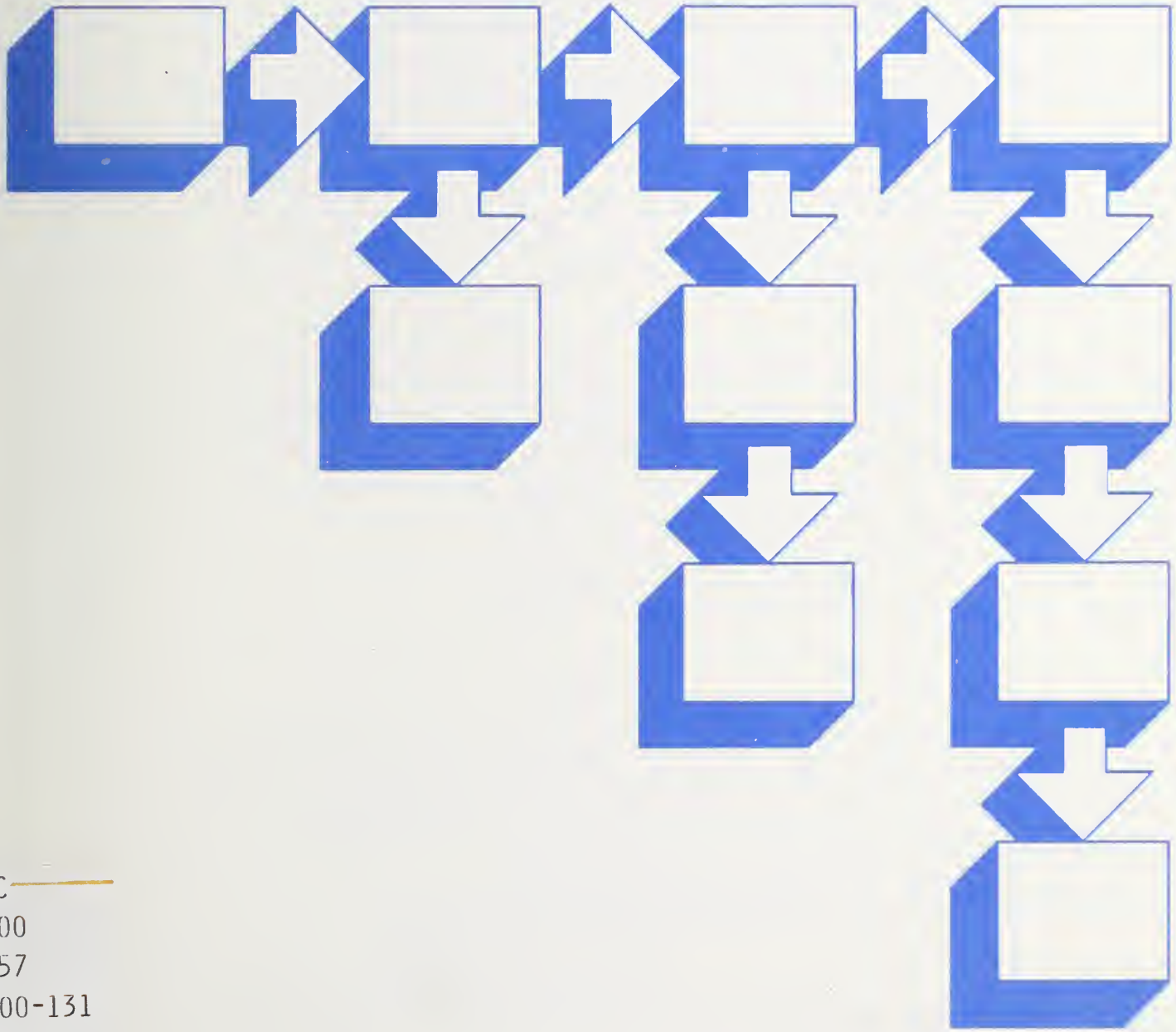
Computer Science and Technology

NBS Special Publication 500-131

Guide for Selecting Microcomputer Data Management Software

Charles L. Sheppard



The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, the Institute for Computer Sciences and Technology, and the Institute for Materials Science and Engineering.

The National Measurement Laboratory

Provides the national system of physical and chemical measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; provides advisory and research services to other Government agencies; conducts physical and chemical research; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

- Basic Standards[2]
- Radiation Research
- Chemical Physics
- Analytical Chemistry

The National Engineering Laboratory

Provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

- Applied Mathematics
- Electronics and Electrical Engineering[2]
- Manufacturing Engineering
- Building Technology
- Fire Research
- Chemical Engineering[2]

The Institute for Computer Sciences and Technology

Conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

- Programming Science and Technology
- Computer Systems Engineering

The Institute for Materials Science and Engineering

Conducts research and provides measurements, data, standards, reference materials, quantitative understanding and other technical information fundamental to the processing, structure, properties and performance of materials; addresses the scientific basis for new advanced materials technologies; plans research around cross-country scientific themes such as nondestructive evaluation and phase diagram development; oversees Bureau-wide technical programs in nuclear reactor radiation research and nondestructive evaluation; and broadly disseminates generic technical information resulting from its programs. The Institute consists of the following Divisions:

- Inorganic Materials
- Fracture and Deformation[3]
- Polymers
- Metallurgy
- Reactor Radiation

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Gaithersburg, MD 20899.
[2]Some divisions within the center are located at Boulder, CO 80303.
[3]Located at Boulder, CO, with some elements at Gaithersburg, MD.

Computer Science and Technology

NBS Special Publication 500-131

Guide for Selecting Microcomputer Data Management Software

Charles L. Sheppard

Center for Programming Science and Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, MD 20899

Issued October 1985

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

Library of Congress Catalog Card Number: 85-600598
National Bureau of Standards Special Publication 500-131
Natl. Bur. Stand. (U.S.), Spec. Publ. 500-131, 65 pages (Oct. 1985)
CODEN: XNBSAV

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1985

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

TABLE OF CONTENTS

LIST OF FIGURES

GUIDE FOR
SELECTING MICROCOMPUTER DATA MANAGEMENT SOFTWARE

Charles L. Sheppard

ABSTRACT

This guide provides information to assist data processing managers in the selection process for microcomputer data management software. General information is provided on the different categories into which microcomputer data management software can be grouped. The features that distinguish the software packages along this spectrum are discussed and illustrated. Inspection of this spectrum shows that there is a common set of features among packages regardless of their classification. This set of features and application requirements are used to develop selection issues. These issues serve as the foundation upon which an evaluation methodolgy is developed.

Key words: application; assessment; benchmark; database; evaluation; microcomputers; multi-file; selection; single-file.

Acknowledgements - I would like to acknowledge the members of our student program team, who worked diligently and competently on the analysis and testing of the microcomputer software assigned to them. I would like to extend special thanks to Dawn Hill who pioneered this effort with me and to Sam Cook, Tammy Kirkendall, Francis Stanbach, and Bill Youstra who joined in later.

1. INTRODUCTION

In recent years, there has been a steady growth in data management software developed for microcomputers. Many of these packages claim to be database management systems (DBMS). The validity of this claim depends upon how one defines the capabilities of a DBMS. The definitions that follow illustrate the diversification associated with defining a DBMS:

> "A DBMS is a set of procedures and data structures that isolates the applications from the details of the creation, retrieval, storage, modification, security, and physical storage structure of computerized data bases. It presents an application with a view, as required by its processing needs, without consideration for the physical storage or access of the data". [TSIC77]
>
> "The database management system (DBMS) is the software that handles all access to the database. Conceptually what happens is the following: (1) A user issues an access request, using some particular data sublanguage; (2) the DBMS intercepts the request and interprets it; (3) the DBMS inspects, in turn, the external schema, and the external/conceptual mapping, the conceptual schema, the conceptual/internal mapping, and the storage structure definition; and (4) the DBMS performs the necessary operations on the stored database". [Date77]
>
> "The software that assists the application designer (who must decide how the data will be stored, accessed, displayed, etc.) and that later allows a computer user to enter, retrieve, and manipulate the information in the database is called a database management system, or DBMS". [Gutt84]
>
> "A data base management system is a software system used to manage and maintain data in a prescribed structure for the purpose of being processed by multiple applications independent of storage device class or access method. A data base management system organizes data elements in some predefined structure, cross-references defined relationships, and retains these relationships between different data elements within the data base". [Data84]

From the varied definitions stated above, it is understandable why so many of the microcomputer data management packages are being marketed as database management systems. This enforces the need to categorize these packages according to their functional capabilities instead of attempting to justify whether a package has all the necessary features that would make it a database management system [Fipsll0].

Data management software for microcomputers can be divided into two basic categories: single-file and multi-file systems (see FIGURE 1). Single-file systems allow users to operate on only a single database file at a time. With multi-file systems, users can operate on one or more database files at a time. In this document, we will consider a database file to have a logical structure that can be compared to a two dimensional array (i.e., a table structure).

1.1 SINGLE-FILE SYSTEMS

A single-file system is generally designed as a separate menu-driven package or as a functional part of an integrated package. As a separate menu-driven package, upon execution, a single-file system presents a user with a list of options (see FIGURE 2), generally referred to as the main menu. By making a selection from among this list of options, the user gains access to other sub-level menus (see FIGURE 3). As a functional part of an integrated package, the main menu lists the database function as an option. Once the user enters this functional level, the database functions are made available. Spreadsheet packages are examples of this type of data management software.

Single-file systems are designed to handle only a single file at a time. That is, facilities are not available to handle database file cross-referencing. Functions are limited to such single-file operations as inserting, deleting, updating, selecting, sorting and indexing. The insert operation stores data into a database file through a predefined record structure. Each insertion is called a record occurrence.

