AN IMPLEMENTATION OF INVENTORY SYSTEM FOR LAFENE HEALTH CENTER BASED ON DBASE III PLUS

by

SOUDABEH FARZBOD

B.S. Tehran College of Business. 1971

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1987

Approved by:

Major Professor

D
2668
R4
.MSC
987
c. 2

A11207 301636

TABLE OF CONTENTS

Chapter | Page

Appendix

Digitized by the Internet Archive
in 2012 with funding from
LYRASIS Members and Sloan Foundation

http://archive.org/details/implementationof00farz

INTRODUCTION

For several years now, The Lafene Health Center at Kansas State University has been in need of an automated inventory system. Such a system would simplify the process of consumable supplies inventory keeping, and also would simplify the ordering process.

Before the computerization of consumable supplies inventory, keeping an accurate and current count of items was tedious and time-consuming. Each item was listed on a page in an "inventory book". When an item was received, the quantity was written in the book and manually added to the total. When an item was used, it was recorded and deducted from the inventory total. The inventory book also contained the information needed for ordering: company name, prices, stock number, units/sizes available, contract item numbers, and amount usually ordered. However, the company address and phone number had to be looked up elsewhere. The minimum amount that is needed to be maintained was also in the inventory book, but many times items that were at their minimum were overlooked, creating shortages in necessary supplies.

It took at least two people inputting the information because it was done on a daily basis. The person in charge

of the inventory did not always have time to do this every day because of other duties. Therefore, a back-up person was needed.

Information about incoming supplies was taken from packing slips; data regarding supplies used were taken from requisitions which were filled out by personnel throughout the building. The requisitions were then totalled, recorded in a "monthly expenditure book" by department, and by object code (each item is assigned an object code according to the nature of the item). At the end of each month the total of each object code for each department was totalled on the calculator.

It is the intent of this Master's Report to provide a computerized inventory system which speeds up this whole process considerably. This system is able to produce a report of all those items that are at their minimum, thus minimizing the risk of overlooking supplies that need to be reordered. By using this system, an employee is able to call an item up to the screen and get all the information, such as: name, size, quantity that is usually ordered, price, stock number, contract number, company name, address, phone number, etc. This also will help them greatly in their bookkeeping and finding each departments' expenditures by object code.

In summary, specific system problems are:

-time consuming process.

-need more than one person to do it daily.

-human error, many times the items that were at their minimum were overlooked, creating shortage in necessary supplies.

-keeping information in several different books.

-making report, slow process, not accurate.

-difficulty in bookkeeping.

Objectives for the automatic system are:

1) daily report of items that are at their minimum and needs to be ordered.
2) access to complete ordering information.
3) calculates and lists department expenditures by object code.
4) list out how much has been spent by total unit in each object code.
5) much more rapid input of inventory information.

CHAPTER 1

The Relational Model

1. A Historic Perspective.

The rational model, which was introduced by E. F. Codd in 1970, formalized the separation of the user view of data from its eventual implementation. Since the introduction of this model, there have been many developments in its theory and application. The early idea of normal form has been extended to include additional criteria, especially fourth normal form, which was introduced in 1977 by R. Fagin.

Relational models have many desirable characteristics. Unlike the hierarchical and network models that are structured and tied to graph theoretic notations; the relational model is an unstructured model based upon set theoretic notations. The importance of this model lies in the way that relationships are represented. The relationships among relations in the data base or among tuples in a relation are embodied in the data itself, thus eliminating the need for external pointers of set relationships.

2. The Basic Structure.

A data base is made up of any number of relations. Each relation is simply a two-dimensional table that is made

up of a number of rows and columns. Each column is called an attribute. A relation that has n columns or n attributes is said to be of degree n. The rows of the relation are called tuples and contain the data. Each attribute has a domain. A domain is a set of values that the attribute can have and it may appear in more than one relation or sometimes more than once in the same relation. It is common to choose domain names to signify value sets; for example character, integer, and so on, are domain names. Attribute names, however, are chosen to be meaningful within the context of the enterprise. The use of such meaningful names adds to the clarity of the relational representation. Each relation possesses the following properties.

1. There is one column in the relation for each attribute of the relation. Each such column is given a name that is unique in the relation.
2. The entries in the column come for the same domain.
3. The order of the column or attributes in the relation has no significance.
4. The order of the rows is not significant.
5. There are no duplicate rows.

3. Keys

A key is the attribute or set of attributes that uniquely identifies tuples in a relation. Each key has three properties for all time and for any instance of the relation.

1. Uniqueness - The set of attributes takes on a unique value in the relation for each tuple.
2. Nonredundancy - If an attribute is renamed from the set of attributes, the remaining attributes do not possess the uniqueness property.
3. Validity - No attribute value in the key may be null.

The relation key is often called the "candidate key." If a key is the only key of the relation, it is generally referred to as the "primary key."

4. Normal Form.

With some relations, changing data (insertion, deletion, update) can have unexpected consequences. These consequences are called modification anomalies and they are not desirable. Relational schemes are normalized to incorporate desirable properties in the data base. The two techniques of normalization are synthesis of the data base as proposed by Bernstein (1976) and by decomposition proposed by Codd.

CHAPTER 2

Data Base Design

A data base system is essentially nothing more than a computerized record keeping system whose overall purpose is to maintain information and to make that information available on demand. The information can be anything that is important to the user. There are no standards for system analysis tools, forms or languages for data base design. A major aim of the initial system analysis effort is to arrive at a conceptualization of the data base, independent of the hardware and data model.

In designing the data base for Lafene Health Center the Unger & Fisher method was used. The steps in this methodology are:

1. Predesign evaluation
2. Information modeling
3. Semantic modeling
4. Logical DB design
5. Cost/Benefit analysis
6. DBMS selection; physical design/implementation

Predesign evaluation.

The first step is predesign evaluation. At this stage we should find out the answer to: what are the functions performed by this enterprise, what forms do they process, and what problems do they have?

The technique used was to interview the user, management, and key employees involved in keeping inventory in Lafene Health Center. The output of this stage were functional specification.

Information modeling.

At this stage all the documents associated with inventory were collected and analyzed. At this time we were looking at the relationships between things and to determine 1:1, 1:n, n:m relationships. Determine functional dependencies and also determine keys. The outputs from this step were Data Dictionary and Functional Dependencies. Data Dictionary is defined by ordered collection of data element descriptions containing specific identification attributes. Functional dependencies are semantic constraints that represent relationships among collections of data in the real world and constrain the tuple values possible in a relation.

Semantic modeling.

The two previous steps help in understanding what the firm is all about. In this step we try to create a model of the firm and show where the information flows in this firm.

At this point a meeting is necessary with key personnel and the results of the document analysis should be discussed. The purpose of the meeting is to make sure that the flow and semantics of the data elements and their potential values have been correctly understood.

Logical db design.

Using the entities and relationships we can design the logical model. The results of this stage is an entity relationship diagram.

These are the steps used in arriving at the entity-relationship diagram:

1. Selection of entities.
2. Selection of relationships between entitites.
3. Selection of entity attributes.
4. Identification of key attributes for entities.

An entity represents a real world concept about which information is recorded; a relationship is an explicit indication of how an entity is related to another. The relationship is as important and as definable as any entity or attribute of an entity. First we should identify and understand each entity then logically relating them to one another.

An entity is a distinguishable object of some particular type such as supplier or department. Entities of the same type are classified as entity sets. In the ER Diagram, rectangles represent entity sets.

For Lafene Health Center inventory system five entities were necessary.

1. Vendors
2. Item received
3. Item inventory
4. Item used
5. Department

The data base designer's responsibility is to identify the relationship sets of interest to the enterprise. Different types of relationship may exist between different types of entities. A relationship set is a set of relationships of the same type. In the entity-relationship diagram, a

relationship is represented by a diamond-shaped box with lines connecting the related entity sets.

A 1:n relationship exists between item inventory and vendor. Each item is purchased from one vendor and a vendor may supply many items, so their relationship is one to many. Many to many relationship is between department and item inventory. Many departments might use one item. Also, one department can use many items.

Entities are described by attributes that provide detailed information about the entity. One or more of the attributes will serve as an identifier (key) to distinguish different instance of the entity.

Selection of Key Attributes for Entities

An entity is a real world concept that is of importance to the organization for which the data base is being designed. Information is gathered and recorded for each instance of the entity. It is vitally important to be able to uniquely identify each instance of an entity. In most cases, entities will become relations in the relational data model; therefore, like a key in a relation, each entity must have an "entity identifier" selected to positively identify each instance of the entity. This is accomplished by selecting an attribute or a combination of attributes that will contain unique values for each instance of the entity.

FUNCTIONAL SPECIFICATIONS

The Functional Specifications of different functions in the Lafene Health Center inventory require many inputs. Some of these inputs have many options (such as item name), therefore, it is not possible to have a menu-driven system to input the data item. It is more feasible to have the user input these fields manually.

1. Function : Create Item Inventory

   Description: This function creates or adds new records for new items in the inventory master file and also reindex the whole master file after adding all the new items.

   Input : Object Code

   Input : Item Number (stock no)

   Input : Item Name

   Input : Item Description

   Input : S_balance

   Input : L_balance

   Input : Flag

   Input : Vendor_Name

   Input : Brand

   Input : Price

   Input : Unit

   Output : On Line Command

   User : Person in charge of the inventory

   Use : On necessity

2. Function : Delete an Item Inventory

Description: This function deletes the record associated with the item that is to be deleted for some reason, i.e. not using it any more, or no longer available in the market.

Input : Item Name

Output : On line error message such as 'No Such Item Name Found!'.

Output : Shows the whole record to be sure that it is the right one to be deleted.

Output : On line message after deleting any record.

User : Person in charge of the inventory

Use : On necessity

3. Function : Update the Item Record

Description: This function modifies or edits the master inventory file. If for any reason there should be a change in any field it can be done by this function.

Input : Item Name

Output : The whole record would be available to user for any change.

Output : On line error messages

User : Person in charge of the inventory

Use : On necessity

4. Function : Add New Department

Description: This function adds new records for new departments. It will reindex the whole department file after adding all the new departments.

Input : Department Name

Input : Department Code

Input : 'Zero' to the total due

Output : On line error message if it is a duplicate

User : Person in charge of the inventory

Use : On necessity

5. Function : Delete a Department

Description: This function deletes the record associated with the department that is to be deleted.

Input : Department Name

Output : On line error message such as "No Such Department Name Fund!".

