3.4.2 displays a programmed append form.

c. The third type of append form is a QBF-append. The initial display of a QBF-append form is shown below in Figure 16.



```
              ADD ID AND LOAN DATA FOR EACH ID LOANED
                          Table is:  loan


id:                          dloan:                   edr:

      loanto:                            lcntph:


APPEND #1 (CONTROL F TO ADD, <MENU KEY> TO RETURN)
```

Figure 16. QBF-append Form

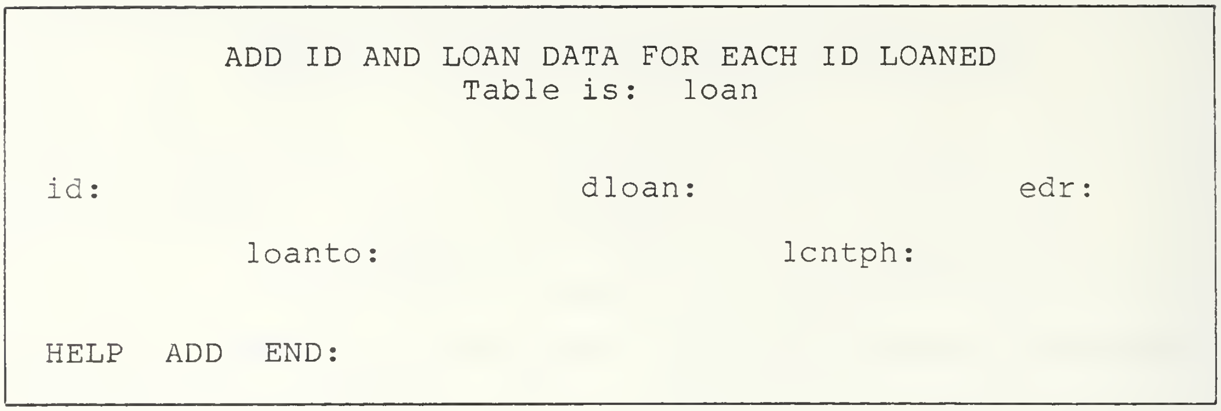
To add data, fill in all the data fields displayed in the upper portion and depress control F; the cursor does not need to be on the command menu line. After the new record has been entered the following form (Figure 17) will be displayed.

```
                 ADD ID AND LOAN DATA FOR EACH ID LOANED
                             Table is:  loan


 id:                              dloan:                     edr:

           loanto:                           lcntph:


 APPEND #2 (CONTROL F TO ADD, <MENU KEY> TO RETURN)
```

Figure 17. QBF-append Form

In some forms, some of the fields will retain the previous values. If the previous value is not displayed, it may be recalled by putting the cursor in the field and depressing control a. This will be especially helpful if you cannot remember the number of the last id entered. To add another record, simply change the appropriate values and enter control f. If you do not desire to add more records, depress the menu key <ESC> and the form shown in Figure 18 will be displayed.

HELP provides information regarding QBF. ADD allows you to continue adding data. If END is selected, the previous form will be displayed. Frames which display QBF-append forms are 1.3.3.1, 1.3.3.2, 1.3.4.1, 1.3.5, and 1.3.6. More detailed information regarding QBF-append frames is provided in the Ingres QBF Users Guide [Ref. 6].



ADD ID AND LOAN DATA FOR EACH ID LOANED
Table is: loan

id:	dloan:	edr:
loanto:	lcntph:	