After storing a record occurrence in a database file, it can be removed through the delete operation. The delete operation locates a record occurrence through a specified constraint which uses the data values contained in the desired record occurrence as location controls. Once located, the content of the record occurrence is erased or the record is flagged as deleted.

Category	Model	Features
Single-File	Non-Relating	* menu driven * does not allow cross-referencing between data contained in separate data files * self-contained or part of an integrated package
Multi-file	Relational-like	* menu driven * allows cross-referencing of data contained in one file with data contained in another * ad hoc cross-referencing cannot be handled * related data files are mapped to separate structure definitions * structure definitions of related data files must reference each other
	Relational	* menu driven, command driven, or both * allows cross-referencing of data contained in one file with data contained in another * ad hoc cross-referencing is allowed * related data files are mapped to separate structure definitions * the structure definition of a data file does not reference the structure definition of a related data file * related structure definitions must have at least one attribute of the same type and size
	Hierarchical	* choice of command or menu dialogue * all data is stored in a single file * the structure definition that maps to the data file logically partitions the data into separate data files containing record occurrences of the same type * cross-referencing across these logically separate data files is accomplished through a concept called segment linking * ad hoc cross-referencing can not be handled, that is, all cross referencing must be handled through predefined segment links
	Network	* program driven * all data is stored in a single file * the structure definition that maps to the data file logically partitions the data into separate data files containing record occurrences of the same type * cross-referencing across these logically separate data files is accomplished through a concept called sets * ad hoc cross-referencing can not be handled, that is, all cross-referencing must be handled through predefined sets
	Free-Form	* menu driven * a record can contain any type of information and it does not have to resemble any other in the database * the orientation away from fixed form records tends to nullify the concept of multiple files; however, the capability to specify multiple indexes maintains the concept of associating records (i.e., cross-referencing) * adhoc cross referencing is allowed
	Multi-	* has the same characteristics as specified for either a relational or hierarchical model

Figure 1: The Spectrum of Microcomputer Data Management Software

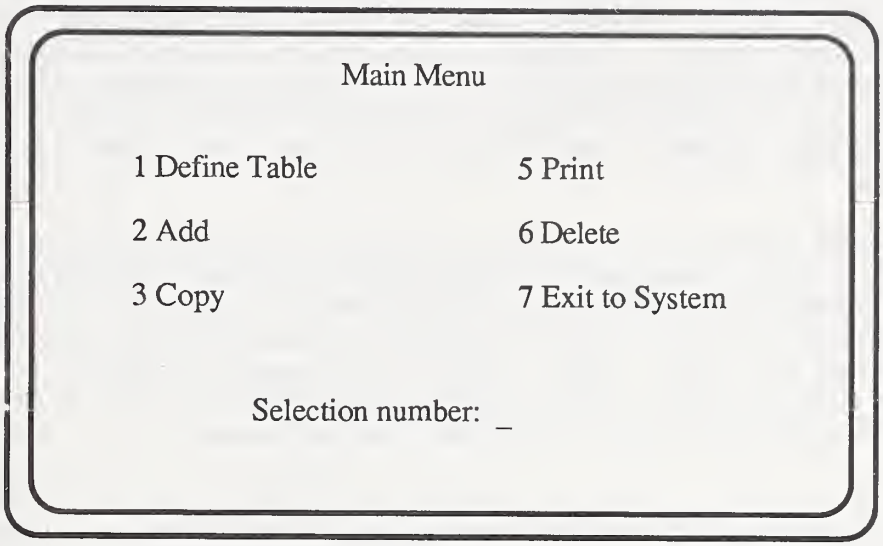


Figure 2: Sample User Menu

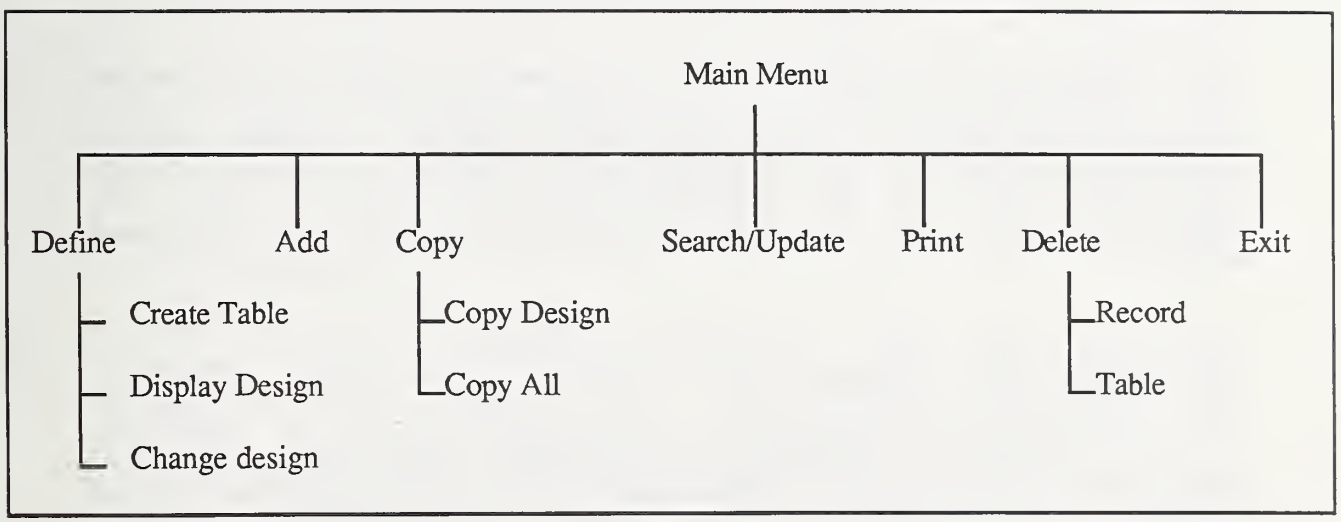


Figure 3: Typical Menu Hierarchy

Sometimes a record occurrence is only modestly corrupted and can be corrected through editing as opposed to deleting the entire record occurrence. Such editing can be handled through the update operation. As with the delete operation, the update operation requires a constraint clause to guide it to the desired record occurrence that is to be modified. Once the record occurrence has been located, changes can be made.

On other occasions, it will be necessary to extract information from a database file. This is handled through the select operation. Like the delete and update operations, the select operation uses a specified constraint to guide it to a desired record occurrence. However, data is only displayed with the select operation and is not disturbed as it is with the delete and update operations.

The sort and index operations control the physical or logical ordering of record occurrences stored in a database file. The sort operation rearranges the physical order of record occurrences within a database file. Record occurrences are arranged sequentially in accordance with the specified ascending or descending order of values in a specified field. The sorted results are placed in either a resident or a temporary result-file. If results are place in a temporary result-file, at the end of the sort process, the original database file may be replaced by the contents of the temporary result-file.

Record occurrences can also be sorted logically through the index operation. Thus, an actual physical reordering of record occurrences is not performed; instead, an index file is generated from the specified fields (attributes) of record occurrences. This index file controls the order in which records are accessed during sequential search operations.

1.2 MULTI-FILE SYSTEMS

Multi-file systems include all the functions of single-file systems plus additional functions such as joins, unions, intersections, comparisons, differences, file-linkings, and projections. The join operation, through either implicit or explicit links, relates record occurrences contained in different database files.

Implicit links allow relationships to be handled in an ad hoc fashion while explicit links are rigid. Explicit links involve the use of file-link information that is prespecified in the data definition tables associated with the database. This file-link information will name the files to

be linked and the fields upon which the files are to be related. Some microcomputer data management packages that use this method to join database files allow only unidirectional relationships between files. That is, the database file that is being pointed to cannot point back to the file that is doing the pointing. Some packages that use this method of joining limit the number of files to which a file can be linked.

The union operation generates a resulting database file that contains record occurrences from either or all specified database files. Each database file which serves as a parameter for the operation must have identical data structures. Thus, the order, type and size of fields are of paramount importance. Generally, the union operator for microcomputer data management packages operates on only two database files at a time.

The intersect operation generates a resulting database file that contains only those record occurrences that are identical across specified database files. Like the union operation, the intersect operation requires the specified database files to have identical data structures, and again, most microcomputer data management packages limit the number of specified files to two.

The compare operation validates record occurrences in one table against those in another; that is, one table acts as a lookup-reference for the other, based on a specified key field. Unlike the union and intersect operations, the compare operation does not require identical data structures; however, the database file that is used as the lookup reference must have at least one field of the same type and size as a field in the database file that it is compared against. This operation accepts only two database files as its parameters.

The difference operation generates from two specified database files a resulting database file that contains record occurrences that are not common across the specified database files. Like the union and intersect operation, the difference operation requires the database files specified as its parameters to have identical data structures.

The project operation extracts and displays data from a subset of the fields contained in each record occurrence that is returned by the select operation. The preferred data fields are specified as part of the parameter-list submitted to the select operation or as parameters submitted directly to the project operation itself. Thus, microcomputer data management packages may combine the select and project operations or treat them as separate operations.

Multi-file systems allow access to their functional capabilities through menus, menus in conjunction with forms, or command language statements. Some systems offer all of these methods. In addition to these, other systems allow interfacing through programming languages such as FORTRAN, COBOL, BASIC, PASCAL, C or their own built-in programming-like language.

There is a spectrum of functional capabilities among the multi-file systems (see FIGURE 1). This ranges from relational-like systems to free-form systems and includes relational, heirarchical, network and multi-model systems.