Output : On screen, all the information about the department to be deleted, to make sure that it is not the wrong department.

User : Person in charge of the inventory

Use : On necessity

6. Function : Update Department Record

Description: This function edits or modifies the department files for necessary changes. New information immediately replaces the old information.

Input : Department Name

Output : On screen, all the information about the department available to user for any changes.

Output : On line error messages

User : Person in charge of the inventory

Use : On necessity

7. Function : Add New Vendor

Description: This function adds new records to the vendor file. It will reindex the whole vendor file after adding new vendors.

Input : Vendor Name

Input : Address

Input : City

Input : State

Input : Zip Code

Input : Phone

Input : FEIN No

Input : Customer No

Output : On line command

User : Person in charge of the inventory

Use : On necessity

8. Function : Update or Edit a Vendor Record

Description: This function edits or modifies the vendor record if for any reason it needs to be changed, i.e., the address of the vendor changed.

Input : Vendor Name

Output : On screen, all information about the vendor is available to user for any changes.

Output : On line error messages

User : Person in charge of the inventory

Use : On necessity

9. Function : Deleting a Vendor

Description: This function deletes the record associated with the vendor that is to be deleted.

Input : Vendor Name

Output : On line error messages. Such as "No Such Vendor Name".

Output : On screen, all the information about the vendor to be deleted. Needs user confirmation before deleting.

User : Person in charge of inventory.

Use : On necessity

10. Function : Restock

Description: This function keeps a record of receipt of new order and updates the inventory. As the items are purchased, the quantity of those items is increased.

Input : Vendor Name

Input : Date of Receipt

Input : Quantity received as Qty_In

Input : Item Name

User : Person in charge of inventory

Use : Daily

11. Function : Consumption

Description: This function keeps a record of each item consumed by the department and updates the inventory according to the consumption. Furthermore, it calculates the tot_due, updates the record, and the department expenditures.

Input : Object Code

Input : Department Code

Input : Quantity consumed as Qty_Out

Input : Date of consumption

Input : Item Name

Input : 'Zero' to total due

User : Person in charge of the inventory

Use : Daily

12. Function : Inventory Item at Minimum

Description: This function produces a list of all the items in the inventory that are at their minimum and need to be reordered. It prints name and price of the item, vendor name, address, and phone.

Input : Query

Output : Message containing list of items and their price and their supplier's information.

User : Person in charge of the inventory

Use : Daily

13. Function : Department Expenditure by Object Code

Description: This function gives the expenditure of department by object code and total expenditure for the department.

Input : Department Code

Input : Date (starting date)

Input : Date (ending date)

Output : List consists of department name, code, interval date, object codes and amount of expenditure, and total of the expenditure by the department.

User : Person in charge of accounting

Use : Monthly

14. Function : Item Consumed by Department.

Description: This function shows items consumed by department by day.

Input : Department Name

Input : Day

Output : List consists of date, department and items consumed.

User : Person in charge of the inventory

Use : On necessity

16. Function : Items Consumed by Day

Description: This function produces a report showing amount of each item that is consumed by day.

Input : Date

Output : Items and amount of consumption of that day

User : Person in charge of accounting

Use : Daily

17. Function : List of Vendors.

Description: This function lists all the vendors names along with all the information about them.

Input : Command

Output : List of all the vendors with pertinent information

User : Person in charge of the ordering items

Use : On necessity

18. Function : List of Departments.

Description: This function lists all the departments along with their total_due.

Input : Command

Output : List of departments with their total expenditure

User : Person in charge of the accounting

Use : On necessity

DATA DICTIONARY

One of the most important DBM tools is the data dictonary. The data dictionary is effectively a data base in its own right; a database that contains "data about data".

The dictionary for the Lafene Health Center Inventory System specifies the attribute name, alias(es), type, format, domain, frequency of use, availability, and FD_On owner.

Name	=	Object_Code
Alias(es)	=	Ob_Code
Description	=	Group of objects or group of items in the inventory will be recognized with the object code. The number refers to a particular inventory item.
Type	=	Character
Format	=	x(3) or xxx
Domain	=	(221 - 361 - 369 - 371 - 392)
Frequency	=	On necessity
Availability	=	On demand
FD-ON Owner	=	Person in charge of inventory

Name	=	Item_Number
Alias(es)	=	Item_No
Description	=	Represents the stock number for the item, shows where items are stocked in the inventory.
Type	=	Character
Format	=	xxxxxx
Domain	=	000001 to 999999
Frequency	=	On necessity
Availability	=	On demand
FD-ON Owner	=	Person in charge of inventory

Name	=	Item_Description
Alias(es)	=	Item_Descr
Description	=	Describes the item.
Type	=	Character
Format	=	x(20)
Domain	=	Alphabetic string of length 20
Frequency	=	On necessity
Availability	=	On demand
FD-ON Owner	=	Person in charge of inventory

Name = Item_Name

Alias(es) = Name of the item

Description = Name of each item

Type = Character

Format = x(30)

Domain = Alphabetic string of length 30

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of inventory

Name = Flag

Alias(es) = Min

Description = Gives the level of the item at which point the new item is reordered.

Type = Numeric

Format = x(5).99

Domain = 000.00 to 999.99

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Vendor_Name

Alias(es) = Vendor_Nam

Description = Name of the supplier.

Type = Character

Format = x(30)

Domain = Alphabetic string of length 30

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Unit

Alias(es) =

Description = Gives the amount of smaller unit which is inside the larger unit. For example, if there are five cases inside each box, then unit is equal to 5.

Type = Numeric

Format = x(6).99

Domain = 0000.01 to 9999.99

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Department_Name

Alias(es) = Dep_Name

Description = Gives the name of the department.

Type = Character

Format = x(12)

Domain = Alphabetic string of length 12

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Dep_Code

Alias(es) = Code

Description = The number refers to a particular department in the Health Center.

Type = Character

Format = x(3)

Domain = 001 to 017

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Total_Due

Alias(es) = Department expenditure

Description = Gives the total expenses for each department.

Type = numeric

Format = x(7).99

Domain = 00000.01 to 99999.99

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Vendor_Name

Alias(es) = Supplier_Name

Description = Gives the name of the vendor.

Type = Character

Format = x(30)

Domain = Alphabetic string of length 30

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Address

Alias(es) = Vendor_Address

Description = Refers to the number and street part of the address of the Vendor.

Type = Character

Format = x(30)

Domain = Alphabetic string of length 30

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = City

Alias(es) = Vendor_City

Description = The city part of the address of vendor.

Type = Character

Format = x(20)

Domain = Alphabetic string of length 20

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = State

Alias(es) = Vendor_State

Description = The state part of the vendor address.

Type = Character

Format = x(2)

Domain = Alphabetic string of length 2

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Zip

Alias(es) = Zip_Code

Description = The zip code part of the vendor address.

Type = Character

Format = x(10)

Domain = Alphabetic string of length 10

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Phone

Alias(es) = Vendor telephone number

Description = Refers to the telephone number of the supplier.

Type = Character

Format = x(18)

Domain = Alphabetic string of length 18

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = FEIN

Alias(es) =

Description = Federal employee identification number

Type = Character

Format = x(10)

Domain = Alphanumeric string of length 10

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Customer N

Alias(es) = Customer_Number

Description = The number that identifies Lafene Health Center for that vendor

Type = Character

Format = x(15)

Domain = Alphanumeric string of length 15

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Date_of_Rec

Alias(es) = Date_of_Receipt

Description = Date of receiving the new order.

Type = Date

Format = (../../..)

Domain = 01/01/01 to 12-31-99

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Qty_In

Alias(es) = New Order

Description = Refers to the amount of new order received.

Type = Numeric

Format = x(6).99

Domain = 0000.01 to 9999.99

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Qty_Out

Alias(es) = Amount of consumption

Description = Refers to the amount of item that was used (consumed) by the department.

Type = Numeric

Format = x(7).99

Domain = 00000.01 to 99999.99

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Date_of_Use

Alias(es) = Date_of_Consumption

Description = Date of the consumption.

Type = Date

Format = x(8)

Domain = 01/01/01 to 12/31/99

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

Name = Tot_Due

Alias(es) =

Description = Refers to the total cost of each consumption and is the amount of item consumed times its price

Type = Numeric

Format = x(7).99

Domain = 00000.01 to 99999.99

Frequency = On necessity

Availability = On demand

FD-ON Owner = Person in charge of the inventory

FUNCTIONAL DEPENDENCIES

A. Various Dependencies are:

1. Record Name : Item Inventory

   Attributes : 1. Ob_Code
   2. Item_No
   3. Item_Name
   4. Item_Descr
   5. S_Balance
   6. L_Balance
   7. Flag
   8. Vendor_Name
   9. Brand
   10. Price
   11. Unit

   Dependencies: Item_name, Item_no -----> Ob_code, Item_descr, S_balance, L_balance, Flag, Vendor_name, Brand, Price, Unit.