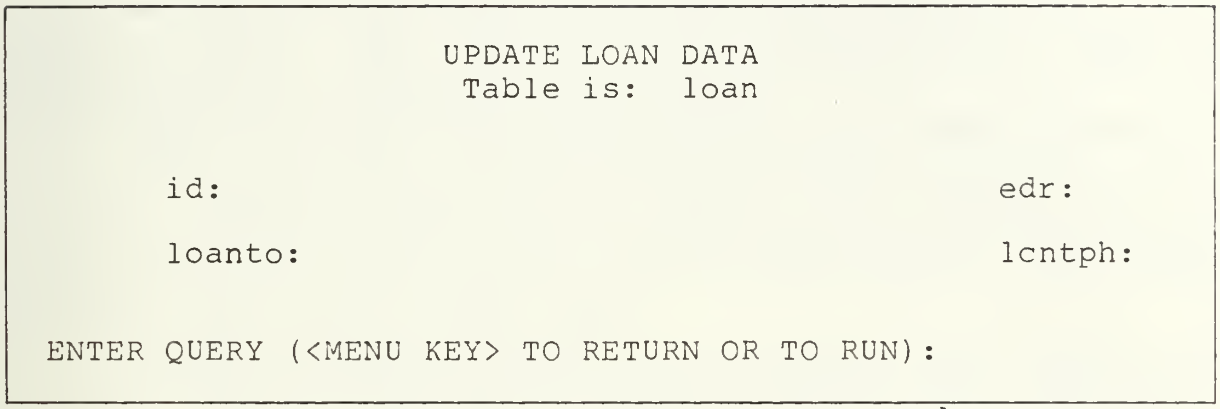
HELP ADD END:

Figure 18. QBF-append Form

5. Update Forms

Update forms are used to change values of records previously entered in the library. The update function consists of two states, the QUERY state and the GO state. While in the QUERY state, you may specify a query by filling in the form with the values of the records you are searching for. After filling in this form, QBF enters the GO state, retrieves the row(s) you asked for, and allows you to update each row, one by one. The changes are stored in a temporary buffer, as you move from row to row. After changing all the rows, you can write out the buffer containing the changes to the table. After you update the table, you are returned to the QUERY state to execute a new query. You can leave the QUERY or GO states at any time, either to execute another query or to return to the main menu of the operating system.

QBF-update frames display the form shown in Figure 19 when selected.



```
                     UPDATE LOAN DATA
                      Table is:  loan


    id:                                           edr:

    loanto:                                       lcntph:


 ENTER QUERY (<MENU KEY> TO RETURN OR TO RUN):
```

Figure 19. Update Form (Query State)

This is the QUERY state. Fill in the value or values of the records you wish to retrieve. For the TLS, you will usually call records for update by id. You may use comparison operators when specifying a query. These are:

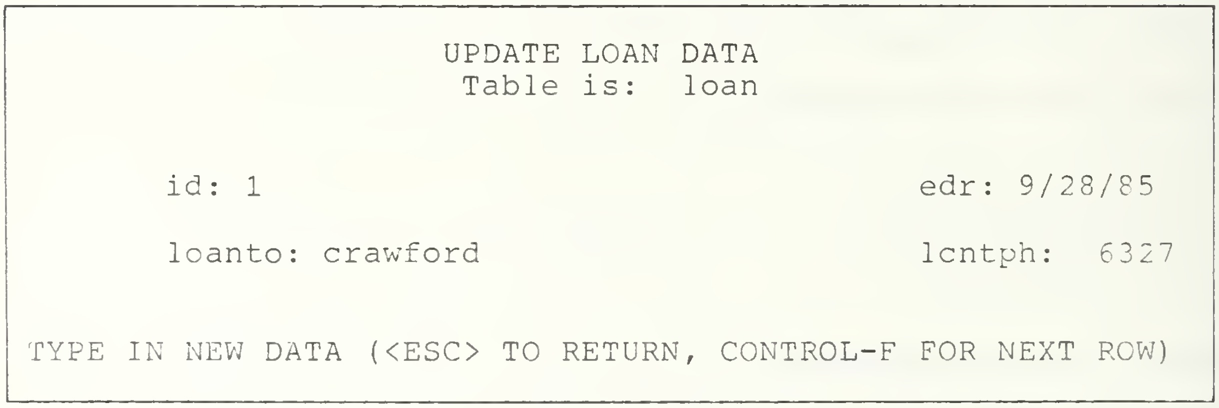
> greater than	>= greater than or equal to
< less than	<= less than or equal to
= equal to	!= not equal to

While in the QUERY state, QBF does not check fields to make sure they contain valid data. This is done when you enter the GO state to run the query.

The QUERY state has a menu for running queries, returning from the function, restarting the query, and getting help. To call the menu, depress the <ESC> key. The menu appears at the bottom of the screen.