1.2.1 Relational-Like.

Relational-like systems handle projections and joins in a superficial fashion (with respect to the definition of the relational model). Some of these systems allow projections to be performed only after the user has defined an output structure containing the desired attributes (fields). To simulate joins, users are required to specify (at definition time) which database file is to be related to the file being defined. This requirement is handled by establishing an explicit link (pointer). The explicit link is usually a combination of the related file's name and the name of the field upon which the link will be based. FIGURE 4 illustrates a relational-like data structure. Other relational-like systems handle joins through a built-in programming-like language or a supported programming language such as BASIC, FORTRAN, COBOL, PASCAL or C. This may require a complicated programming effort.

1.2.2 Relational.

Relational systems use data values (implicit links) to relate database files. Thus, data definitions are self-contained; that is, the database definition does not reference related database files. FIGURE 5 illustrates a database file in relational systems. Through the use of implicit links, relational systems are designed to handle projections and joins with a minimum amount of user effort. The user can develop projections without having to define an output structure which contains the desired attributes (fields). This is accomplished as a result of the relationship between the project and select operations. The project operation is incorporated as part of the select operation which allows desired attributes to be submitted as part of its parameter-list. The relational system uses this parameter-list, in conjunction with the definition of the queried database file, to control the form of projected output. Joins

```
Phase I:  Define a logical structure
          for each database file.


          Enter name of database file:  Emp

          ENTER RECORD STRUCTURE AS FOLLOWS:
            FIELD       NAME,TYPE,WIDTH,DECIMAL PLACES
            001         name,c,20
            002         deptno,n,3
            003         address,c,45
            004         city,c,15
            005         state,c,3
            006         zip,n,5
            007         salary,n,6,2
            008         hdate,d
            009 <return>
          ENTER ANOTHER STRUCTURE? (Y/N)y

          Enter name of database file:  Dept

          ENTER RECORD STRUCTURE AS FOLLOWS:
            FIELD       NAME,TYPE,WIDTH,DECIMAL PLACES
            001         deptno,n,3
            002         deptname,c,15
            003 <return>
          ENTER ANOTHER STRUCTURE? (Y/N)n



Phase II:  Specify the fields on
           which related database
           files are to be linked.


           Enter name of database file:  Emp
           Enter name of link field:  deptno
           Enter name of related database file:  Dept
```

Figure 4: Defining a Relational-Like Data Structure

```
Enter the name of the database file:  Emp

ENTER RECORD STRUCTURE AS FOLLOWS:
  FIELD       NAME,TYPE,WIDTH,DECIMAL PLACES
  001         name,c,15
  002         deptno,n,3
  003         address,c,45
  004         city,c,15
  005         state,c,3
  006         zip,n,5
  007         salary,n,6,2
  008         hdate,d
  009 <return>

ENTER ANOTHER STRUCTURE? (Y/N)y


Enter the name of the database file:  Dept

ENTER RECORD STRUCTURE AS FOLLOWS:
  FIELD       NAME,TYPE,WIDTH,DECIMAL PLACES
  001         deptno,n,3
  002         deptname,c,15
  003 <return>

ENTER ANOTHER STRUCTURE? (Y/N)n



****Note:  A command syntax simular to the following would
           be used to JOIN the Emp file to the Dept file.


           SELECT name
             FROM Dept,Emp
             WHERE Dept.deptno EQ Emp.deptno;
```

Figure 5: Defining a Relational Data Structure

are handled through either a form of relational algebra or relational calculus. If a system uses relational algebra, the user is restricted to joining two database files at a time. The generated output from the join is placed in a third file referred to as the result-file. The joining of any results deposited in this result-file with additional database files is the only way to extend joining across multiple database files. Thus, if a user desires to join database files A, B and C, the user must generate a result-file by joining database files A and B, first. Then, the generated result would be joined to database file C which in turn would generate the final result.

Relational algebra is a sufficient method for handling simple joins; however, it can be burdensome to use when joining across more than two database files (as illustrated in the above paragraph). This burden is relieved through the use of relational calculus which uses all the primitive operations performed by relational algebra but without the user being aware of any intermediate results when joining more than two database files.

1.2.3 Hierarchical.

Hierarchical systems permit defining data structures in terms of segments which are very similar to what is called records in other systems. However, the segment is comprised of data elements and repeating groups. The data elements are equivalent to attributes in a record definition. The repeating group describes a structure for storing a set of data elements. Repeating groups serve a twofold purpose: they link the levels of a hierarchial structure and incorporate the concept of multiple files. Each repeating group can be viewed as a file containing a single record type. This simulates defining more than one record type per data definition file. FIGURE 6 is an example of data structures using the hierarchical model.

1.2.4 Network.

As with hierarchical systems, network systems permit defining more than one record type per data definition file. However, the concept of a record is explicit instead of being implied by repeating groups. The capability of defining more than one record type per file incorporates into the network model the concept of multiple files. Additionally, the network model uses the concept of a set to link (relate) the different record types. FIGURE 7 illustrates a data definition for the network model.

```
1*  DEPT-NUMBER (KEY INTEGER 9(3)):
2*  DEPT-NAME (NON-KEY CHARACTER X(15)):
3*  EMPLOYEES (RG):
4*  EMPLOYEES NAME (KEY CHARACTER X(15) IN 3):
5*  ADDRESS (NON-KEY CHARACTER X(45) IN 3):
6*  CITY (NON-KEY CHARACTER X(15) IN 3):
7*  STATE (NON-KEY CHARACTER X(3) IN 3):
8*  ZIP (NON-KEY INTEGER 9(5) IN 3):
9*  SALARY (NON-KEY DOLLAR $9(3).99 IN 3):
10*  DATE-HIRED (NON-KEY DATE MM/DD/YY IN 3):
```

Figure 6: Hierarchical Data Definition

```
/*****   IDENTIFICATION SECTION   *****/

db DEPARTMENTS
      file "C:DEPT.DB"
      size 300 pages, page size 1024
      title "DEPARTMENTS schema"

/*****   RECORD SECTION   *****/

record DEPARTMENT
     item DEPTNO int 3
     item DEPTNAME chr 15

record EMPLOYEE
     item NAME chr 15
     item ADDRESS chr 45
     item CITY chr 15
     item STATE chr 3
     item ZIP int 5
     item SALARY dollar
     item HDATE date

/*****   SET SECTION   *****/

set SYSSET, type 1:n
     owner SYSTEM
     member DEPARTMENT, order sorted ascending (DEPTNO)
                        duplicate not allowed
                        insertion automatic

set EMPLOYEES, type 1:n
     owner DEPARTMENT
     member EMPLOYEE, order sorted ascending (NAME)
                      duplicate allowed
                      insertion automatic

end
```

Figure 7: Network Model Data Definition

1.2.5 Multi-model.

Multi-model systems permit users the option of defining data structures in accordance with one of the three most popular models (i.e., hierarchical, network or relational). FIGURE 8 illustrates a data structure in multi-model systems. These systems are useful for prototyping, because they aid in identifying the most appropriate model for candidate applications. Thus, the functional capabilities available to users depend on the type of data structure being used. For example, if the data structure satisfies the hierarchical model, only the functional capabilities of a hierarchical model are available to the user.

1.2.6 Free-form.

Free-form systems allow users the option of entering data into a database file in any format, such as text, tables or numbers. Whatever the format, each information unit may be associated with one or more keywords. For example, a paragraph of text could be assigned the keywords "microcomputer" and "data management". The paragraph could then be retrieved by asking the system for any information on microcomputer, data management, or both. In free-form systems, the concept of a fixed record format is not available. Thus, information can be entered without regard to previous entry forms. FIGURE 9 is an example of defining structures for free-form systems.

```
FILENAME=Sample, EXTENSION='DBF' #

SEGNAME=Dept, SEGTYPE=root #
   FIELDNAME=deptno, FIELDTYPE=number, FIELDLENGTH=3 #
   FIELDNAME=dname, FIELDTYPE=character, FIELDLENGTH=15 #

SEGNAME=Emp, SEGTYPE=child, PARENT=Dept #
   FIELDNAME=ename, FIELDTYPE=character, FIELDLENGTH=15 #
   FIELDNAME=address, FIELDTYPE=character, FIELDLENGTH=45 #
   FIELDNAME=city, FIELDTYPE=character, FIELDLENGTH=15 #
   FIELDNAME=state, FIELDTYPE=character, FIELDLENGTH=3 #
   FIELDNAME=zip, FIELDTYPE=number, FIELDLENGTH=5 #
   FIELDNAME=salary, FIELDTYPE=dollar, FIELDLENGTH=6 #
   FIELDNAME=hire-date, FIELDTYPE=date #
```

Figure 8: Multi-Model Data Definition

PHASE 1: DEFINE INDEXED ITEMS

Item Name	Type	Length	Pattern
DEPTNO	INTEGER	3	
ENAME	CHARACTER	15	

PHASE 2: DEFINE NONINDEXED ITEMS

Item Name	Type	Length	Pattern
DNAME	CHARACTER		
ADDRESS	CHARACTER		
CITY	CHARACTER		
STATE	CHARACTER	2	
ZIP	INTEGER		
SALARY	MONEY	7	$$$$.$$
HIRE-DATE	DATE	8	MM-DD-YY

PHASE 3: DEFINE RECORDS

Record Name	Items
DEPARTMENT	DEPTNO, DNAME
EMPLOYEE	ENAME, ADDRESS, CITY, STATE, ZIP, SALARY, HIRE-DATE

PHASE 4: DESIGN/SAVE SCREEN FORMS FOR EACH RECORD

DEPT

DEPTNO:
DNAME:

EMP

ENAME:
ADDRESS:
CITY:
STATE: ZIP:
SALARY: HIRE-DATE:

Figure 9: Free-Form Data Definition

2. GENERAL FEATURES

Most data management software for microcomputers is single-user oriented. That is, the more popular packages are designed to run under single-user operating systems (for microcomputers). This is predominantly the case for packages that have been designed to run under 8-bit architectures. Under 16-bit architectures, both single- and multi-user operating systems are generally available. As a result, some vendors offer both single- and multi-user versions of their data management packages. Under 32-bit architectures, multi-user versions are predominant. Regardless of the category, there are four basic features common to all of these packages. These features are:

- o data definition,
- o data entry,
- o data retrieval, and
- o report generators.