   Key : (Item_name, Item_no)

2. Record Name : Department

   Attributes : 1. Dep_Name
   2. Dep_Code
   3. Total_due

Dependencies: Dep_code -----> Dep_name, total_due

Dep_name -----> Dep_code, total_due

Key : (Dep_Code)

3. Record Name : Vendor

Attributes : 1. Vendor_name

2. Address

3. City

4. State

5. Zip

6. Phone

7. FEIN_No

8. Customer No

Dependencies: Vendor_name -----> Address, City, State, Zip, Phone, FEIN_No, Customer No

Key : (Vendor name)

4. Record Name : Item Received

Attributes : 1. Vendor_Name

2. Date_of_Receipt

3. Qty_In

4. Item_Name

Dependencies: Item name, Date of Receipt -----> Qty_In

Item name -----> Vendor name

Key : (Item name, Date of Receipt)

5. Record Name : Item Use

Attributes : 1. Ob_Code

2. Dep_Code

3. Qty_Out

4. Date_of_Use

5. Item_Name

6. Total_Due

Dependencies: Date of Use, Item_Name ----> Ob_Code

Dep_Code, Qty_Out, Tot_Due

Dep_Code -----> Total_Due

Dep_Code, Item_Name -----> Qty_Out

Key : (Date of Use, Item Name)

The following are the abbreviations for the various fields in the Bern 2 output:

OBJECT_CODE	OB_CODE
ITEM_NUMBER	ITEM_NO
ITEM_NAME	ITEM_NAME
ITEM_DESCRIPTION	ITEM_DES
S_BALANCE	S_BALANCE
L_BALANCE	L_BALANCE
FLAG	FLAG
VENDOR_NAME	VENDOR_NAME
BRAND	BRAND
PRICE	PRICE
UNIT	UNIT
DEPARTMENT_NAME	DEP_NAME
DEPARTMENT_CODE	DEP_CODE
TOTAL_DUE	TOT_DUE
ADDRESS	ADDRESS
CITY	CITY
STATE	STATE
ZIP_CODE	ZIP
PHONE	PHONE
FEIN_NUMBER	FEIN_NO
CUSTOMER_NUMBER	CUST_NO
DATE_OF_RECEIPT	DT_OF_RE
QTY_IN	QTY_IN

QTY_OUT	QTY_OUT
DATE_OF_USE	DT_OF_US
TOT_DUE	TOT_DUE

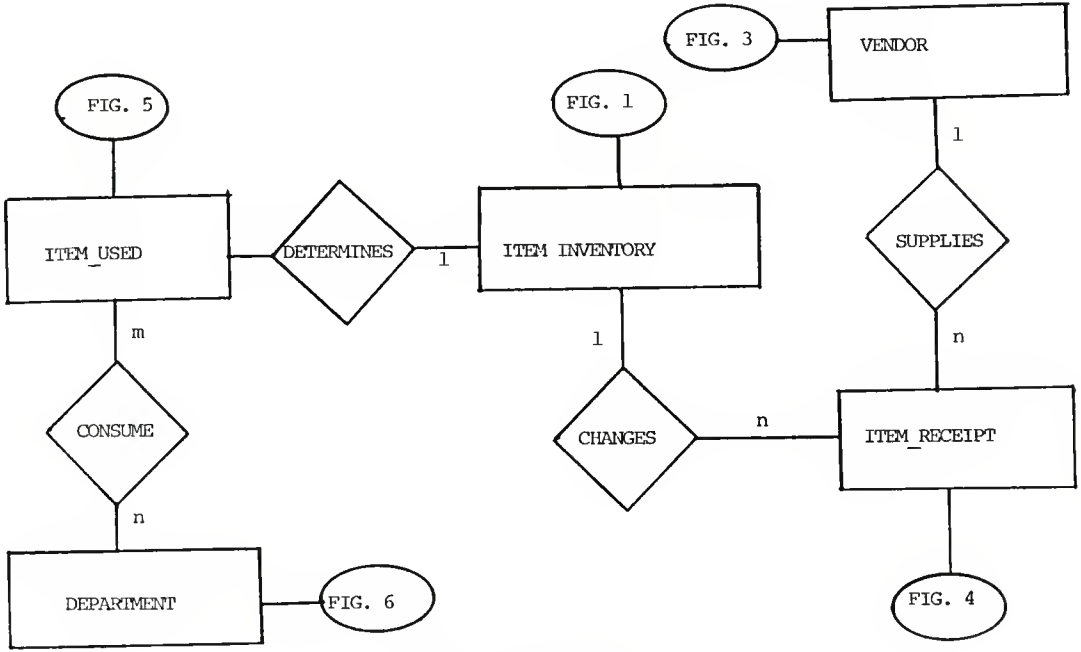

Fig. 1. E-R Diagram

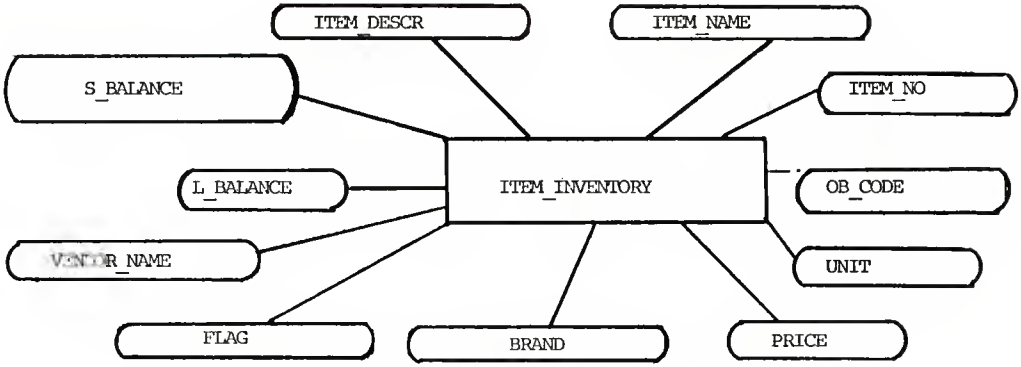
ITEM_DESCR
ITEM_NAME
S_BALANCE
ITEM_NO
L_BALANCE
ITEM_INVENTORY
OB_CODE
VENDOR_NAME
UNIT
FLAG
BRAND
PRICE

Fig. 2

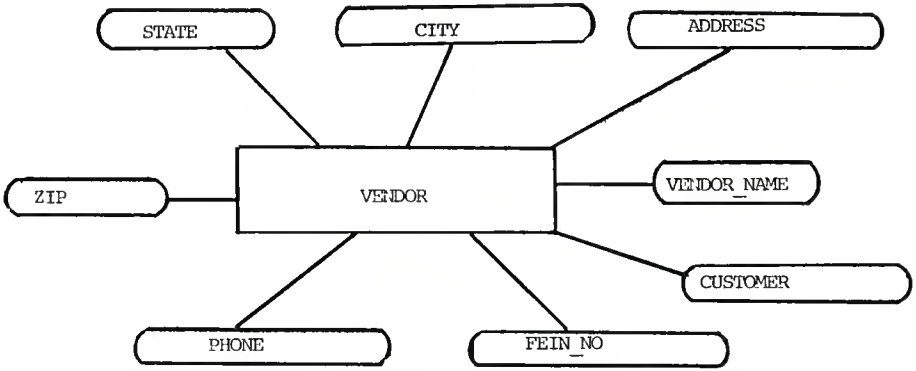
STATE
CITY
ADDRESS
ZIP
VENDOR
VENDOR_NAME
CUSTOMER
PHONE
FEIN_NO

Fig. 3

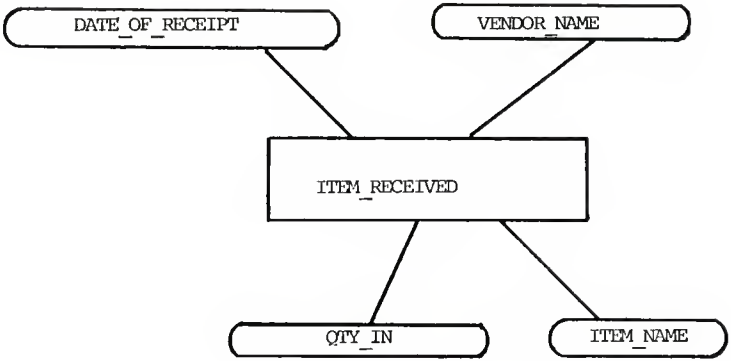

Fig. 4

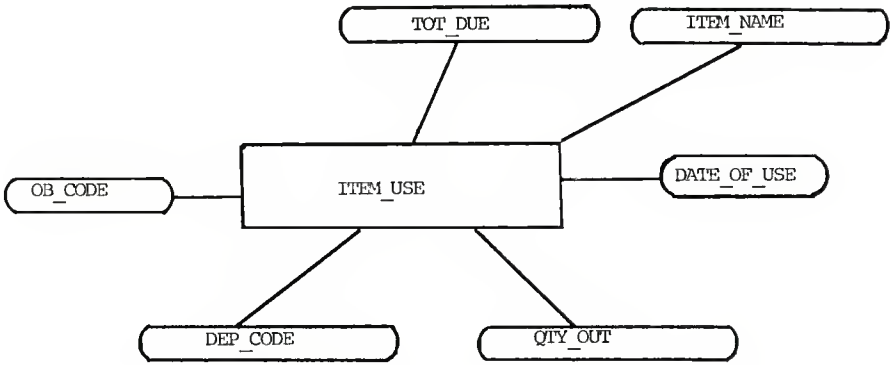

Fig. 5

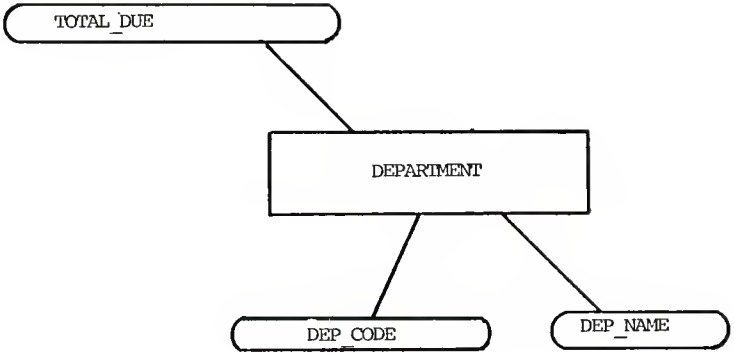

Fig. 6

RELATIONAL SCHEMA

The schema is the logical description of the data base. It includes the definition of the name and data type of each field making up each relation in the data base and also defines the relationships between the relations. Conversion from E-R Diagram to Relational Schema is shown in Figure 7.

1. VENDOR (Vendor_name, Address, City, State, Zip, Phone, Fein_no., Customer_no.
2. ITEM_RECEIPT (Vendor_name, Date of Receipt, Qty in, Item name)
3. SUPPLIES (Vendor name, Date of Receipt, Item name)
4. ITEM_INVENTORY (Object_Code, Item No., Item Name, Item_Description, S_balance, L_balance, Flag, Vendor_Name, Brand, Price, Unit)
5. CHANGES (Item Name, Date of receipt)
6. ITEM_USE (Object_code, Dep_Code, Qty_out, Date of use, Item name, Tot_due)
7. DETERMINE (Item name, Date of use)
8. DEPARTMENT (Department_name, Department code, Tot due)
9. CONSUME (Department code, Item name, Date of use)

Fig. 7. CONVERSION FROM E-R DIAGRAM TO RELATIONAL SCHEMA

CHAPTER 3

IMPLEMENTATION

The programming language environment which was selected to host the inventory management system on microcomputer is dbase III plus, a relational data base management system from Ashton-Tate which provides excellent programming capability as well as good native facilities for management of data on microcomputer. The system is available on a wide range of microcomputers and has a programming language designed to make the system into an application-development-system. Also, dbase III plus is well documented and supported, and uses a block-structured language in which modular and highly reliable code is produced.