HELP QUERY GO END <command>

You may call the menu at any time; it does not affect your data. If the menu was accidently called, depress RETURN to return to your form. To exit UPDATE, select the END command and you will be returned to the previous frame. If you select QUERY, QBF clears all the fields on your form so that you can enter a new query. All data on the old form is lost. If you select GO the query is run. QBF enters the GO state. QBF will now display the following form (Figure 20) with the values of the record(s) you requested. You can now edit the rows you have just asked for by typing the new data over the old.



```
                         UPDATE LOAN DATA
                          Table is:  loan


       id: 1                                      edr: 9/28/85

       loanto: crawford                           lcntph:  6327


 TYPE IN NEW DATA (<ESC> TO RETURN, CONTROL-F FOR NEXT ROW)
```

Figure 20. Update Form (GO State)

If no rows which satisfy the query are found, QBF displays the message - NO ROWS FOUND - and returns to the QUERY state so you may enter a new query.

If you wish to go to the next row specified by your query, depress the control F key. If no rows are left, NO MORE ROWS LEFT IN QUERY, is displayed.

At this point you have three options: (1) write your changes to the table; (2) start a new query; or (3) exit UPDATE, using the GO state menu.

Your changes are stored in a buffer each time you depress the control f key. These changes do not change the table in the library until you issue the WRITE command in the GO state menu.

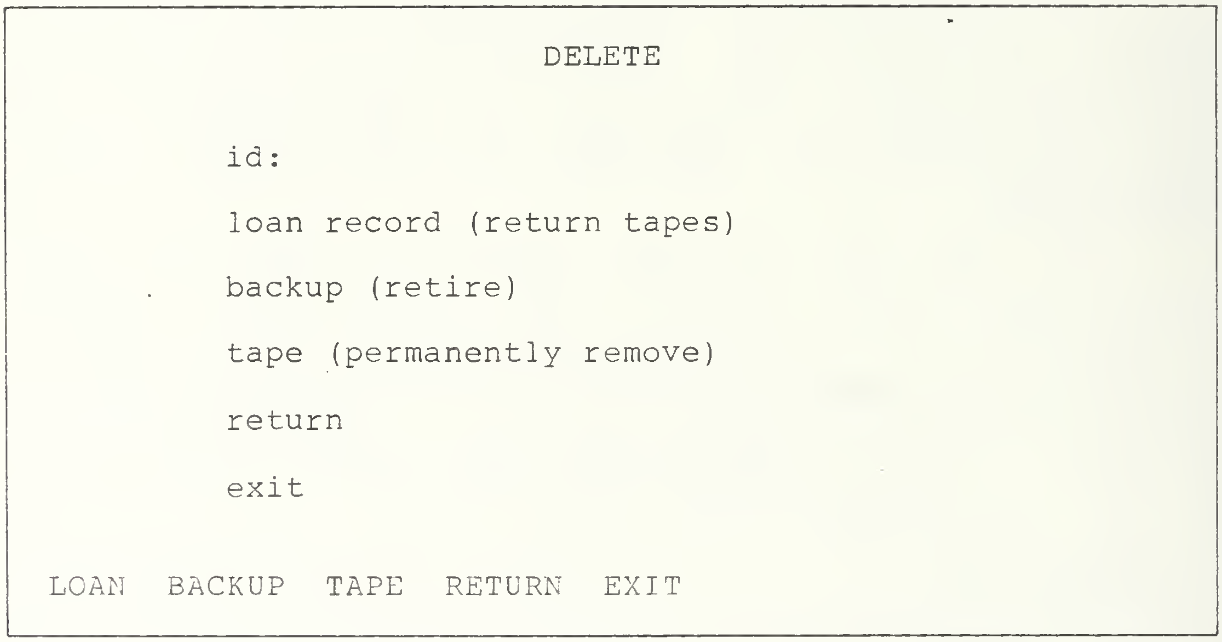
To enter the GO state, depress the <ESC> key. The following menu will be displayed: HELP QUERY WRITE DELETE END. You may call this menu at any time. It does not affect your data. To exit without writing your changes, select END. You will be returned to the previous frame. If you want to begin your query again, select QUERY. The changes made will be erased from the buffer and the QUERY form will be displayed. The DELETE key deletes the current row displayed on the screen. There is no reason to use this key in the library.

To write the changes you have made to the row(s), select the WRITE operation. If you have made changes to data and do not select the WRITE operation, the following message is displayed: DO YOU WANT TO LEAVE UPDATE WITHOUT WRITING CHANGES? Enter y for yes and n for no.