Each of these common features will be discussed in the following sections.

2.1 DATA DEFINITION

The data definition specifies the logical framework through which data structures are accessed by users. The data definition feature of those microcomputer data management packages that fall into the categories single-file, relational-like, or relational allow users to define data structures in terms of two-dimensional tables consisting of rows and columns. Entry of the data definition for these structures is generally accomplished through either:

- o an attribute file,
- o a form file, or
- o parallel development of attribute and form files.

An Attribute File - In the first approach, the user is prompted to enter the attributes of each field according to a predefined format. FIGURE 10 illustrates this approach. The NAME attribute is generally limited to a maximum of ten characters. The base set of data TYPES usually includes the NUMERIC and CHARACTER data types; however, most packages extend this set with the DATE data type. The WIDTH is the maximum number of characters that may be contained in an assigned data value. The DECIMAL PLACES is the specified precision in digits to the right of the decimal point for numeric data types.

A Form File - A second way to enter the definition for data structures is for the user to build a form file. This is, in effect, defining the data structure by designing its data entry form. As shown in FIGURE 11, this method requires attribute names to be enclosed in square brackets or some delimiting symbols. The permitted length in characters for assigned data values is indicated by an appropriate number of underline keystrokes. After building the form file, the user is then prompted to specify the data type for each attribute painted as part of the screen form.

Parallel Development - A third way to enter the definition for data structures is for the user to specify the data types for each attribute as the form is being developed. As shown in FIGURE 12, this method is generally a menu driven process. After the user selects the file creation option and specifies a filename, the microcomputer data mangement system enters a screen-edit mode. In this mode, scrolling the cursor to any position on the screen is possible. This feature enables the user to create a form. The user simply scrolls the cursor to any position on the screen and enter an attribute name. Once the end of the attribute name is indicated, the system highlights another set of commands in the menu window beneath the designated screen-edit area. At this point, the characteristic of assignable values (such as data type, size, precision and range) can be specified.

Those microcomputer data management systems that satisfy the characteristics of either the hierarchical or network models provide another means of entering the definition of data structures. Entry of the definition is generally achieved through the aid of a text editor. After creating the definition file through the aid of a text editor, it is submitted to a data definition processor which uses the information in this file to generate a data dictionary file and to initialize the database file needed to

```
ENTER RECORD STRUCTURE AS FOLLOWS:
  FIELD       NAME,TYPE,WIDTH,DECIMAL PLACES
  001         name,c,20
  002         address,c,25
  003         city,c,20
  004         state,c,2
  005         zip-code,c,5
```

Figure 10: Data Field Attribute Specification

```
Enter a new form (Y/N)?y

[NAME]:____________________

[ADDRESS]:_________________________

[CITY]:____________________   [STATE]:__   [ZIP-CODE]:_____


Enter data definition in the following format:

>FIELD-NAME: FIELD-TYPE, FIELD-SIZE, MIN-VALUE, MAX-VALUE, "DEFAULT-VALUE"
   Choices :  (ANJ$R)    (1-127 bytes)   (1-10 digits)    (0-15 Characters)

>1.NAME:a,20
>2.ADDRESS:a,25
>3.CITY:a,20
>4.STATE:a,2
>5.ZIP-CODE:a,5
```

Figure 11: Data Definition through Form-Building

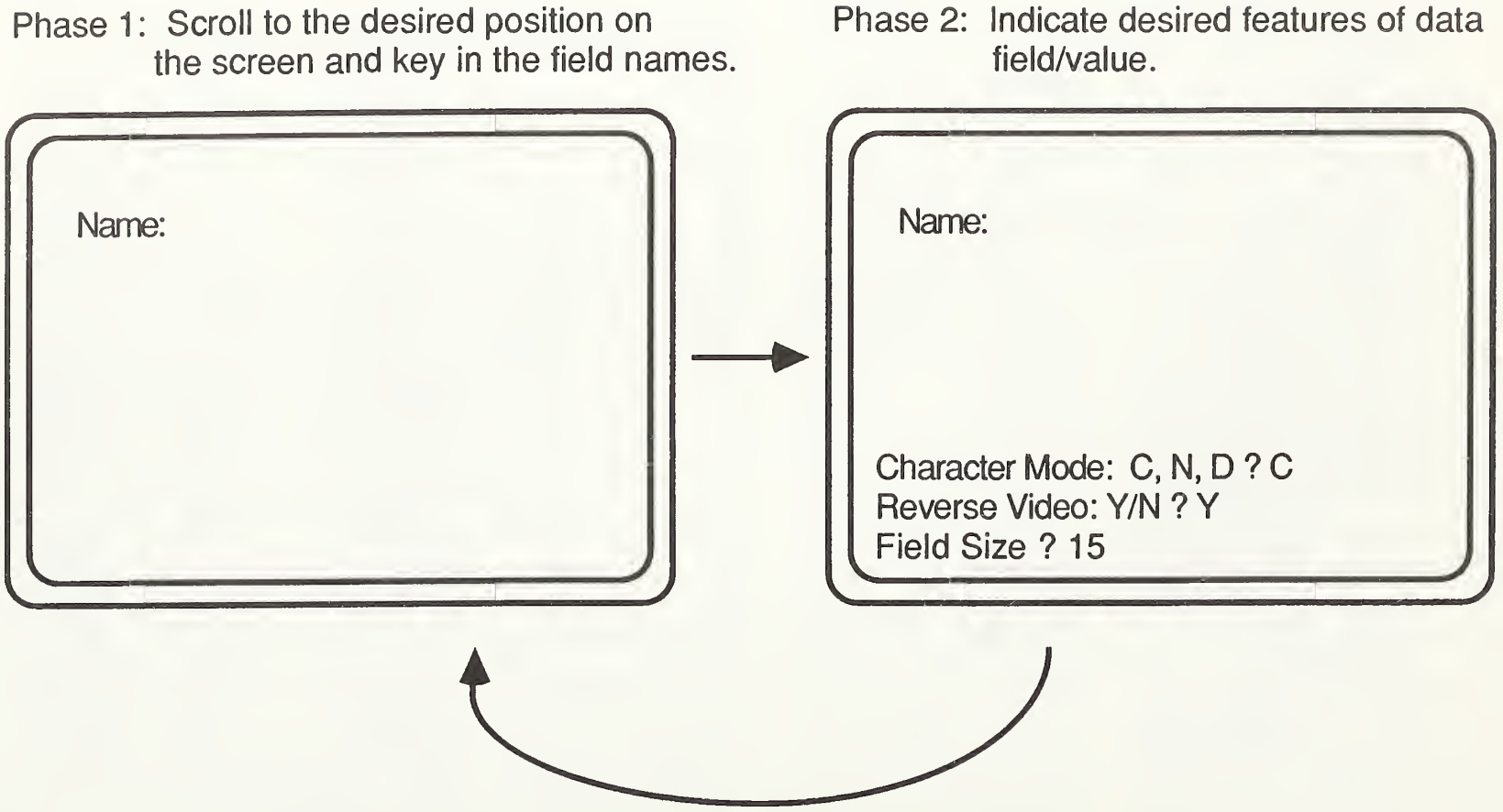


Figure 12. Screen-Oriented Form Building

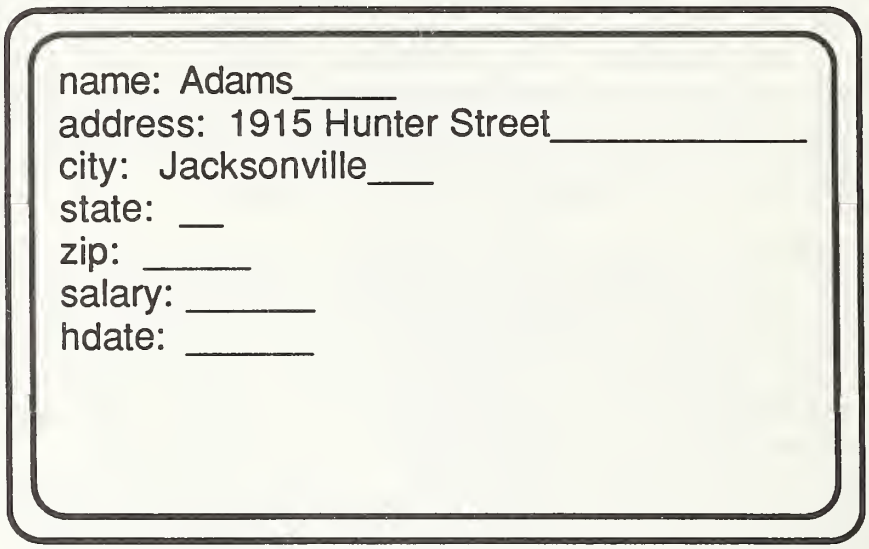
store record occurrences. An example of the data definition file for the hierarchical model is shown in FIGURE 6. Inspection of this file will reveal a logical view that is built around the concept of nesting children records within an associated parent record. A parent record contains at least one attribute that serves as a label for a list of children attributes. In FIGURE 6, line 3* is such a parent attribute. This attribute explicitly links the attributes of the parent record (i.e., DEPT-NUMBER and DEPT-NAME) with the attributes of a child record (i.e., EMPLOYEES, ADDRESS, CITY, STATE, ZIP, SALARY and DATE-HIRED).