One major advantage of the use of dbase III plus as the host system for the inventory control is that dbase III plus is designed as a user-friendly data management system. Its commands are well named to be indicative of their function and the manual which describes the dbase III plus system is quite clear. Also, dbase III plus can be regarded as a query/report language used to access the inventory information being maintained by the inventory management system.

The system as currently implemented, does not require any interfaces outside of the dbase III plus environment. The user communicates with the data bases through interactive menus or screen. Basically each screen is associated with a particular function and therefore is associated with a particular module.

System Files and Descriptions

In the inventory control system, the following data files are used and all of them are stored on the hard disk drive of a microcomputer.

Item Inventory File

This file maintains the item's quantity on hand and other necssary information. Each record contains the item object code, item stock number, item name, item description, the balance of both smaller unit and larger unit of item, the reorder level (Flag), name of supplier, brand, price, and the number of small units in the larger unit. These records are indexed on item names. The structure of record is presented in Figure 8.

Vendor File

This file contains the information about the vendor that supplies one or more items. See Figure 9 for the structure of vendor record.

Field	Field Name	Type	Width	Dec
1	Ob_Code	Character	3	
2	Item_No	Character	6	
3	Item_Name	Character	30	
4	Item_Descr	Character	20	
5	S_balance	Numeric	7	2
6	L_balance	Numeric	7	2
7	Flag	Numeric	5	2
8	Vendor_Nam	Character	30	
9	Brand	Character	15	
10	Price	Numeric	6	2
11	Unit	Numeric	7	2

Figure 8. Structure of Item Inventory Record

Field	Field Name	Type	Width	Dec
1	Vendor_Name	Character	30	
2	Address	Character	30	
3	City	Character	20	
4	State	Character	2	
5	Zip	Character	18	
6	Phone	Character	10	
7	FEIN_No	Character	10	
8	Customer_N	Character	15	

Figure 9. Structure of Vendor Record

Department File

This file contains the information about the department. See the structure in Figure 10.

Field	Field Name	Type	Width	Dec
1	Dep_Name	Character	12	
2	Dep_Code	Character	3	
3	Total_Due	Numeric	7	2

Figure 10. Structure of Department

Items Received File

This file contains the information about received item. The quantity and date of receipt is part of the information. See the structure of item received record in Figure 11.

Field	Field Name	Type	Width	Dec
1	Vendor_Nam	Character	30	
2	Date_of_Rec	Character	8	
3	Qty_In	Numeric	6	2
4	Item_Name	Character	30	

Figure 11. Structure of Item Received Record

Item Used File

This file contains information about items being used by departments. See the structure of item used record in Figure 12.

Field	Field Name	Type	Width	Dec
1	Ob_Code	Character	3	
2	Dep_Code	Character	3	
3	Qty_Out	Numeric	7	2
4	Dat_of_Use	Date	8	
5	Item_Name	Character	30	
6	Tot_Due	Numeric	7	2

Figure 12. Structure of Item Used Record

System Modules and Screen Descriptions

In this section details of the operation of each module which produces screens and appearance of each screen is discussed briefly. A description of each of these screens is presented below.

Since only authorized people have access to the computer, there is no password routine to get to the inventory system.

When the menu program is invoked, it first displays a menu on the screen providing the operator with several

functional alternatives from which the operator chooses one function to perform. That choice results in the execution of a program called by the menu program. At the end of the chosen function, the menu screen comes up again asking the operator for another choice of function. This process goes on until the operator chooses the option that will cause dbase to exit from the menu loop. See Figure 13 for System Calling Tree.

```
                 ┌─ pgm1a { Item_Form { New_Form
          pgm1 <   pgm1b { Vendor_Form { layout1b
                 │ pgm1c { Dep_Form { layout1c
                 └─

                 ┌─ pgm2a { layout2a
                 │ pgm2b { L1b
         pgm2 <  │ pgm2c { layout 2c
                 │ pgm2d { pgm2dres { L2d
                 │ pgm2e { pgm2econs { layout 2e
prgmain <        └─
                 ┌─ pgm3a { layout3a
         pgm3  <  pgm3b { layout3b
                 │ pgm3c { layout3e
                 └─

                 ┌─ p4a
                 │ p4b
                 │ p4c
         pgm4  <  p4d
                 │ p4e
                 │ p4f
                 │ p4g
                 │ p4h
                 └─
```

Figure 13. Calling Tree

Inventory Control

==========

(1) - Adding a new record

(2) - Updating a new record

(3) - Deleting a new record

(4) - Listing or print out

(5) - Exit out of inventory

==========

Please Enter Your Selection:

Figure 14. Main Inventory Menu

'Main menu selection "1 - Adding a New Record": Provide facilities for adding new items, adding new departments, or adding new vendors. The programs are cyclic; that is you can add as many items, departments or vendors as necessary once the program has been started. After adding one record, program will display:

Would You Like to Add Another Record? y/n

Please Enter Y or N

When additions are made to the item master file, the new record is added and placed in proper position. The file will be reindexed before other processing takes place for the new items.

Adding New Record

```
==================================
    (A) - Adding New Item
    (B) - Adding New Vendor
    (C) - Adding New Department
    (D) - Exit to Main Menu
==================================
    Please Enter Your Selection:
```

Figure 15. Adding New Record

A check to assume non-duplication of item is made before it is added. If the item already exists, the message "Duplicate Item" will be displayed. If the operator is done adding new records to the file he can always return to the main menu by choosing "D - Exit to Main Menu".

*Main menu selection "2 - Updating a New Record": provide facilities for changing item records, vendor records, and department records. Also, will record all the received or used items.

Updating a Record

```
================================
   A - Editing the item record
   B - Editing the vendor record
   C - Editing the department record
   D - Restock
   E - Consumption
   F - Exit to main menu
================================
   Please Enter Your Selection:
```

Figure 16. Updating a Record Menu

Changing an existing item, department or vendor can be accomplished by entering the name of the item, department or vendor to be changed. Then the program will display the item, department, or vendor information. The operator can make any changes to the fields and later verify it. The record will be rewritten to the item inventory file.

In case of entering the item name incorrectly, a message "No Such Item Name" will be displayed on the screen.

Selection of "D" will let operator enter all the information about received item. Also, the balance of the item inventory will change accordingly.

Selection "E" will let operator enter all the consumption by the department. It will update the balance of the item inventory accordingly. Also, it will make a record of the amount of money that each department has to pay for the consumed item.

·Main menu selection "3 - Deleting a Record": provide facilities for deleting an item or a vendor or a department. The program will ask for item name (or vendor or department name) to be deleted. Then the program will display the item name and descriptions of that item for which a deletion was requested. It will ask for confirmation. Operator can confirm selection by entering y or cancel selection by entering n.

<u>Deleting A Record</u>

```
==================================
     (A) - Deleting an Item
     (B) - Deleting a Vendor
     (C) - Deleting a Department
     (D) - Exit to Main Menu
==================================
   Please Enter Your Selection:
```

Figure 17. Deleting a Record

When an item is deleted, the operator sees the message "One Item Was Deleted From The Inventory". When an item is deleted from the file it is also physically removed from the file.

Main menu selection "4 - Listing or Printout": provide facilities for getting several reports. See Figure 18.

Listing or Printout

```
=====================================
(A) - List of all the items, restocked
(B) - List of all the Inventory Items
(C) - List of all the Items at minimum
(D) - List of all the Departments
(E) - List of all the Vendors
(F) - List of all the Consumed Items
(G) - Department Expenditure by Object Code
(H) - Total Expenditure by Object Code
(I) - Exit to Main Menu
=====================================
     Please Enter Your Selection:
```

Figure 18. Listing or Printing Menu

In all reports, the user has a choice of either printing the reports or just seeing it on the screen. Also, the user is given an opportunity to start the printer and align the paper before the printing of the report is started.

Report printing has no side effect on the condition of the files; reports in this menu can be obtained repeatedly without altering the file contents in any way.

The source listing of all programs can be found in Appendix A.

SUMMARY

The inventory system designed and implemented for Lafene Health Center at Kansas State University speeds up the whole process of inventory. The system is able to alert the user when reorder levels have been reached and provide information on the suppliers of items. By using this system, an employee is able to call an item up to the screen and get all the information about the item. This also will help the user greatly in the bookkeeping and determining each departments' expenditures by object code.

The system has been designed in such a way that it is very easy to add any new functions without modifying the structure of the data base.

REFERENCES

1. Chen, Peter, Pin-Shan. The Entity Relationship Approach to Logical Database Design. The Q.E.D. Monograph Series, Q.E.D. Information Sciences, Inc., 141 Linden Street, Wellesley, Massachusetts 02181. pp. 73.
2. Date, C. J. An Introduction to Database Systems. Addison-Wesley. 1986.
3. Hawryszkiewycz, I. T. Database Analysis and Design. Science Research Associates. 1984.
4. Kroenke, David. Database Processing. Chicago Science Research Associates, Inc. 1983.
5. Parkinson, Richard C. Data Analysis the Key to Database Design. Q.E.D. Information Sciences, Inc., 141 Linden Street, Wellesley, Massachusetts 02181. 1984.
6. Teorey, Toby J., and Fry, James P. Design of Database Structures. Prentice-Hall, Inc. Englewood Cliffs, New Jersey 07632. 1982. pp. 492.
7. Ullman, Jeffrey D. Principles of Database Systems. Rockville, Maryland. Computer Science Press. 1982.