Further information about UPDATE is provided in the Ingres QBF User's Guide [Ref. 6]. QBF UPDATE frames are 1.4.1, 1.4.2, and 1.4.3.

6. <u>Delete Form</u>

The Delete Form (Figure 21), is used to delete records.



```
                          DELETE

          id:

          loan record (return tapes)

          backup (retire)

          tape (permanently remove)

          return

          exit

 LOAN  BACKUP  TAPE  RETURN  EXIT
```

Figure 21. Delete Form

On the upper portion, a field for id and a list of operations to be performed on the id number is displayed.

The command menu repeats the operations listed on the upper portion. Additionally, RETURN calls the previous frame; EXIT exits the application.

To use the delete frame, enter the id number of the tape on which the delete operation is to be performed. Move the cursor to the command menu and select the delete operation desired. The delete occurs immediately.

Frame 1.5 displays a delete form.

B. THE TLS APPLICATION

Each of the librarians has been granted access to the library. It may be called from UNIX by entering (% is the UNIX prompt): % cslib

The first TLS frame will display a form on the screen. The name of the frame is Topframe and is number 1 in Appendix D. Topframe is the main menu of the application. It displays the major operations performed in the library:

(1) TQuery/Report - print a report to the terminal

(2) PQuery/Report - print a report to the librarian's UNIX workspace.

(3) Add - add new records

(4) Update - correct incorrect values

(5) Delete - delete records

1. TQuery/Report (1.1)

If TQuery/Report is selected, a choice of the following reports is displayed on the terminal:

- Scratch (1.1.1) - for scratch tapes available.
- Periodic (1.1.2) - for aperiodic backups stored in the library (daily backup (db), weekly backup (wb), monthly backup (mb), or graduation backup (gb).
- Individual (1.1.3) - backups belonging to individual staff (is), faculty (if), or students (iu) (before they graduate).
- Vendor (1.1.4) - vendor (v) tapes, no matter to whom they belong.
- Student (1.1.5) - graduated students and the tapes on which their accounts are logged.
- Loaned (1.1.6) - a choice of Loaned Scratch Tapes (1.1.6.1) or Loaned Backup or Vendor Tapes (1.1.6.2) (Date to Retire (1.1.7) tapes by retirement date).

2. PQuery/Report (1.1.2)

If PQuery/Report is selected, a choice of the same reports as in TQuery/Report, and one additional report is displayed. The additional report frame available is DUMP.

- Dump (1.2.8) - Dump provides a choice of five reports. They are each a "dump" of one of the five tables: Tape (1.2.8.1), Sets (1.2.8.2), File (1.2.8.3), Student (1.2.8.4), and Loan (1.2.8.5). They contain all of the fields in the tables and are sorted by id number.

The reports selected by the PQuery/Report frame will be written to your UNIX workspace file. The name of the file is displayed on the terminal. The file may then be printed by using UNIX.

3. Add (1.3)

Add should be selected whenever you desire to add any new records to the library. Add provides a choice of adding the following:

- o Scratch (1.3.1) - to add new scratch tapes
- o Vendor (1.3.2) - to add vendor tapes
- o Backup (1.3.3) - to be used when a backup has been read to a scratch tape. Backup provides a choice to add:
  - Periodic backups (1.3.3.1) (type db, wb, mb, gb) or
  - Individual backups (1.3.3.2) (type is, if, iu)
- o Files (1.3.4) - is to be used to record the file paths which have been read to each tape. A choice of situations is provided:
  - New Backup or Vendor (1.3.4.1) - This frame will be used to add file paths at all times except the following
  - Old Partial Individual (1.3.4.2) - This frame will be used only when an individual has requested a file path be read to tape and the librarian selects one of the individual's old partially used tapes to perform the read. The new file path should be recorded using the id of the old tape.

4. Update (1.4)

Update should be selected whenever you desire to change values which have been previously entered in the library. Update provides three choices for the changes:

- o Errors (1.4.1) - After a tape is read or written, the utility may indicate that the number of tape errors is different than those recorded in the library. Use Errors to update the number of errors and the date the new errors are entered.