An example of the definition file for the network model is shown in FIGURE 7. As shown, the definition of a data structure requires identifying three sections. In the first section, the Identification Section, the logical naming label for the database is linked to the actual database file (i.e., DEPARTMENTS is linked to C:DEPT.DB). In addition, the expected amount of storage space is specified. The second section, the Record Section, identifies the name of each record type along with the names and characteristics of their respective attributes. In FIGURE 7, there are two record types, DEPARTMENT and EMPLOYEE. In the third section, the Set Section, the relationships among the different record types are indicated. Each relationship is assigned a set name. Each set contains an owner record type and one or more member record types. In FIGURE 7, the set section identifies two set types, SYSSET and EMPLOYEES. SYSSET is a special set type that serves as the entry point for the data structure.

2.2 DATA ENTRY

Data entry in a large number of microcomputer data management packages is accomplished through facilities that prompt a user to enter a value for each defined attribute comprising a record occurrence. Generally, the prompting is through a screen form that was previously designed by the user, or determined by the data management system from the previously specified data definition. The screen form will, in most cases, indicate the allowed number of characters for an attribute by an appropriate number of underscores or a boxed-in number of character positions highlighted in reverse video. FIGURE 13 illustrates this method of data entry. In the first case, the underscores are overwritten as data values are entered. In the case of reverse video, the characters for data values are accented by a rectangular shape.

Case 1: Use of underscores to indicate the maximum character length of data values



Case 2: Use of reverse video to indicate the maximum character length of data values

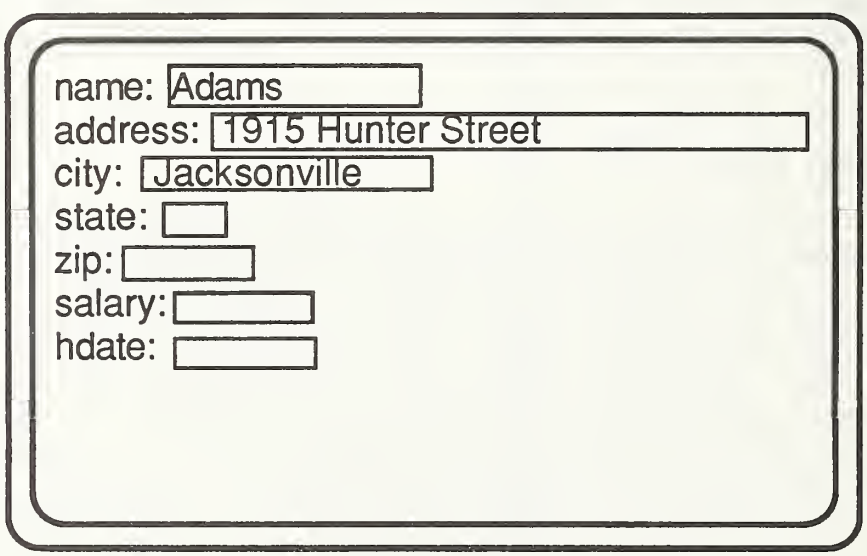


Figure 13: Screen-Oriented Data Entry

Additionally, many packages support a bulk loading facility. Such a facility requires the input file to be in a format that is familiar to the data management package. A format that is common across many packages is an ASCII file containing attribute values separated by commas. Each line in the ASCII file must contain only enough attribute values for a single occurrence of a record type. The order of the attribute values must match the order of the attribute names in the definition for the associated record type in the database.

2.3 DATA RETRIEVAL

Data retrieval in microcomputer data management packages is performed through either menu or non-menu query facilities. Some packages that are driven completely by menus allow users to control the format of the menus while others do not. In either case, at execution time a set of options is presented from which the desired data retrieval action can be specified. The retrieved data can then be displayed on the screen, directed to a printer, or stored in a file.

Those packages that support only non-menu query capabilities use either a query command syntax or an interface with a programming language such as BASIC, FORTRAN, COBOL, PASCAL, or C. The query command languages retrieve data through command statements that are syntactically correct. An example of such a command statement is shown in FIGURE 14. This command statement would list those records from the table that satisfy the specified constraint (FOR comm > 1000).

Data retrieval with a programming language (e.g. COBOL) is achieved through call statements or embedded query command statments. If call statements are used, an appropriate sequence of parameters must be passed to the routine that is functionally capable of handling the desired request. An example of call statements in a COBOL program is shown in FIGURE 15. This program can be submitted directly to the COBOL compiler; however, a program with embedded query command statements must be submitted to a precompiler before being submitted to the compiler of the programming language. There are, however, advantages in not having to learn the necessary call routines and the required parameters that must be passed to them. Instead, a user has to learn only the syntax of the query language and place the query statements at the proper location within the lines of programming code. After learning the syntax for the query language, embedding a COBOL program with query statements will generally be easier than working with CALL statements. Also, the code of a

Database File: Employee

Empno	EName	Job	Sal	Comm
6359	Milton	Clerk	1,300.00	
6495	Allen	Salesman	1,600.00	300.00
6621	Jones	Analyst	2,800.00	
7031	Martin	Salesman	1,800.00	1,200.00
7545	Scott	Clerk	1,500.00	
7820	Sims	Salesman	1,200.00	1,500.00

Objective – to select names of those employees that have commissions greater than $1,000.00.

Command Syntax –

```
SELECT EName
    FROM Employee
    WHERE Comm > 1000;
```

Figure 14: Sample Query Statement

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SAMPLE.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  DBLOGON_AREA.
    02  DBLOGON_ERROR          PIC S999.
    02  DBLOGON_WA             PIC S999 OCCURS 20 TIMES.
01  CURSOR.
    02  CURSOR_ERROR           PIC S999.
    02  CURSOR_WA              PIC S999 OCCURS 30 TIMES.
01  OUTPUT_RECORD.
    02  NAME                   PIC X(15).
    02  FILLER                 PIC XX VALUE SPACES.
    02  JOB                    PIC X(40).
    02  FILLER                 PIC XX VALUE SPACES.
    02  AGE                    PIC S999.
    02  FILLER                 PIC XX VALUE SPACES.
    02  SAL                    PIC S9(6)V99.
    02  FILLER                 PIC XX VALUE SPACES.
    02  COMM                   PIC S9(6)V99.
77  USER-ID                    PIC X(7) VALUE "manager".
77  SQL-SELECT                 PIC X(60) VALUE
    "SELECT NAME,JOB,AGE,SAL,COMM FROM EMPLOYEE WHERE COMM > 1000".
PROCEDURE DIVISION.
BEGIN.
    CALL "DBLOGON" USING DBLOGON_ERROR, USER-ID.
    IF DBLOGON_ERROR NOT = 0
        PERFORM DB-ERROR
        GO TO EXIT-STOP.
    CALL "OPENCURSOR" USING CURSOR_ERROR, DBLOGON_ERROR.
    IF CURSOR_ERROR NOT = 0
        PERFORM DB-ERROR
        GO TO EXIT-DB.
    PERFORM DISPLAY-OUTPUT THROUGH EXIT-OUTPUT
        UNTIL CURSOR_ERROR = 8.
EXIT-CLOSE.
    CALL "CLOSE_CURSOR" USING CURSOR_ERROR.
    IF CURSOR_ERROR NOT = 0
        PERFORM DB-ERROR.
EXIT-DB.
    CALL "LOGOFF_DB" USING DBLOGON_ERROR.
    IF DBLOGON_ERROR NOT = 0
        PERFORM DB-ERROR.
EXIT-STOP.
    STOP RUN.
DISPLAY-OUTPUT.
    CALL "EXECUTE_SQL" USING CURSOR_ERROR, SQL_SELECT, NAME,
         JOB, AGE, SAL, COMM.
    IF CURSOR_ERROR NOT = 0
        IF CURSOR_ERROR NOT = 8
            PERFORM DB-ERROR
            GO TO EXIT-OUTPUT.
    DISPLAY OUTPUT-RECORD.
EXIT-OUTPUT.
    EXIT.
DB-ERROR.
    IF DBLOGON_ERROR NOT = 0
        DISPLAY "Experiencing database error" DBLOGON_ERROR
    ELSE
       IF CURSOR_ERROR NOT = 0
        DISPLAY "Experiencing table error" CURSOR_ERROR.
```

Figure 15: Database Query through Programming Language CALL Statements

program that is embedded with query command statements is easier to understand than one with embedded CALL statements. FIGURE 16 is an example of a COBOL program that is embedded with query command statements.

Additionally, some microcomputer data management systems offer the choice of data access through either menu or non-menu facilities. These systems attempt to offer the best of both worlds. Menus for their ease of use and non-menus for user sophistication.

2.4 REPORT GENERATORS

Report writer facilities for microcomputer data management packages generally fall into two categories: free-form report writers and fixed-form report writers. Free-form report writing is accomplished either through a screen editing mode which allows the user to scroll upon the screen painting a desired report format, or through screen utility commands embedded in a command file. In either case, the set of format controls is linked to the database file which contains the data needed to generate the desired report. FIGURE 17 illustrates use of a command file embedded with screen utility commands.

Fixed-form report writing requires users to supply appropriate responses to a set of prompts. The set of prompts presented is to some degree determined by the user's responses. FIGURE 18 illustrates use of a fixed-form report writer.