APPENDIX A

SYSTEM SOURCE CODE

COCO.PRG

```
Clear all
Set confirm on
Set talk off
Set bell off
Set delet on
Store ' ' to errmsg
Do while .T.
Clear
?? "                    INVENTORY CONTROL "
?  "              ------------------------------"
Text
                     (1) - Adding a new record
                     (2) - Updating a new record
                     (3) - Deleting a new record
                     (4) - listing or print out
                     (5) - exit out of inventory

                  -------------------------------
Endtext
?
?
? '               '+errmsg
```

```
Wait '        Please enter your selection...'  to action
Store ' ' to errmsg
If action <'1' .OR. upper(action) >'5'
Store '   please reenter,' to errmsg
?Chr(7)
Endif
If action = '1'
Do pgm1
Endif
If action = '2'
Do pgm2
Endif
If action = '3'
Do pgm3
Endif
If action = '4'
Do pgm4
Endif

If action = '5'
Return
Endif
*If action <'1' .or. action >'5'
*Store ' reenter ' to errmsg
```

```
*?Chr(7)
*Endif
Enddo
```

PGM1.PRG

```
* Program adding a new record
Clear
Store ' '  to errmsg
Do while .T.
Clear
??"                    ADDING NEW RECORD "
? "              ---------------------------"
Text
                     (A) - Adding New Item
                     (B) - Adding New Vendor
                     (C) - Adding new department
                     (D) - Exit to Main Menu


                ----------------------------------
Endtext
?
?
?'                '+ errmsg
Wait '              Please Enter Your Selection...'  to
Action
Store ' ' to errmsg
```

```
Do case
   Case upper(action) = 'D'
      Return
   Case upper(action) = 'A'
     Do pgm1a
   Case upper(action) = 'B'
     Do pgm1b
   Case upper(action) = 'C'
     Do pgm1c
  * Case upper(action) = 'D'
   *  Return
Otherwise
     Store '  Please Reenter,'  to errmsg
? Chr(7)
Endcase
Enddo
```

```
Clear
Store '       ' to errmsg
Action = 'Y'
Do while upper(action) = 'Y'

 Do  item_form
Clear
Text
                    Would You Like to Add Another Record? y/n

                    Please Enter Y or N
Endtext
?
?
? '           '+ errmsg
Wait '         option?'  to action
Store ' ' to errmsg

If upper(action) ='N'
Text

              Hit control W to save the records.
Endtext
```

```
Wait
Close all
Reindex
Use item_inv
*Index on item_name to name_indx
Use
Return
Endif
If upper(action) = 'Y'
Store 'Y' to action
Endif
Store 'Y' to action
Enddo
```

ITEM_FOR.PRG

```
* This prg adds record to master file
  Clear
  Use item_inventory index name_indx
  Set format to newform
  Append
  Set format to
  Return
```

NEWFORM.FMT

```
@  3, 19  SAY "DATA ENTRY FOR ITEM INVENTORY"
@  6,  4  SAY "STOCK_NO"
@  6, 16  GET  ITEM_INV->ITEM_NO
@  6, 26  SAY "ITEM NAME"
@  6, 37  GET  ITEM_INV->ITEM_NAME
@  8, 26  SAY "ITEM_DESCR"
@  8, 38  GET  ITEM_INV->ITEM_DESCR
@ 10,  4  SAY "S_BALANCE"
@ 10, 16  GET  ITEM_INV->S_BALANCE
@ 11,  4  SAY "L_BALANCE"
@ 11, 16  GET  ITEM_INV->L_BALANCE
@ 14, 25  SAY "VENDOR_NAM"
@ 14, 37  GET  ITEM_INV->VENDOR_NAM
@ 15,  7  SAY "Min"
@ 15, 16  GET  ITEM_INV->FLAG
@ 16, 30  SAY "BRAND"
@ 16, 39  GET  ITEM_INV->BRAND
@ 17,  4  SAY "PRICE"
@ 17, 16  GET  ITEM_INV->PRICE
@ 18, 4   SAY "OBJECT_CODE"
@ 18, 16  GET  ITEM_INV->OB_CODE
@ 18, 30  SAY "UNIT"
@ 18, 39  GET  ITEM_INV->UNIT
```

```
@  1,  0  TO 19, 74    DOUBLE
@  4, 17  TO  4, 50
@  8,  2  TO 12, 25
```

PGM1B.PRG

```
Clear
Store '       ' to errmsg
Action = 'Y'
Do while upper(action) = 'Y'

 Do  vendor_form
Clear
Text
                    Would you like to add another record? y/n

                    Please enter y or n
Endtext
?
?
? '           '+ errmsg
Wait '         option?'  to action
Store '     ' to errmsg
If upper(action) ='N'
Text

              Hit control W to save the records.
Endtext
Wait
```

```
Close all
Reindex
Use vendor
*Index on vendorname to ven_indx
Close all
Return
Endif
If upper(action) = 'Y'
Store 'Y' to action
Endif
Store 'Y' to action
Enddo
```

LAYOUT1B.FMT

```
@  2, 22  SAY "Adding data for the vendor"
@  6,  4  SAY "VENDORNAME"
@  6, 16  GET  VENDOR->VENDORNAME
@  8,  4  SAY "ADDRESS"
@  8, 16  GET  VENDOR->ADDRESS
@ 10,  4  SAY "PHONE"
@ 10, 16  GET  VENDOR->PHONE
@ 11, 42  SAY "FEIN_NO"
@ 11, 54  GET  VENDOR->FEIN_NO
@ 13, 42  SAY "CUSTOMER_N"
@ 13, 54  GET  VENDOR->CUSTOMER_N
@  3, 19  TO  3, 51
@  0,  0  TO 16, 71    DOUBLE
```

PGM1C.PRG

```
Clear
Store '       ' to errmsg
Action = 'Y'
Do while upper(action) = 'Y'

 Do  dep_form
Clear
Text
                    Would you like to add another record? y/n

                    Please enter y or n
Endtext
?
?
? '           '+ errmsg
Wait '        option?'  to action
Store ' ' to errmsg

If upper(action) ='N'
Text

             Hit control W to save the records.
Endtext
```

```
Wait
*Close all
*Reindex
Use department Sset index to cod_indx,dep_indx
Reindex
*Index on dep_name to dep_indx
*Close all
*Use department
*Index on dep_code to cod_indx
Close all
Return
Endif
If upper(action) = 'Y'
Store 'Y' to action
Endif
Store 'Y' to action
Enddo
```

DEP_FORM.PRG

```
* This prg adds record to master file
  Clear
  Use department index dep_indx
  Set format to layoutlc
  Append
  Set format to
  Return
```

PGM2.PRG

```
* This prg will update a record
Clear
Store ' ' to errmsg
Do while .T.
Clear
??"                          Updating a Record "
? "                 ------------------------------------"
Text
                        (A) - Editing the Item Record
                        (B) - Editing the Vendor Record
                        (C) - Editing the Department Record
                        (D) - Restock
                        (E) - Consumption
                        (F) - Exit to Main Menu

                      ------------------------------------
 Endtext
?
?
? '                   '+errmsg
Wait '              Please Enter Your Selection...'  to
Action
Store ' ' to errmsg
```

```
Do case
    Case upper(action) ='F'
    Return
    Case upper(action) = 'A'
    Do pgm2a
    Case upper(action) = 'B'
    Do pgm2b
    Case upper(action) = 'C'
    Do pgm2c
    Case upper(action) = 'D'
    Do pgm2d
    Case upper(action) = 'E'
    Do pgm2e
Otherwise
   Store  '  Please Reenter ..'  to errmsg
   ? Chr(7)
Endcase
Enddo
```

PGM2A.PRG

```
*This prgm will edit randomly on key, with duplicate checks
Use item_inventory index name_indx
Do while .T.
Goto top
Clear
@ 01,01 say date()
@ 01, 18 say ' Edit Randomly on item_name, with
Duplicate_check '

Store '                                  ' to mstart
@ 10, 01 say ' Enter item_name '
@ 10, 22 get mstart PICTURE
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'

@ 12, 01 say 'Touch <cr>,to exit...'
Read
If mstart = '                                  '
Use
Return
Endif
Find &mstart
If .NOT. found()
@20, 15 say 'No such item name ! touch <cr> '
```

```
?Chr(7)
Wait ' '
Loop
Endif
Store item_name to mitem_name,real_name
Store ' ' to mwarn
Store .T. to nogood
Do while nogood
Set format to layout2a
Read
Store' ' to mwarn
If upper(mitem_name)# upper(real_name)
Find &mitem_name
If found()
  Store 'duplicate key! ' to mwarn
?Chr(7)
Find &real_name
Loop
Else
Find &real_name
Replace item_name with mitem_name
Endif
Endif
Store .F. to nogood
```

```
Enddo
Set format to
Enddo
Use
Return
*Endpgm2a
```

```
@ 03,01 say date()

@ 3, 28 say ' INVENTORY DATA EDIT SCREEN '
@ 4, 28 say '----------------------------'
@ 5, 20 say 'Random edit on item_names, with dup_checks '
@ 8,1 say 'ITEM_NAME        ' get mitem_name
@ 9,1 say 'ITEM_NO          ' get item_no picture 'xxxxxx'
@10,1 say 'ITEM_DESCRIPTION' get item_descr
@ 11,1 say 'L_BALANCE        ' get l_balance picture
                               '9999.99'
@ 12,1 say 'S_BALANCE        ' get s_balance picture
                               '9999.99'
@ 13,1 say 'FLAG             ' get flag picture '999.99'
@ 14,1 say 'VENDOR           ' get vendor_name
@ 15,1 say 'BRAND            ' get brand picture
                               'xxxxxxxxxxxxxxx'
@ 16,1 say 'PRICE            ' get price picture '999.99'
@ 17,1 say 'UNIT             ' get unit picture '999.99'
@ 18,1 SAY 'OBJECT CODE      ' GET OB_CODE PICTURE 'XXX'
@ 21, 30 say mwarn
```

PGM2B.PRG

```
*This prgm will edit randomly on key, with duplicate checks
Use vendor index vendindx
Do while .T.
Goto top
Clear
@ 01,01 say date()
@ 01, 18 say ' Edit Randomly on vendor_name, with
Duplicate_check '

Store '                                  ' to mstar
@ 10, 01 say ' Enter Vendor_name ' get mstar

@ 12, 01 say 'Touch <cr>,to exit...'
Read
If mstar = ' '
Use
Return
Endif
Find &mstar
If .NOT. found()
@20, 15 say 'No such vendor name ! touch <cr> '
?Chr(7)
Wait ' '
```

```
Loop
Endif
Store vendor_name to mven_name,re_name
Store ' ' to mwarn
Store .T. to nogood
Do while nogood
Set format to llb
Read
Store' ' to mwarn
If upper(mven_name)# upper(re_name)
Find &mven_name
If found()
  Store 'duplicate key! ' to mwarn
?Chr(7)
Find &re_name
Loop
Else
Find &re_name
Replace vendor_name with mven_name
Endif
Endif
Store .F. to nogood
Enddo
Set format to
```

```
Enddo

Use

Return

*Endpgm2a
```

```
@  4, 19  SAY "UPDATING VENDOR RECORD"
@  8,  7  SAY "VENDOR_NAME"
@  8, 19  GET  VENDOR->VENDOR_NAME
@ 10,  7  SAY "ADDRESS"
@ 10, 19  GET  VENDOR->ADDRESS
@ 11,  7  SAY "CITY"
@ 11, 19  GET  VENDOR->CITY
@ 12,  7  SAY "STATE"
@ 12, 19  GET  VENDOR->STATE
@ 12, 39  SAY "ZIP CODE"
@ 12, 50  GET  VENDOR->ZIP
@ 14,  7  SAY "PHONE"
@ 14, 19  GET  VENDOR->PHONE
@ 15, 39  SAY "FEIN_NO"
@ 15, 50  GET  VENDOR->FEIN_NO
@ 16,  7  SAY "CUSTOMER_NO."
@ 16, 21  GET  VENDOR->CUSTOMER_N
@  1,  1  TO 18, 67    DOUBLE
@  5, 14  TO  5, 46
```