- o Backup (1.4.2) - Backup enables updates to the following fields concerning backup or vendor tape records:

  dcreate, dretire, owner, advisor, contph, descr, sequ.

- o Loan (1.4.3) - Loan allows you to change fields edr, loanto, and lcntph.

5. Delete (1.5) - Delete is the only frame which should be used to delete records from the library. The operations available are:

- o Loan - Use when a loaned tape is returned. This operation deletes the record of the loan.

- o Backup - Use when a backup tape is retired or erased, but the tape is retained to be used as a scratch. All records of the backup are deleted from the library. The data regarding the physical characteristics and condition of the tape are retained. The tape keeps its id number, and must be left in its cabinet space.

- o Tape - When a tape is permanently disposed. All records of the tape are deleted from the library. Only the id number of the cabinet space is retained and the TLS will define the id number as an empty slot. The tape must be removed from the cabinet space, and the id number must be removed from the tape. The cabinet space retains the id number.

C. MAINTENANCE PROCEDURES

As discussed in Chapter IV, Section J, the database administrator will be required to perform tasks to maintain peak performance, integrity, and the security of the database. The following is recommended.

1. <u>Peak Performance</u>

On a monthly basis, run a system modification on each table in the database. This may be done by entering the following at the UNIX operating system level (% UNIX):

```
% sysmod talibrary
% sysmod talibrary tape sets file student loan
```

2. <u>Integrity</u>

Only the librarians should be granted permits to affect the TLS tables. Destroy and define permits to tables as personnel are relieved of their librarian duties and new personnel are assigned. The procedure for granting and destroying permits is provided in Appendix F, Section D.

3. <u>Security</u>

Only librarians should have access to the library. Access is made available by defining the TLS login symbol in the login files of new librarians. The login sign should be deleted from the login files of personnel relieved of librarian duties. The procedure for providing and denying access is discussed in Appendix F, Section C.

REFERENCES

1. Ingres Self Instruction Guide, VAX/UNIX Version 1.1, October 1982

2. Ingres Reference Manual, Version 2.1, VAX/UNIX, Relational Technology, Inc., July 1984

3. Kroenke, D., Database Processing, Science Research Assoc., Inc., 1983

4. DeMarco, T., Structured Analysis and System Design, Yourdin, Inc., 1979

5. Boehm, B. W., Software Engineering, IEEE Transaction on Computers, December 1976

6. Ingres QBF User's Guide, Version 2.1, VAX/UNIX, Relational Technology, Inc., July 1984

7. Ingres Applications by Forms User's Guide, Version 2.1, VAX/UNIX, Relational Technology, Inc., July 1984

BIBLIOGRAPHY

Chen, P., The Entity-Relationship Model-Toward a Unified View of Data, ACM Transactions on Database Systems, Volume I, No. 1, March 1976

Date, C. J., An Introduction to Database Systems, Addison-Wesley Publishing Company, 1981

Dolan, K., Business Computer Systems Design, Yourdin, Inc., 1979

Orr, K., Structured Systems Development, Yourdon Press, 1977

Page-Jones, M., The Practical Guide to Structured Systems Design, Yourdon Press, 1980

Yourdon, E., Techniques of Program Structure and Design, Prentice Hall, Inc, 1975

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Assistant Professor Daniel R. Dolk, Code 54Dk Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5100	1
4.	LCDR Barry Frew, Code 54Fw Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5100	2
5.	Mrs. Sue Walen, Code 52 Computer Science Department Naval Postgraduate School Monterey, California 93943-5100	6
6.	Commanding Officer Military Sealift Command Office Korea APO San Francisco, California 96259-0264	2
7.	Curricular Officer, Code 37 Naval Postgraduate School Monterey, California 93943-5100	1
8.	Willis R. Greer, Jr., Code 54Gk Chairman, Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5100	1
9.	Mrs. Marjorie J. Crawford 758 NE 47th Court Pompano Beach, Florida 33064	1

10.	Mr. Bruce MacLennan, Code 52 Chairman Computer Science Department Naval Postgraduate School Monterey, California 93943-5100	1
11.	Nita Raichart 288 B Central Avenue Pacific Grove, California 93950	1
12.	Ms. Alyce Austin 801 Ocean Avenue Monterey, California 93940	1

609 - 197