```
IDENTIFICATIONN DIVISION.
    PROGRAM-ID.  SAMPLE.
ENVIRONMENT DIVISION.
DATA DIVISION.
SUB-SCHEMA SECTION.
    DB  COMPANYDB
        SQLCODE IS DB-STATUS.
    TD  TABLE SECTION.
        01  EMPLOYEE.
            02  NAME PIC X(15).
            02  JOB  PIC X(40).
            02  AGE  PIC S999.
            02  SAL  PIC S9(6)V99.
            02  COMM PIC S9(6)V99.
WORKING-STORAGE SECTION.
    01  LASTREC   PIC X(3)     VALUE "NO".
    01  DB-STATUS PIC S9(5)    VALUE 0.
PROCEDURE DIVISION.
    DECLARATIVES.
    LASTREC-ERROR SECTION.
        USE FOR DB-EXCEPTION ON +100.
        MOVE "YES" TO LASTREC.
    ABNORMAL-ERROR SECTION.
        USE FOR DB-EXECPTION ON OTHER.
        DISPLAY "DB ERROR". ROLLBACK. STOP RUN.
    END DECLARATIVES.
OPEN LISTEMP CURSOR FOR
    SELECT NAME, JOB, AGE, SAL, COMM
        FROM EMPLOYEE
        WHERE COMM IS GREATER THAN 1000.
FETCH LISTEMP.
PERFORM UNTIL LASTREC = "YES".
    DISPLAY NAME, JOB, AGE, SAL, COMM.
    FETCH LISTEMP.
END-PERFORM.
STOP RUN.
```

Figure 16. Database Query using Imbedded Query Language Statements

```
Screen Command Statements

ERASE SCREEN
OPEN Employees_table
@10,5 SAY 'Name:' GET Name
@12,5 SAY 'Address:' GET Address
@14,5 SAY 'State:' GET State
@14,30 SAY 'Zip Code:' GET Zip-Code
@17,5 SAY 'Salary:' GET Salary
@17,27 SAY 'Hdate:' GET Hdate


Output Format

     Name: [                  ]

     Address: [                                          ]

     State: [                  ]  Zip Code: [     ]


     Salary: [            ]    Hdate: [        ]
```

Figure 17: Free-Form Report Specification

System Prompts for User Responses