PGM2C.PRG

```
*This prgm will edit randomly on key, with duplicate checks
Set exact off
Use department index dep_indx
Do while .T.
Goto top
Clear
@ 01,01 say date()
@ 01, 18 say ' Edit Randomly on dep_name, with
Duplicate_check '

Store '              ' to mstart
@ 10, 01 say ' Enter department_name ' get mstart

@ 12, 01 say 'Touch <cr>,to exit...'
Read
If mstart = ' '
Use
Return
Endif
Find &mstart
If .NOT. found()
@20, 15 say 'No such department name ! touch <cr> '
```

```
?Chr(7)
Wait ' '
Loop
Endif
Store dep_name to mdep_name,rea_name
Store ' ' to mwarn
Store .T. to nogood
Do while nogood
Set format to layout2c
Read
Store' ' to mwarn
If upper(mdep_name)# upper(rea_name)
Find &mdep_name
If found()
  Store 'duplicate key! ' to mwarn
?Chr(7)
Find &rea_name
Loop
Else
Find &rea_name
Replace dep_name with mdep_name
Endif
Endif
Store .F. to nogood
```

```
Enddo
Set format to
Enddo
Use
Return
*Endpgm2a
```

```
@  6, 26  SAY "EDITING  DEPARTMENT"
@ 10, 17  SAY "DEP_NAME"
@ 10, 29  GET  DEPARTME->DEP_NAME
@ 12, 17  SAY "DEP CODE"
@ 12, 29  GET  DEPARTME->DEP_CODE
@ 13, 37  SAY "TOTAL_DUE"
@ 13, 49  GET  DEPARTME->TOTAL_DUE
@  2,  0  TO 18, 71    DOUBLE
@  7, 22  TO  7, 50
```

PGM2D.PRG

```
*This prgm will enter all the new items
*that are received into the inventory (restock)
Clear
Store '       ' to errmsg
Action = 'Y'
Do while upper(action) = 'Y'
Do pgm2dres
Clear
*Store '       ' to errmsg
Text

                Would You Like to Try Again ?    Y/N

                        Please enter y or n

Endtext
?
?
? '           '+ errmsg
Wait '     option ? ' to action
Store '      ' to errmsg

*If upper(action) = 'N'
```

```
*Text
*                    Hit Control W to save the changes
*
*Endtext
*Wait
*Return
*Endif
*If upper(action) = 'Y'
*Action = 'Y'
*Endif
*Store 'Y' to action
*Enddo
Do case
   Case upper(action) ='N'
        Text
                     Hit Control W to Save the Changes
        Endtext
        Wait
        Return

   Case upper(action)= 'Y'
         Action = 'Y'

   Otherwise
```

```
        Store ' Invalid !! please  Reenter ,.. ' to errmsg
        Action ='Y'
 Endcase
Enddo
```

PGM2DRES.PRG

```
*This program will add new order to the item_rec file and
* will change the balance in the item_inv accordingly.
Clear
Public mqty
*Store 0 to mqty
@ 2, 33 say ' restock new order'
@ 3,33  say '-------------------'
Store '  ' to errmsg
Public mdat_of_rec
Public mvendorname
*Store ' ' to mvendorname
Public mitem_name
Public ml_balance
Store 0 to ml_balance
Select 1
Use item_rec

Append blank
Do 12d
Replace qty_in with mqty
Replace vendor_name with mvendorname
Replace dat_of_rec with mdat_of_re
Replace item_name with mitem_name
```

```
Do while mitem_name <>  ' '

Select 2
Use item_inv index name_indx
Find &mitem_name
If found()
Repl l_balance with  (l_balance + mqty)
Close all
Return
Else
Store ' No such Item in The Inventory !!  '  to errmsg
? Chr(7)
Close all
Return
Endif
Store 0 to mqty
Store ' ' to mitem_name
Store date() to mdat_of_rec
Store ' ' to mvendorname
Enddo
Close all
Return
```

L2D.PRG

```
Clear
Store '                                  ' to mitem_name
Store 0000.00   to mqty
Store date() to mdat_of_rec
Store '                                   ' to mvendorname
@ 2,20 say ' DATA ENTRY FOR ITEM RECEIVED'
@3,18 say '----------------------------------'
@ 5,3 say 'Item_name'
@ 5, 15 get mitem_name
@ 9,3 say ' QTY_IN '
@ 9,15 get mqty
@ 11,3 say 'VENDOR NAME '
@ 11,15 get mvendorname
@ 13,3 say 'DATE'
@ 13,15 get mdat_of_rec
Read
```

PGM2E.PRG

```
* This prgm will make a record of all the items consumed
*and it will let you do it as long as you need.
Clear
Store '       ' to errmsg
Action = 'Y'
Do while upper(action) = 'Y'
Do pgm2econs
Clear
Store '       ' to errmsg
Text

        WOULD YOU LIKE TO RECORD ANOTHER CONSUMPTION  ? Y/N

                        Please enter y or n

Endtext
?
?
? '           '+ errmsg
Wait '     option ? ' to action
Store '      ' to errmsg
If upper(action) = 'N'
Text
```

```
                    Hit Control W to save the changes

Endtext
Wait
Return
Endif
If upper(action) = 'Y'
Action = 'Y'
Else
Store ' Invalid !!  Please Reenter Again,..  ' to errmsg
Action = 'Y'

Endif
Enddo
```

```
* This prgm will record: the amount of item used each day,
* department that used the item, and will record the amount
* that the department has to pay for the item.
Clear
Public mqty
Public count
Store 1 to count
Store 0000.00 to mqty
Public mdat_of_use
Store date() to mdat_of_use
Public mitem_name
Store '                                    ' to mitem_name
Public mOB_CODE
Store '    ' to MOB_CODE
Public mdep_code
Store '    ' to mdep_code
Public mtot
Store 000.00 to mtot
Public mprice
Store 0 to mprice
Public munit
Store 0 to munit
Public nmunit
```

```
Store 0 to nmunit
Select 1
Use item_use
Append blank
Do 12e
Replace OB_CODE with MOB_CODE
Replace item_name with mitem_name
Replace dep_code with mdep_code
Replace qty_out with mqty
Replace dat_of_use with mdat_of_use
Replace tot_due with mtot
Do while mitem_name <> ' '
Select 2
Use item_inv  index name_indx
Find &mitem_name
If found()
 Store l_balance to mlbal
 Store s_balance to msbal
 Store unit to munit
 Store unit to nmunit
 Store price to mprice
     Do case
       Case ( msbal > mqty )
          Replace s_balance with ( msbal - mqty )
```

```
        Case ( msbal = mqty )
           Replace s_balance with ( msbal - mqty )
        Case ( msbal < mqty )
          * Clear
           *   @ 2,3 say ' iam inside case'
            * Wait
             Do while  ( msbal + nmunit ) < mqty
                  Nmunit = munit + nmunit
                  Store ( count + 1 ) to count
            *       Clear
             *      @ 2,3 say ' inside while'
              *     Wait
             Enddo
          Replace l_balance with  ( mlbal - count )
    Replace s_balance with (( msbal + nmunit ) - mqty )
   Endcase
 Store ( mqty * mprice ) to mtot

 Else
 Clear

 @ 2,10 say '  NO Such Item in Inventory!! '
 ? Chr(7)
 Endif
```

```
Select 1
Use item_used
Go bott

Repl tot_due with mtot
Select 3
Use department index cod_indx
Find &mdep_code
If found()
Repl total_due with (total_due + mtot)
Else
Clear

@ 4,10 say  'NO Such Dep_code !!  '
? Chr(7)
Endif
*Clear all
Close all
Store 0000.00 to mqty
Store '                                  ' to mitem_name
Store '    ' to mOB_CODE
Store '    ' to mdep_code
Store 0000.00 to mtot
Store 000.00 to munit
```

```
Store 000.00 to nmunit
Store 000.00 to mprice
Store 0 to count
Store date() to mdat_of_use
Enddo
Close all
Return
```

L2E.PRG

```
Clear
Store date() to mdat_of_use

@ 2, 18 say ' DATA ENTRY FOR ITEM USED '
@ 3, 16 say '----------------------------'
@ 5,3 say 'ITEM NAME '
@ 5,20 get mitem_name
@ 7,3 say 'OBJECT CODE'
@ 7,20 get mOB_CODE
@ 9, 3 say 'DEPARTMENT CODE '
@ 9, 20 get mdep_code
@ 11, 3 say 'QTY OUT '
@ 11, 20 get mqty
@13, 3 say 'DATE OF USED '
@ 13, 20 get mdat_of_use
@ 15 , 3 say ' TOTAL AMOUNT '
@ 15, 20 get mtot
Read
```

PGM3.PRG

```
* This prg will delete a record
Clear
Store' ' to errmsg
Do while .T.
Clear
??"                    Deleting a Record"
? "                -------------------------"
Text
                    (A) - Deleting an Item
                    (B) - Deleting a Vendor
                    (C) - Deleting a Department
                    (D) - Exit to Main Menu

                 ----------------------------

Endtext
?
?
?'                '+ errmsg
Wait '            Please enter your selection..' to action
Store ' ' to errmsg
If upper(action) = 'D'
Return
```

```
Endif
If upper(action) $('ABC')
Store 'pgm3'+ upper(action) to choice
Do &choice
Else
Store '  Please reenter...' to errmsg
? Chr(7)
Endif
Enddo
```

