ARAB REPUBLIC OF EGYPT

THE INSTITUTE OF NATIONAL PLANNING

Memo No. (1501)

An Information System for Personnel Management System { using Advanced Techniques of dBASE III PLUS }

by

Dr. Abdalla El-Daoushy
Dr. M. Yehia Youssef

May, 1989

CAIRO
SALAH SALEM St-NASR CTTY

Foreword

This memo. emphasize on the practical side of the dataBase and information systems. It helps in getting the job done quickly and efficiently.

The memo. presents working dataBase and information systems that do not only perform useful information tasks, but also demonstrate advanced computer programming techniques that can be used in many dataBase and information applications.

The memo. is not intended for the computer novice, However, familiarity with the basic commands of dBase III PLUS will be sufficient background.

This memo. is divided into two main parts of three chapters. Part one which comprises chapter 1 discusses the programming considerations and emphasize on advanced techniques for maximizing the speed and performance of designing and building software systems.

Part two comprises chapters 2 and 3 and presents a custom software system for managing a Personnel Management System. The techniques presented in designing this software system can be used to manage any single dataBase system.

Part two also presents the basics of creating user-friendly, menu-driven systems, creating and using custom screens and reports, and using index files for maximum speed.

Contents

Chapter 1

ADVANCED TECHNIQUES OF dBASE III PLUS

1.1 DATABASE DESIGNS :

Deciding what to store in a dataBase is one of the first steps in designing any software system. Many software systems involve only a single dataBase and perhaps one or two index files. More sophisticated software systems may use several dataBase files interactively.

In this section, four of the most commonly used dataBase designs are discussed: single dataBase, single dataBase with Memo field, relational dataBase, and master-file/transaction-file dataBase systems.

1. Single dataBase File System :

In such case, notice that:

- Whenever you add new RECORDs, or make changes to dataBase through:

  EDIT DELETE PACK BROWSE

  make sure Index files are active.

- Consider the command:

  USE Mail INDEX Names , Zips

  { where Names indexed on L_NAME and F_NAME and Zips indexed on ZIP_CODE}

  since Names is first-listed index file, the commands:

  DISPLAY LIST REPORT FORM

  will display the RECORDs in alphabetical order by Last and First Names.

  To display the RECORDs in ZIP_CODE order, just make Zips the first-listed index file as:

  USE Mail INDEX Zips , Names

2. Single dataBase File with Memo Field :

When you use the LIST command to view the RECORDs of a dataBase file including Memo field, the word Memo is displayed in the listing instead of the contents of the field itself.

However, if you specify the field names by typing ,for example,

```
LIST OFF AUTHOR , TITLE , PUBLISHER , PUB_YEAR , PAGES , ABSTRACT
                                    { ABSTRACT is a field of type Memo }
```

the contents of the ABSTRACT field are included in the listing.

You can gain some control over the appearance of the Memo field using the SET MEMOWIDTH command. For example, to specify the width of the Memo field by 40 characters (the default width is 50), enter the following command:

```
SET MEMOWIDTH TO 40
```

Another way to gain control over the appearance of the Memo field is to use a program. For example, the following program displays data including the Memo field ADDRESS from the Library MASTER.DBF file:

```
* Program..: Library.PRG
* Date.....: May, 1989
* Remark...: To print dataBase with Memo field.
*
USE MASTER
GO TOP
DO WHILE .NOT. EOF()
   ?
   ? "          Author:" , AUTHOR
   ?
   ? "           Title:" , TITLE
   ?
   ? "       Publisher:" , PUBLISHER
   ?
   ? "Publishing Date:" , PUB_YEAR
   ?
   ? "    No. of Pages:" , PAGES
   ?
   ? "Key Words:" , TOPICS
```

```
?
? "-------------------------------------------------------------"
? ABSTRACT
?
?
?
*
SKIP    && very important to avoid looping
*
ENDDO
* eof()
```

3. Relational dataBases :

Suppose there are two Accounts Receivable dataBases: AR1.DBF and AR2.DBF with the following structure:

AR1.DBF Structure:

Field #	Field name	Type	Width
1	BILL_DATE	Date	8
2	AMOUNT	Numeric	9.2
3	VEND_CODE	Character	7

AR2.DBF Structure:

Field #	Field name	Type	Width
1	VEND_CODE	Character	7
2	VENDOR	Character	25
3	ADDRESS	Character	25
4	CITY	Character	25
5	GOVERNRATE	Character	25
6	ZIP	Character	10

To get the relationship between the two dataBases, the LOOKUP (i.e. the 2nd dataBase file) dataBase file must be INDEXed on the key field (VEND_CODE) that relates the two dataBases as follows:

```
USE AR2
INDEX ON VEND_CODE TO Vendor
```

Both dataBases must be opened simultaneously with the SELECT command and the RELATIONship command as follows:

```
SELECT 1
USE AR1
SELECT 2
USE AR2 INDEX Vendor
SELECT 1
SET RELATION TO VEND_CODE INTO AR2
```

If you were to use the LIST command now, you would still see the RECORDs from the AR1.DBF.

But,to see the associated Vendor names and addresses, use the LIST command again, but this time specify field names from the AR2.DBF, using the B -> specification. (Note that AR2 in the example above is the B dataBase because it was SELECTed 2nd. A 3rd file would be the C -> dataBase, a 4th would be the D -> dataBase, and so forth).

```
LIST BILL_DATE , AMOUNT , B -> VENDOR , B -> ADDRESS
```

Then, the information is readily available from both dataBases simultaneously.

4. Master-File / Transaction-File DataBase System :

The 4th commonly used dataBase design involves Master Files and Transaction Files. These are most often used ,for example, in Inventory and Bookkeeping applications; where one dataBase file contains (up to the minute information about) the stock on hand or account balances, and additional dataBases contain RECORDs of individual transactions, such as goods sold and received.. etc.

A simple Inventory System might ,for example, use a Master dataBase file (MASTER.DBF) and two transaction dataBase files (SALES.DBF and NEWSTOCK.DBF).

The MASTER.DBF contains current information about the stock on hand. It gets its information on individual sales transactions and newstock transactions from transaction files SALES.DBF and NEWSTOCK.DBF. In a sense, MASTER.DBF is a summary of the SALES.DBF and NEWSTOCK.DBF.

The benefit of this dataBase structure is that it allows to keep abreast (i.e., to be informed of the latest information) of the exact stock on hand at any given moment, while at the same time it provides a separate, permanent RECORD of every item sold and received.

The SALES.DBF contains information about individual sales transactions. The NEWSTOCK contains information about coming goods.

The structure of both MASTER.DBF and SALES.DBF are shown as follows:

MASTER.DBF Structure:

Field #	Field name	Type	Width
1	PART_NO	Character	7
2	TITLE	Character	20
3	ON_HAND	Numeric	4
4	COST	Numeric	8.2
5	REORDER	Numeric	4

SALES.DBF Structure:

Field #	Field name	Type	Width
1	PART_NO	Character	7
2	INVOICE_NO	Character	7
3	SALSPERSON	Character	25
4	CUSTOMER	Character	25
5	QTY	Numeric	4
6	PRICE	Numeric	S.2
7	DATE	Date	S

Notice that both the SALES.DBF and MASTER.DBF have the same field called PART_NO. This common field allows dBASE to relate RECORDs in both files.

dBASE uses the UPDATE command to keep the MASTER.DBF current as follows:

i, Index each dataBase file on PART_NO field:

```
USE MASTER
INDEX ON PART_NO TO MASTER
USE SALES
INDEX ON PART_NO TO SALES
```

ii, Open both dataBases with the SELECT command and use the UPDATE command to transfer information from the SALES.DBF to the MASTER.DBF as follows:

```
CLEAR ALL
SELECT 2
USE SALES INDEX SALES
SELECT 1
USE MASTER INDEX MASTER
UPDATE ON PART_NO FROM SALES REPLACE ON_HAND WITH ON_HAND-B-> QTY
```

Conclusion:

As you see, there are many ways to design dataBases and dataBase systems. Which design you use depends on what you need to do and the limits imposed by your hardware.