```
ENTER NAME OF REPORT:Example
ENTER OPTIONS,M=LEFT MARGIN,L=LINES/PAGE,W=PAGE WIDTH:W=65
PAGE HEADING? (Y/N)y
ENTER PAGE HEADING:List of Employees
DOUBLE SPACE REPORT? (Y/N)n
ARE TOTALS REQUIRED? (Y/N)n
ENTER HEADING:< Name
COL          WIDTH,CONTENTS
001          15,NAME
ENTER HEADING:< Job
002          20,JOB
ENTER HEADING:< Manager
003          15,MANAGER
ENTER HEADING:< Salary
004          9,SAL
ENTER HEADING:< 'Hire Date'
005          10,HDATE
ENTER HEADING:<
```

Output Format

Name	Job	Manager	Salary	Hire Date
Johnson	President		$4,500.00	Jan-18-85
Miller	Clerk	Brown	$1,500.00	Feb-01-83
Brown	Manager		$2,500.00	Mar-05-81
		.		
		.		
		.		
		.		

Figure 18: Fixed-Form Report Specification

3. SELECTION ISSUES

Selection of microcomputer data management software should not be performed haphazardly. A prospective user should consider and employ selection criteria such as the following [Gall84]:

- o application requirements,
- o desired data management features,
- o user support,
- o existing and projected user base, and
- o price.

Each criterion will be discussed in the following sections. FIGURE 19 illustrates the activities that occur when applying this selection scheme with the evaluation methodology discussed in Section 4.

3.1 APPLICATION REQUIREMENTS

An assessment of what is functionally required to implement candidate applications should be performed before procuring any microcomputer data management software. The functional assessment will identify the category of data management software that will best handle the implementation of candidate applications. If an application can be handled through nonrelated database files, the data management packages that fall into the single-file category will be an appropriate selection. In contrast, applications that require cross referencing of two or more database files are handled best by those packages that fall into the multi-file category. However, the variety of models in this category makes the selection process a difficult one. The relational-like, hierarchical, and network packages are all appropriate for applications that have stable data structures. Such structures require studying the relationship among data items so that storage and access methods for best performance can be defined at data definition time. Additionally, a stabilized data structure demands that a database be defined around the type of queries that will be made

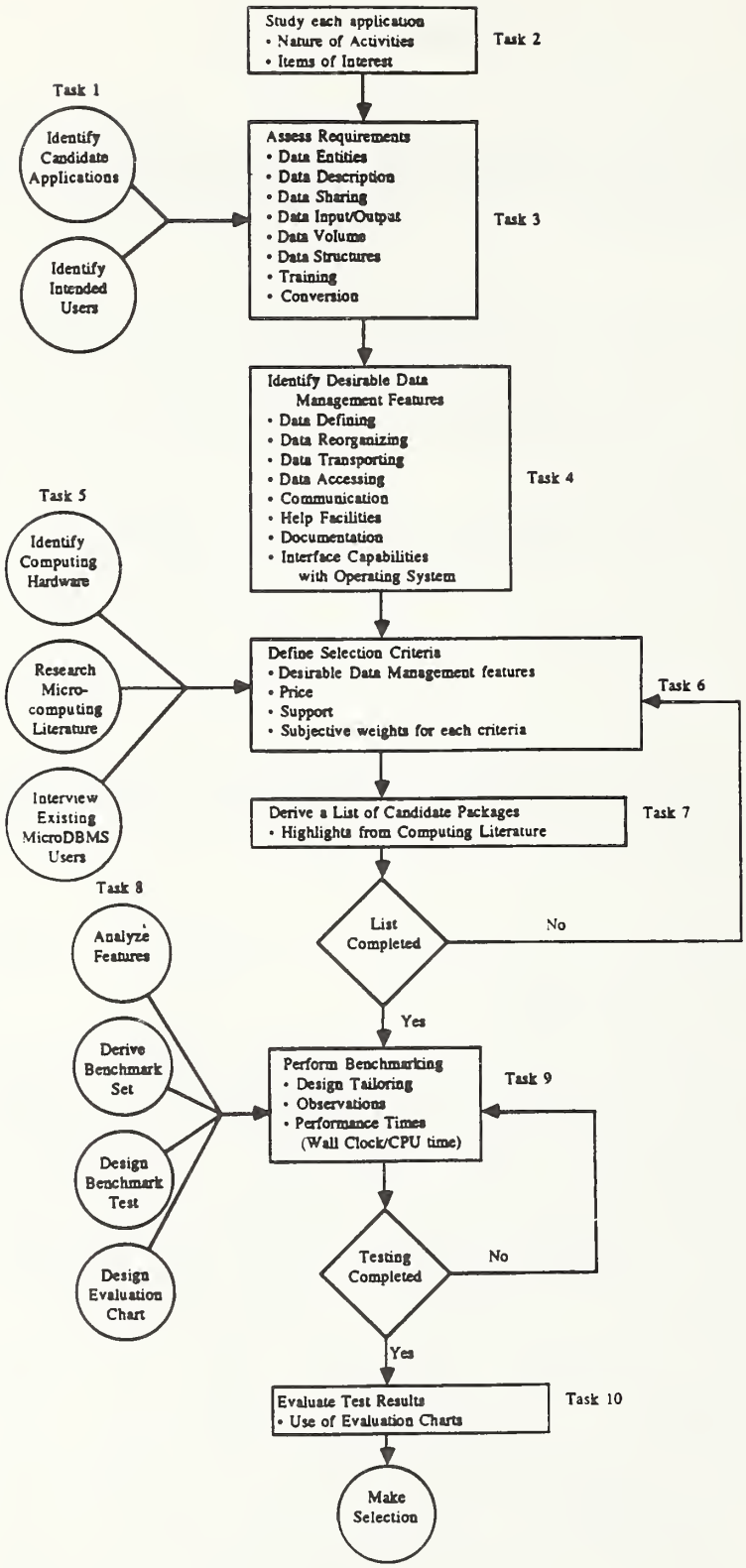


Figure 19: Evaluation and Selection Process

against it. Thus, to insure a stable data structure, a lot of work must be performed before defining the data structure.

Relational packages are often more appropriate for applications that do not have stable data structures; that is, data structures can be changed easily on a periodic basis without affecting the functional capabilities of software developed for the applications [Jame84]. Thus, relational packages allow complete data structures to be defined in stages. They are also good at handling applications that require unforeseen queries.

Free-form packages lend themselves to applications that are not disciplined by any specific data structure (i.e., hierarchical, network, or relational). Thus, a predefined data-entry format is not necessary. Instead, each record occurrence is associated with a unique format (i.e., a format that is specific to the set of attributes for that record occurrence).

Multi-model packages serve as excellent prototyping tools. They allow for performance testing on a small database of the desired application under different models before committing full software development for a specific model. Multi-model packages are also good for tracking data structures from a dynamic to a stable state. For example, during the dynamic period, the relational model could be used. Once the data structure stabilizes, either the hierarchical or network model could be used. The final model selection would be determined by the one that offered the best performance.

Other considerations during the needs assessment are the characteristics of the data and the physical limitations that are characteristic of microcomputers. The main physical limitation of microcomputers is data storage. A potential purchaser of data management software should identify both internal and external memory requirements for a candidate package. The user should verify whether internal memory requirements by the package can be adequately handled by the target microcomputer. If internal memory requirements for the package exceed the amount of internal memory installed in the target microcomputer, the cost associated with the necessary expansion should be estimated. As for external memory requirements, it should be determined whether the capacity of a single diskette, if the target microcomputer has only floppy disk drives, can store all of a package's software. This point is emphasized because of the cumbersome operating procedures that will be required should a single diskette prove to be insufficient storage space for all the software of a package. An example of the difficulty that

could arise is the user having to manually swap diskettes to find the software needed for a desired operation. To combat this situation, a user can select a package that has a storage requirement that is less than the capacity of a single diskette or target a microcomputer that has a hard disk drive.

Another key evaluation factor is the size of database files for each candidate application. Limitations on database file sizes are both physical and logical. The physical limitation has to do with the type of disk medium available on the target microcomputer (e.g., hard disk versus floppy diskette). For example, a hard disk generally allows storage in excess of 10 million characters while a typical floppy diskette allows only about 360 thousand characters.

Logical limitations result from the constraint that is placed on the logical design of a database file by a candidate package. For example, the maximum number of characters that can be assigned as a value for an attribute may be insufficient for a given application. Thus, if assigning memo notes or abstracts that have character string lengths of a thousand or more characters is required, it would be impossible in a system which allows a maximum of only 132 characters. Other examples of logical limitations include the maximum number of records per database file, attributes (fields) per record, and database files per database.

3.2 DESIRED DATA MANAGEMENT FEATURES

Every microcomputer data management package has a means of allowing users to define data structures and to logically store and retrieve data from these structures. However, the manner in which these features are made available to users can be crucial to the performance of a candidate application. Some issues pertaining to desirable data management features are listed below:

- o data definition and reorganization,
- o data transporting,
- o data accessing,
- o communication,

- documentation and help facilities, and
- interface capabilities to operating system and hardware.

Each of these features will be discussed in the following sections.

3.2.1 Data Definition and Reorganization.

The data definition language is central to the operations of any data management package. Based on this premise, users should consider the following list of features when selecting from a set of candidate data management packages:

- the ease by which data definitions can be specified,
- the set of available data types,
- the database indexing facilities,
- the levels of data integrity,
- the levels of data security, and
- the ease by which data definitions can be modified.

Ease of Specification - The above features determine the degree of flexibility available to users when defining data structures. For example, a data definition facility that offers screen editing capabilities makes it easer to enter and change the definition of data structures.

Data Types - A large set of data types increases the strength of operations that can be performed on the defined data structures. An illustration of this is a microcomputer data management package that has only the CHARACTER and NUMBER data types. With such a package, implementing an application requiring date operations could be very difficult and, in some cases, the associated cost in time would make it impractical.

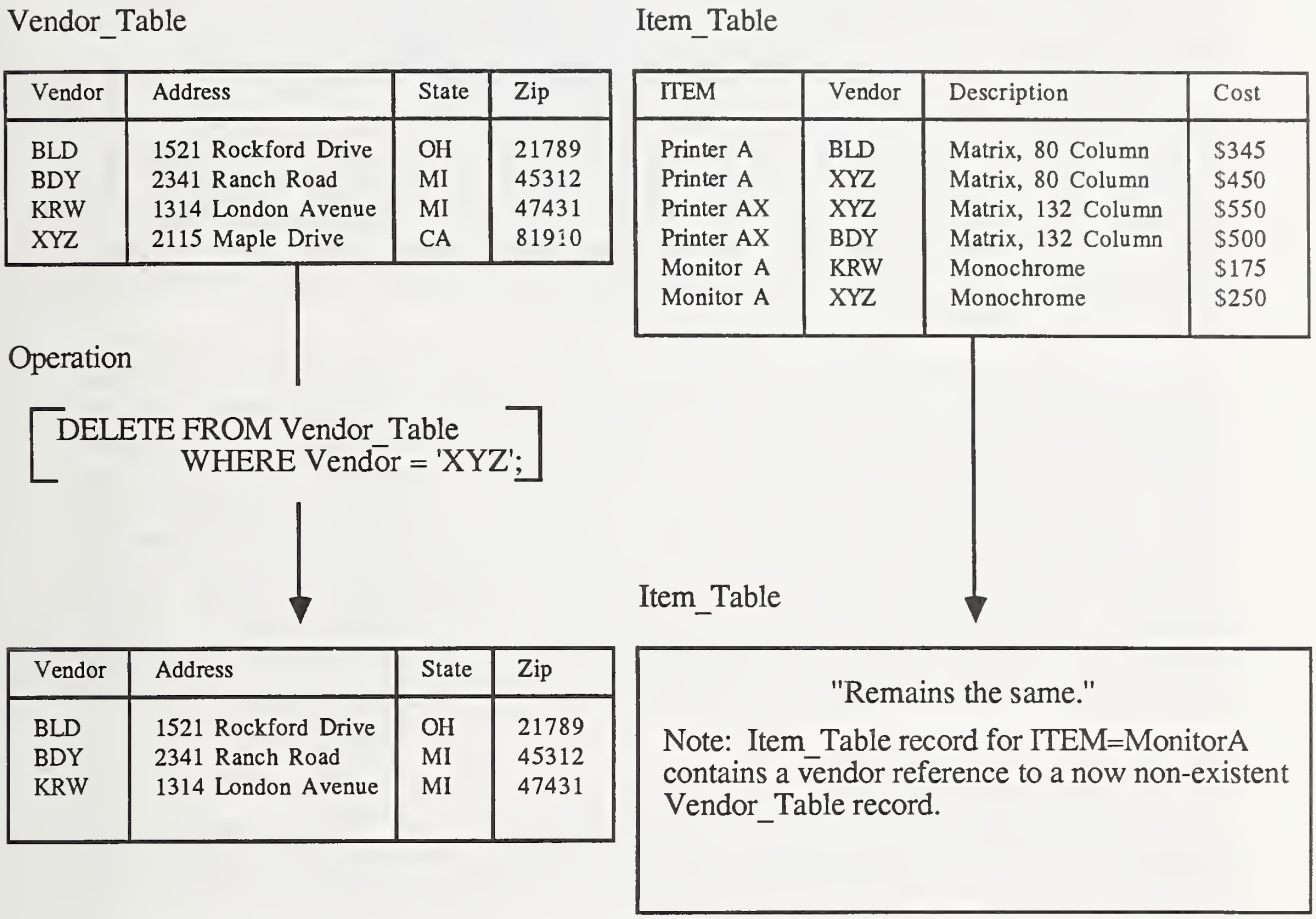
Indexing Facilities - Large database files will require some form of indexing to improve performance in record retrievals. Database files that contain one thousand or more records are generally considered to be large database files

for microcomputer data management systems. Indexing will occur automatically for some microcomputer data management systems. On such systems, when defining the data structure, specification of one of the attributes as a key is required. The system will index record occurrences by the values assigned to the key attribute. On other systems, indexing is handled by an indexing facility that is independent of the data definition facility. Thus, the name of a database file and a selected key attribute must be submitted to the indexing facility. The indexing facility uses these parameters to either update the definition file for the data structure or generate an index file. In the latter case, both the data definition file and the index file have to be open simultaneously for the proper indexing effect.

Integrity Features - Integrity features will help ensure reliable data. These features should be incorporated by the data definition facility. An example is the specification of value-acceptance constraints to the attributes defined in the definition of a data structure. This constraint assignment would serve as a filter through which unacceptable values would not be passed. Integrity checking is generally performed at the attribute, entity and referential levels. At the attribute level, assigned values are restricted to a specific domain. However, it is the integrity checking at the entity level which assures uniqueness in record occurrences. The third level of integrity checking is necessary to control anomalies when performing updates. An illustration of an update anormaly is shown in FIGURE 20. In this example, vendor XYZ was deleted from the VENDOR table; however, without any form of referrential integrity checking, reference to vendor XYZ would remain in the ITEM table.

Security Features - Data security features for microcomputer data management packages are not commonly available. This is particularly true for those data management packages that are single-user oriented, because it is assumed that equipment and data will be physically protected from unauthorized users. However, security features are beginning to be offered with some data management packages. For those packages that offer security features, the levels of security will vary from securing access to an entire database file (through passwords or encryption) to securing access to portions of a database file (i.e., records and/or attributes).

Database Structure Modification - After defining data structures, circumstances may arise that will require a reorganization. In such a case, it is desirable to have available a facility that will allow modifications to the data structure without having to reload the database. Data structure modifications are generally allowed by microcomputer data management systems; however, in many cases, an



Vendor	Address	State	Zip
BLD	1521 Rockford Drive	OH	21789
BDY	2341 Ranch Road	MI	45312
KRW	1314 London Avenue	MI	47431
XYZ	2115 Maple Drive	CA	81910

ITEM	Vendor	Description	Cost
Printer A	BLD	Matrix, 80 Column	$345
Printer A	XYZ	Matrix, 80 Column	$450
Printer AX	XYZ	Matrix, 132 Column	$550
Printer AX	BDY	Matrix, 132 Column	$500
Monitor A	KRW	Monochrome	$175
Monitor A	XYZ	Monochrome	$250

Vendor	Address	State	Zip
BLD	1521 Rockford Drive	OH	21789
BDY	2341 Ranch Road	MI	45312
KRW	1314 London Avenue	MI	47431

Figure 20: Potential Database Integrity Problem

explicit dumping of data has to be performed or it will be lost during the reorganization process.

3.2.2 Data Transporting.

A data transport facility is important for bulk loading of data and migrating data to or from another data management package. Loading data by keyboard entry one record at a time is not practical when loading large volumes of data. Bulk loading facilities are more appropriate in such instances. These facilities are supported by many microcomputer data management packages. The ASCII file format shown in FIGURE 21 is a data file format that is commonly accepted for initial bulk loading of data occurrences. In this format, attribute values are separated by commas and nonnumeric values are enclosed by double quotes.

Those packages that accept such an ASCII file form for initial loading of data occurrences usually allow data to be unloaded into the same format. This unloaded format, in some cases, is the only format available for transferring data to other data management packages. However, another format that is commonly used for transferring data among microcomputer data management packages is the Data Interchange Form (DIF). FIGURE 22 is an ASCII file example of the DIF format. This format is comprised of two sections, a header and a data section. The header section contains descriptions of the file and the data section contains the actual values.

3.2.3 Data Accessing.

The next critical evaluation factor is the level of support for data access. The following items are useful in determining the strength of the data access capability:

- command and programming languages,
- menus,
- performance controls,
- updating options, and
- report generating.

Command Language - Only a few popular microcomputer data management packages offer a command language as a major data access method. In addition, these packages generally offer a