PGM3A.PRG

```
* This prgm will delete record randomly on key
Use item_inventory index  name_indx
Store ' ' to errmsg
Do while .T.
Clear
?'            '+ errmsg
Wait
Store '                                  ' to mitem_name
@ 01, 01 say date()
@ 01, 20 say ' random delete,via item_name:'
@ 10, 01 say ' Please Enter the item_name for delete' get
               mitem_name ; l

Picture 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxx'l

@ 12, 01 say ' touch <cr>, to exit ..'
Read
If mitem_name = ' '
Use

Return
Endif
Find &mitem_name
```

```
If .NOT. found()
@ 20, 17 say ' No such item_name found! touch <cr>...'
?Chr(7)
Wait' '
Loop
Endif
Store ' 'to mconfirm
Set format to layout3a
Read
If upper(mconfirm) ='Y'
*Dele
*Pack
*Else
*Store ' Invalid!!  Please Try again' to errmsg
*? Chr(7)
Clear
@ 5,5 say ' One item Was deleted from the inventory '
Dele
Pack

Endif
Set format to
Enddo
```

```
@ 01,01 say date()
@ 1,28 say ' INVENTORY DELETE SCREEN '
@ 2,28 say  '-------------------------'
@ 3,25 say ' RANDOM DELETE ON ITEM_NAME '
@ 6,20 say ' ARE YOU SURE YOU WANT TO DELETE THIS RECORD
                (y/n);
?' get mconfirm
@ 7,20 say '-------------------------------------------'
@ 7,20 say '   '
@ 8,01 say 'stock_no:'
@ 8,20 say item_no pict 'xxxxxx'
@ 10,1 say 'item_name: '
@ 10, 20 say item_name
@ 11,1 say 'item description:'
@ 11,20 say item_descr
@ 12,1 say 'l_balance: '
@ 12,20 say l_balance
@ 13,1 say 's_balance:'
@ 13,20 say s_balance
@ 14,1 say 'min (flag): '
@ 14,20 say  flag
@ 15,1 say 'vendor: '
@ 15,20 say vendor_name
```

```
@ 16,1 say 'brand:'
@ 16,20 say brand
@ 17,1 say 'price: '
@ 17,20 say price
@ 18, 1 say 'unit: '
@ 18,20 say unit
@ 19, 1 say 'object code:'
@ 19,20 say ob_code
```

PGM3B.PRG

```
* This prgm will delete record randomly on key
Use vendor index  vendindx
Store ' ' to errmsg
Do while .T.
Clear
? '          '+errmsg
Wait
*Clear
Store '                                  ' to mvendorname
@ 01, 01 say date()
@ 01, 20 say ' random delete,via vendor_name:'
@ 10, 01 say ' Please Enter the vendor_name for delete'
               get mvendorname
@ 12, 01 say ' touch <cr>, to exit ..'
Read
If mvendorname = ' '
Use
Return
Endif
Find &mvendorname
If .NOT. found()
@ 20, 17 say ' No such vendor_name found! touch <cr>...'
?Chr(7)
```

```
Wait' '
Loop
Endif
Store ' 'to mconfirm
Set format to layout3b
Read
If upper(mconfirm) ='Y'
Clear
@ 9,9 say ' one vendor deleted '
Wait
Dele
Pack
Endif
Set format to
Enddo
```

```
 * this is the format for vendor delete
@ 1, 1 say date()
@ 1, 28 say ' Vendor Delete Screen '
@ 2, 26 say '---------------------------'
@ 3, 25 say  ' CONFIRM THE DELETE !   (Y/N)' get mconfirm
@ 8,2 say 'vendor name '
@ 8,20 say vendor_name
@ 9,2 say 'Address   '
@ 9,20  say address
@ 10, 2 say 'Phone'
@ 10, 20 say phone
@ 11,2 say 'FEIN NO.'
@ 11,20 say  fein_no
@ 12,2 say ' Customer NO.'
@ 12, 20 say customer_no
```

PGM3C.PRG

```
* This prgm will delete record randomly on key
Use department index  dep_indx
Store ' ' to errmsg
Do while .T.
Clear
? '       '+errmsg
Wait
Store '              ' to mdepname
@ 01, 01 say date()
@ 01, 20 say ' Random Delete,Via Department_name:'
@ 10, 01 say ' Please Enter the Department_name for
Delete';
 get mdepname  picture 'xxxxxxxxxxxx'
@ 12, 01 say ' touch <cr>, to exit ..'
Read
If mdepname = ' '
Use
Return
Endif
Find &mdepname
If .NOT. found()
@ 20, 17 say ' No such department_name found! touch
<cr>...'
```

```
?Chr(7)
Wait' '
Loop
Endif
Store ' 'to mconfirm
Set format to layout3c
Read
If upper(mconfirm) ='Y'
Clear
@ 7,7 say  ' one dpartment is deleting '
Dele
Pack
Endif
Set format to
Enddo
```

```
 * this prgm will be the format of the department to delete
@ 1, 1 say date()
@ 1, 28 say ' DEPARTMENT DELETE SCREEN '
@ 2, 28 say '---------------------------'
@ 3, 25 say ' Random Delete on Department Name  '
@ 6, 20 say ' CONFIRM THE DELETE   !  (Y/N)  ' get mconfirm
@ 8, 3 say ' DEPARTMENT NAME '
@ 8, 20 say dep_name
@ 9, 3 say ' DEPARTMENT CODE '
@ 9, 20 say dep_code
@ 10, 3 say 'TOTAL DUE'
@ 10, 20 say total_due
```

PGM4.PRG

```
* This prg will produce all the listing or printing
Clear
Store' ' to errmsg
Do while .T.
Clear
??"                    Listing and Printing"
? "         ---------------------------------------------"
text
                (A) - List of all the Items Re_stocked
                (B) - List of all the Inventory Items
                (C) - List of all Items at Minimum
                (D) - List of all the departments
                      Department Expenditurres
                (E) - List of all the vendors
                (F) - List of all the consumed Items
                (G) - Department Expenditure By Object Code
                (H) - Total Expenditure by Object Code
                (I) - Exit to Main Menu
           ---------------------------------------------
Endtext
?
?
?'                        ' + errmsg
```

```
Wait '            Please enter your selection..'  to action
Store ' ' to errmsg
If upper(action) = 'I'
Return
Endif
If upper(action) $('ABCDEFGH')
Store 'p4' + upper(action) to choice
Do &choice
Else
Store '  please reenter.. '  to errmsg
? Chr(7)

Endif
Enddo
```

```
Clear
Store ' to print ' to p
Store ' ' to ans
Do whil .NOT. ans $ 'yYnN'
Clear
@ 2,2 say ' IS YOUR PRINTER READY   (Y/N) ' get ans
@ 4, 2 say ' type  E to exit '
Read
If upper(ans) = 'N'
Ans = ' '

Wait
Loop
Endif
If upper(ans) = 'E'
Ans = ' '
Return
Endif
If upper(ans) = 'Y'
Store ' ' to ans
Clear
Mdate = ctod('  /  /  ')
@ 2,2 say 'PLEASE ENTER DATE' GET MDATE
```

```
Read
Mldate = dtoc(mdate)
Store ' set filter to (dat_of_rec) = ' to al
Store "ctod('" to bl
Store "')" to b2
C = bl + mldate + b2
Use item_rec
&al &c
Clear
Store ' ' to ch
@ 4,3 say ' IF YOU WANT TO GET PRINT OUT PLEASE ENTER  Y';
Get ch
Read
If upper(ch) = 'Y'

Set printer on
Clear
@ 2,18 say 'LIST OF ITEM RE-STOCKED '
@ 3,15 SAY '-----------------------------'
List all trim(item_name), qty_in to print
Wait
Use
Set printer off
*Return
```

```
Else
Clear
@ 2,17 say 'LIST OF ITEM RE-STOCKED '
@ 4,15 SAY ' ---------------------------'
DISP ALL TRIM(ITEM_NAME), QTY_IN
Wait

Use

*Return
Endif
Else
? Chr(7)
Loop
Endif
Enddo
Return
```

P4B.PRG

```
Clear
Store ' ' to ans
Do whil .NOT. ans $ 'yYnN'
Clear
@ 2,2 say ' IS YOUR PRINTER READY   (Y/N) ' get ans
@ 4, 2 say ' type  E to exit '
Read
If upper(ans) = 'N'
Ans = ' '
Wait
Loop
Endif
If upper(ans) = 'E'
Ans = ' '
Return
Endif
If upper(ans) = 'Y'
Store ' ' to ans
Clear
Store ' ' to ch
@ 2, 5 say ' IF YOU WANT TO GET PRINT OUT PLEASE ENTER  Y
';
Get ch
```

```
Read
If upper(ch) = 'Y'
Set printer on

@ 2,20 say 'INVENTORY LIST '
@ 3,18 SAY '--------------------'
USE ITEM_INV INDEX NAME_INDX
List all trim(item_name) ,s_balance, l_balance , unit ;
To printer

Wait
Use
Set printer off
*Return
Else
Use item_inv index name_indx
Clear
@ 2,3 say 'INVENTORY LIST'
@ 3, 2 SAY '------------------'
Disp all trim(item_name), s_balance, l_balance, unit
Wait
Use
*Return
Endif
```

```
Else
Chr(7)
Loop
Endif
Enddo
Return

@ 1,2 say date()
@ 1,11 say ' List of all the Item in The Inventory '
Clear
Use item_inv index name_indx
Display all  s_balance, l_balance, item_no, item_name, flag
Wait
Use
Return
```

P4C.PRG

```
Clear
SELECT 1
USE ITEM_INV INDEX VITEM ALIAS VENDITEM
SELECT 2
USE VENDOR INDEX VENDINDX ALIAS VEND
SELECT 1
SET RELATION TO VENDOR_NAM INTO VEND
Store ' ' to ans
Do whil .NOT. ans $ 'yYnN'
Clear
@ 2,2 say ' IS YOUR PRINTER READY   (Y/N) ' get ans
@ 3,2 SAY ' PLEASE USE LARGE SIZE PAPER'
@ 4, 2 say ' type  E to exit '
Read
If upper(ans) = 'N'

Ans = ' '
REPORT FORM RPT4C
Wait
CLOSE DATABASES
RETURN
Endif
```