1.2 Power of Indexed Files :

Using indexed files can actually save hours from a program's processing time just by setting the appropriate index files to handle the job properly.

Displaying RECORDs in Sorted Order :

Four methods are compared to each other:

1. The SORT Method:

The 1st method (the slowest if you have more than 50 RECORDs) is simply to use the SORT command to create a new physically SORTed file. For example, suppose you had already SORTed a file called PERSONEL.DBF and then added some new RECORDs with the commands:

```
USE PERSONEL
APPEND
```

You can get the RECORDs back into alphabetical order using SORT command as follows:

```
USE PERSONEL
SORT ON F_NAME TO Temp
CLOSE DATABASES
ERASE PERSONEL.DBF
RENAME Temp.DBF TO PERSONEL.DBF
USE PERSONEL
LIST
```

To sort 1000 RECORDs back into the first name order will take an estimated time of 6 minutes on a Floppy-Disk System and about 4.5 minutes on a Hard-Disk System.

2. The APPEND and INDEX Method:

```
USE PERSONEL
APPEND
INDEX ON F_NAME TO NAMES
LIST
```

This method requires about 45 seconds to re-SORT the 1000 RECORDs back into first name order on a Floppy-Disk System and about 23 seconds on a Hard-Disk System.

3. The LOCATE and INSERT Method:

This method would be to LOCATE the position in the SORTed dataBase file where each new RECORD belongs and INSERT the new RECORD directly into the proper alphabetical location.

```
USE PERSONEL
LOCATE FOR F_NAME = "Abdalla"
INSERT
LIST
```

The time consumed would be long since it has to find the appropriate place to INSERT each new RECORD plus about 50 seconds for each separate INSERT command to INSERT the RECORD on a Floppy-Disk System or about 30 seconds on a Hard-Disk System.

4. Creating an Index File and APPEND Command:

```
USE PERSONEL INDEX NAMES
```

Any changes to the PERSONEL.DBF, whether they are made through:

```
APPEND, EDIT, BROWSE, DELETE, PACK, or READ
```

are made to the index file (NAMES) as well, and the index file is automatically resorted and adjusted. For example,

```
USE PERSONEL INDEX NAMES
APPEND
LIST
```

dBASE automatically puts the index back into the first name order in less than one second and the LIST command immediately displays the RECORDs in last name order.

The following table shows the restoring times for the 4 techniques just described:

Method	Commands used	time required (in seconds) Hard-Disk System	Floppy-Disk System
1	USE, APPEND, and SORT	270	360
2	USE, APPEND, and INDEX ON	23	45
3	USE, LOCATE, and INSERT	30	50
4	USE, INDEX file and APPEND	1	1

Clearly, the 4th method using an active index file while adding the new RECORDs is the fastest.

Faster Searching :

Consider the following two different programs for two different approaches and compare their processing time:

The 1st Approach:

This approach uses the standard LIST FOR approach to display all RECORDs with first name Abdalla.

```
*
* LIST FOR Approach
*
CLEAR
USE PERSONEL INDEX NAMES
ACCEPT "List all people with what first name? " TO Search
*
LIST FOR F_NAME = Search
* eof()
```

The time required to display 10 Abdallas is about 2 minutes on a Floppy-Disk System and 32 seconds on a Hard-Disk System.

The 2nd Approach:

This approach uses the FIND or SEEK commands to look up the first Abdalla in the NAMES index, then use the WHILE option to display the remaining Abdallas in the dataBase file.

```
*
* FIND, SEEK, and LIST WHILE Approach
*
CLEAR
USE PERSONEL INDEX NAMES
ACCEPT "List all people with what last name? " TO Search
SEEK Search
LIST WHILE F_NAME = Search
* eof()
```

The processing time here compared with the program above is 5 seconds on a Floppy-Disk System and less than 4 seconds with a Hard-Disk System.

The following table compares the processing time for the two different approaches:

		time required (in seconds)	
Method	Commands used	Hard-Disk System	Floppy-Disk System
1	LIST FOR	32	120
2	SEEK and LIST WHILE	3.79	5.57