P4D.PRG

```
Clear
Store ' ' to ans
Do whil .NOT. ans $ 'yYnN'
Clear
@ 2,2 say ' IS YOUR PRINTER READY   (Y/N) ' get ans
@ 4, 2 say ' type  E to exit '
Read
If upper(ans) = 'N'
Ans = ' '
Wait
Loop
Endif
If upper(ans) = 'E'
Ans = ' '
Return
Endif
If upper(ans) = 'Y'
Store ' ' to ans
Clear
STORE ' ' TO CH
@ 2,2 say ' IF YOU WANT TO GET A PRINT OUT PLEASE ENTER Y';
GET CH
READ
```

```
IF UPPER(ch) = 'Y'

Set printer on
Clear
@ 2,20 say 'DEPARTMENT EXPENDITURES '
@ 3,18 SAY '----------------------------'
USE department INDEX dep_INDX

List all trim(dep_name), dep_code, total_due to print
Wait
Set printer off
Use
Return
Else
Clear
@ 2,2 say ' DEPARTMENT EXPENDITURES '
@ 3,2 SAY '----------------------------'
Use department index dep_indx
Disp all trim(dep_name), dep_code, total_due
Wait
Use
Return
Endif
Else
```

```
Chr(7)
Loop
Endif
Enddo
Return

If upper(ans) = 'E'
Ans = ' '
CLOSE DATABASES
Return
Endif
If upper(ans) = 'Y'
Store ' ' to ans
REPORT FORM RPT4C  TO PRINTER  FOR  L_BALANCE < FLAG + 1
Wait
CLOSE DATABASES
RETURN
Else
?Chr(7)
Loop
Endif
Enddo
CLOSE DATABASES
Return
```

```
Clear
Store ' ' to ans
Do whil .NOT. ans $ 'yYnN'
Clear
@ 2,2 say ' IS YOUR PRINTER READY   (Y/N) ' get ans
@ 3,2 SAY ' PLEASE USE LARGE SIZE PAPER'
@ 4, 2 say ' type  E to exit '
Read
If upper(ans) = 'N'
Ans = ' '
Wait
Loop
Endif
If upper(ans) = 'E'
Ans = ' '
Return
Endif
If upper(ans) = 'Y'
Store ' ' to ans
Clear
Store ' ' to ch
@ 2, 5 say ' IF YOU WANT TO GET PRINT OUT PLEASE ENTER  Y
';
```

```
Get ch
Read
If upper(ch) = 'Y'
*Set printer on

*@ 2,20 say 'VENDOR LIST '
*@ 3,18 SAY '--------------------'
USE VENDOR INDEX  VENDINDX
*List all trim(vendor_name)-'   '-trim(address)-'
'-trim(city);
*-'   '- trim(state)- '   '- trim(zip) -'   '- trim(phone);
*-'   '- trim(fein_no)-'   '- trim(customer_n)  to printer
Report form ved_rep  to printer
Use
*Set printer off
*Wait
Else
Use vendor index vendindx
Clear
@ 2,3 say 'VENDOR LIST'
@ 3, 2 SAY '------------------'
Disp all trim(VENDOR_name)+ ' ' + PHONE, CUSTOMER_NO
WAIT
```

```
Use
Endif
Else
Chr(7)
Loop
Endif
Enddo
Return

@ 1,2 say date()
@ 1,11 say ' List of all the Item in The Inventory '
Clear
Use item_inv index name_indx
Display all  s_balance, l_balance, item_no, item_name, flag
Wait
Use
Return
```

```
Clear
Store ' ' to ans
Do whil .NOT. ans $ 'yYnN'
Clear
@ 2,2 say ' IS YOUR PRINTER READY   (Y/N) ' get ans
@ 4, 2 say ' type  E to exit '
Read
If upper(ans) = 'N'
Ans = ' '
Wait
Loop
Endif
If upper(ans) = 'E'
Ans = ' '
Return
Endif
If upper(ans) = 'Y'
Store ' ' to ans
Clear
Mdate = ctod('  /  /  ')
@ 2,2 say 'PLEASE ENTER DATE' GET MDATE
Read
Mldate = dtoc(mdate)
```

```
Store ' set filter to (dat_of_use) = ' to a1
Store "ctod('" to b1
Store "')" to b2
C = b1 + m1date + b2
Use item_use
&a1 &c
*Set printer on
Clear
@ 2,18 say 'LIST OF ITEM CONSUMED '
@ 3,15 SAY '-----------------------------'
Disp all trim(item_name), qty_out
Wait
Use
*Set printer off
Return
Else
Chr(7)
Loop
Endif
Enddo
Return

@ 1,2 say date()
@ 1,11 say ' List of all the Item in The Inventory '
```

```
Clear
Use item_inv index name_indx
Display all  s_balance, l_balance, item_no, item_name, flag
Sait
Use
Return
```

P4G.PRG

```
Clear
N = '1'
STORE 'Y' TO ANS
DO WHILE UPPER(ANS) = 'Y'
STORE '    ' TO M1CODE
MD1 = CTOD('  /  /  ')
MD2 = CTOD('  /  /  ')
@ 2,2 SAY ' Please Enter Starting Date  ' GET MD1
@ 4,2 say ' Please Enter Ending date    ' get md2
@ 6,2 SAY ' Please Enter Department Code' get m1code
Read
D1 = DTOC(MD1)
D2 = DTOC(MD2)
STORE "'" TO B
STORE "DAT_OF_USE > CTOD(&B&D1&B)" TO COND1
STORE " DAT_OF_USE < CTOD(&B&D2&B)" TO COND2
Store " dep_code = &b&M1CODE&B" TO COND3
STORE 0 TO FINTOT
SET DEVICE TO PRINTER
@ 1,2 SAY 'Date'
@ 1, 8 say date()
@ 3, 11 SAY " Total Expenditure of Department"
@ 3, 45 SAY M1CODE
```

```
@ 3, 50 SAY "by Object Code"
@ 4, 14 SAY 'From'
@ 4, 20 SAY  D1
@ 4, 32 SAY 'To'
@ 4, 36 say d2
@ 5, 9 say "----------------------------------------"
@ 6, 11 say " Object Code "
@ 6, 30 say " Total "
@ 7, 11 say '    '
Set device to screen
USE ITEM_USE
DO WHILE VAL(N) < 6
DO CASE
   CASE N = '1'
   OBJ = '369'
   CASE N = '2'
   OBJ = '392'
   CASE N = '3'
   OBJ = '221'
   CASE N = '4'
   OBJ = '371'
   CASE N = '5'
   OBJ = '361'
ENDCASE
```

```
STORE "OB_CODE = &B&OBJ&B " TO COND
STORE 0 TO TOTAL
SET FILTER TO &COND .AND. &COND1 .AND. &COND2 .AND. &COND3
DO WHILE .NOT. EOF()
STORE TOTAL  + TOT_DUE  TO TOTAL
SKIP
ENDDO
SET FILTER TO
SET DEVICE TO PRINTER
@ 5 + ( VAL(N) * 2 ) , 11 say OBJ
@ 5 + ( VAL(N) * 2 ), 30 SAY TOTAL
@ 6 + (VAL(N) * 2 ), 11 SAY '      '
Set device to screen
STORE VAL(N) + 1 TO VN
STORE STR(VN,4) TO TN
STORE RIGHT(TN,1) TO N
STORE  FINTOT + TOTAL TO FINTOT
ENDDO
SET DEVICE TO PRINTER
@ 17,30 SAY '-----------'
@ 18, 11 SAY 'Total '
@ 18 , 30 say fintot
@ 19, 11 SAY '         '
SET DEVICE TO SCREEN
```

```
STORE ' ' TO ANS
CLEAR
@ 2,2 SAY 'Would You Like to Get Another Department
Expenditure';
Get ans
Read
Enddo
CLOSE DATABASES
RETURN
```

P4H.PRG

```
Clear
N = '1'
MD1 = CTOD('  /  /  ')
MD2 = CTOD('  /  /  ')
@ 2,2 SAY ' Please Enter Starting Date ' GET MD1
@ 4,2 say ' Please Enter Ending date   ' get md2

Read
D1 = DTOC(MD1)
D2 = DTOC(MD2)
STORE "'" TO B
STORE "DAT_OF_USE > CTOD(&B&D1&B)" TO COND1
STORE " DAT_OF_USE < CTOD(&B&D2&B)" TO COND2
STORE 0 TO FINTOT
SET DEVICE TO PRINTER
@ 1,2 SAY 'Date'
@ 1, 8 say date()

@ 3, 11 SAY " Total Expenditure of Each Object Code"
@ 4, 14 SAY 'From'
@ 4, 20 SAY  D1
@ 4, 32 SAY 'To'
@ 4, 36 say d2
```

```
@ 5, 9 say "----------------------------------------"
@ 6, 11 say " Object Code "
@ 6, 30 say " Total "
@ 7, 11 say '    '
Set device to screen
USE ITEM_USE
DO WHILE VAL(N) < 6
DO CASE
  CASE N = '1'
  OBJ = '369'
  CASE N = '2'
  OBJ = '392'
  CASE N = '3'
  OBJ = '221'
  CASE N = '4'
  OBJ = '371'
  CASE N = '5'
  OBJ = '361'
 ENDCASE
STORE "OB_CODE = &B&OBJ&B " TO COND
STORE 0 TO TOTAL
SET FILTER TO &COND .AND. &COND1 .AND. &COND2
DO WHILE .NOT. EOF()
STORE TOTAL  + TOT_DUE  TO TOTAL
```

```
SKIP
ENDDO
SET FILTER TO
SET DEVICE TO PRINTER
@ 5 + ( VAL(N) * 2 ) , 11 say OBJ
@ 5 + ( VAL(N) * 2 ), 30 SAY TOTAL
@ 6 + (VAL(N) * 2 ), 11 SAY '      '

Set device to screen
STORE VAL(N) + 1 TO VN
STORE STR(VN,4) TO TN
STORE RIGHT(TN,1) TO N
STORE  FINTOT + TOTAL TO FINTOT
ENDDO
SET DEVICE TO PRINTER
@ 17,30 SAY '-----------'
@ 18, 11 SAY 'Total '
@ 18 , 30 say fintot
@ 19, 11 SAY '         '
SET DEVICE TO SCREEN
CLOSE DATABASES
RETURN
```

AN IMPLEMENTATION OF INVENTORY SYSTEM FOR LAFENE HEALTH CENTER BASED ON DBASE III PLUS

by

SOUDABEH FARZBOD

B.S. Tehran College of Business. 1971

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1987

ABSTRACT

This report is about design and implementation of a database system for Lafene Health Center's inventory system. The report gives the various stages that were involved in the design of this inventory system.

First, functional specifications were specified. Next the data dictionary was developed and, subsequently, the functional dependencies were obtained. The implementation of Bern 2 resulted in a 3NF schema, which also helped to get a grasp of entities. The analysis of Bern 2 output lead to the E-R diagram which was later transformed into the relational model.

dbase III plus was used to implement the system.