```
"f.d.roosevelt",1882,"democrat","new york"
"ford",1913,"republican","michigan"
"l.b.johnson",1908,"democrat","texas"
"kennedy",1917,"democrat","massachusetts"
"hoover",1874,"republican","iowa"
"nixon",1913,"republican","california"
"reagan",1911,"republican","illinois"
"eisenhower",1890,"republican","kansas"
"carter",1925,"democrat","georgia"
"truman",1884,"democrat","missouri"
```

Figure 21: ASCII File for Bulk Loading a Database

```
TABLE
0,1
"U. S. Presidents"
VECTORS
0,4
""
TUPLES
0,10
""
LABEL
1,0
"President"
LABEL
2,0
"Birth_Year"
LABEL
3,0
"Party"
LABEL
4,0
"Birth_State"
DATA
0,0
""
-1,0
BOT
1,0
"f.d.roosevelt"
1,0
"1882"
1,0
"democrat"
1,0
"new york"
    .
    .
    .
    .
    .
-1,0
BOT
1,0
"truman"
1,0
"1884"
1,0
"democrat"
1,0
"missouri"
-1,0
EOD
```

Header Section

Data Element Specifications

Data Section

Figure 22: Data Interchange Format (DIF) File for Loading a Database

built-in programming-like language that allow pre-determined queries to be embedded in program structures. Some of these packages also allow embedded queries in such programming languages as BASIC, COBOL, FORTRAN, PASCAL, and C. Such programming capabilities offer the flexibility of tailoring the functions of the data management package to specific applications.

Menus - Many data management packages do not offer a command language and/or programming capability. Instead, they offer data accessing through menus. Menu-driven systems generally shift the burden of operational controls from the user to the system. However, a user does not have the flexibility that is available through command-driven systems. Thus, menu-driven data management packages can tend to bog users down in a maze of menus and prompts. Purely command-driven packages tend to assume that users know exactly what they are doing, and generally provide little in the way of guidance for new users. For these reasons, some packages offer a combination of menu and command language capabilities.

Performance Controls - Microcomputer data management packages that have performance controls enable users to obtain better response time than those that do not. The need for performance controls is of paramount importance when queries are made against large database files. For example, it takes longer to search a database file using a nonindexed field than it does to search on an indexed field. Indexing can mean the difference between 30 minutes and 3 minutes search time for large applications!

Updating Options - Throughout the life of an application, there will be times when it is necessary to make changes (updates) to the data in a database. Making these changes can be time-consuming and tedious unless adequate facilities are available. A user may desire some form of screen editing that allows searching on a specified string value or a facility that allows multiple updates through a single command.

Report Generation - A report generator is both convenient and desirable for the application user. A report generator allows a user to format output. The power of a report writer can be measured in terms of flexibility. The more flexible a report writer is, the more likely it will allow a user to tailor reports to the requirement needs of an application. Some desirable characteristics for a report writer include free-form placement of headings, titles, labels, lines of text, footnotes, and free specification of output values. Additionally, the report writer should allow specifying such common functions as totalling and

subtotalling.

3.2.4 Communication.

A communication feature is important in the case of distributed data processing. For example, if your application uses a subset of the data in a large database on a minicomputer or mainframe computer, the communication feature will allow downloading that subset to the microcomputer where operations on it can be performed at lower cost. Preferably, any structural conversion of data that is required during the download process should be invisible to the user. Thus, an ideal communication feature would merge the data management environments (mainframe and microcomputer) in such a way that it appears as if the user is working in a single environment. It should be noted, however, that such features usually require specially designed software on both the microcomputer and the mainframe. The alternative is to use microcomputer-only software which causes the microcomputer to respond like a standard terminal device to the mainframe. In this way, existing data management software on the mainframe can be used to download data to the microcomputer where it can then be reformatted for the microcomputer's data management system.

3.2.5 Documentation and Help Facilities.

An essential reference for the daily operation of any software is its documentation. Desirable documentation for microcomputer data management software includes a tutorial manual, a users manual, and a reference card [Soft84]. The tutorial manual should include examples that illustrate the functional capabilities of the software. The users manual should group commands according to their functional capabilities. The reference card should briefly summarize the actions of each command.

In addition to manuals, automated help facilities are desirable features which allow the user to inquire about the use of a command without having to refer to a bound document. An inquiry would cause a brief but informative explanation to appear on screen, perhaps with a reference to the location in the printed documentation for more details.

3.2.6 Interface Capabilities to Operating System and Hardware.

There are five areas to consider when looking at how well a data management package interfaces with the operating system of a target microcomputer. They are:

- o memory requirements,
- o keyboard characteristics,
- o operating system compatibility,
- o access to operatory system functions, and
- o multi-user processing.

Memory Requirements - The minimum amount of memory specified for a data management package may not be enough for maximum performance of that package. This may be the case for those packages that can take advantage of additional internal memory to minimize disk access time. The computing literature and the user-base for a given data mangement package are good sources for finding out whether additional memory will increase performance.

Keyboard Characteristics - A keyboard with an appropriate set of function keys is required to fully utilize the capabilities that are offered by some packages. As an example, a data management package may require a special set of function keys that are available on only certain microcomputer models. Thus, some data management packages are designed for a specific microcomputer model.

Operating System Compatibility - Since the operating system controls the interaction between the application program and the microcomputer hardware, it is of paramount importance that the data management software be compatible with the operating system of the targetted microcomputer. For example, when the data management system makes a request, the operating system translates that request into specific instructions for the hardware. Thus, a data management package running under an incompatible version of an operating system may cause extraneous errors and may fail to fully utilize all the features that are available through the operating system.

Access to Operating System Functions - The ability to execute operating system commands while running a data management package is a desirable feature. Such a feature would provide users with the convenience of listing, typing, and copying files with familiar operating system commands during the execution of the data management software. Often, these functions are available only through an interpretive set of commands that is supplied by the microcomputer data management package. That is, the syntax of of these commands is not a duplicate of the syntax that is supported

by the operating system. Thus, the user must learn a new set of utility commands. Alternatively, the package may have the ability to pass operating system commands directly to a resident command processor, thus providing full access to operating system facilities from within the package.

Multi-User Processing - Some operating systems permit multitasking and multiusers, but most 8 and 16-bit microcomputer operating systems cannot take full advantage of these features because they are limited by microprocessor speed and memory constraints. As a result, data management packages for microcomputers are generally designed for single-user environments. However, there are packages that can be tailored to either single or multi-user environments.

3.3 AVAILABLE SUPPORT

Before purchasing any software, a user should carefully consider the options available for support, including [Soft84]:

- o available training,
- o telephone assistance,
- o revisions and updates,
- o user groups, and
- o warranty.

Available Training - Classroom training for microcomputer data management users is generally available only if there is a large user base for a specific package. Instead, demonstration diskettes, tutorial documents, and help facilities are commonly used to guide the novice user.

Telephone Assistance - A call-up service is desirable for handling any problems in using a data management package. If this service is offered by the vendor of a package through a toll-free number, this would be a preferred option. However, another option is the availability of such a service through a local computer store that markets the software.

Revisions and Updates - Revisions and updates are generally provided to users at additional cost; however, in some cases, a dealer or vendor may offer a form of

maintenance contract. A maintenance contract is advisable upon expiration of the warranty period. The cost effectiveness of a maintenance contract is determined by whether a package is heavily used and whether the contract will cover the cost of all technical support, revisions, and updates.

User Groups - The popularity of a microcomputer data management package will determine the possibility of a user group. In many cases, the existence of a user group is just as important as a maintenance contract. For example, users experiencing problems with a selected package could call on other users who may have experienced a similar problem and have a ready solution. Also, members of a user group will have more of a guarantee of getting the attention of the vendor of the selected package. As a final point, a user group is a good source of reference before selecting a package and can serve as a continuing source of information through meetings and newsletters.

Warranty - Terms of the warranty should be discussed with a local dealer. It should be determined if the local dealer is authorized to sell the software. If not, the warranty that is offered by the software vendor may be violated. In some cases, dealers may offer their own warranties. In such cases, a potiential user should carefully study the particulars that are contained in the warranty and determine whether the dealer's business is well established.

3.4 EXISTING AND PROJECTED USER BASE

The type of users that will be using a system is an important issue to keep in mind when inspecting the features of a candidate microcomputer data management package. A casual user will find a menu-driven system easier to work with. A menu-driven system will carry on a dialogue with the user providing ready responses, and eliminating the need for the user to consult a manual for required procedures [Ever83]. All the user has to do at this point is select an appropriate set of options. However, in order not to stifle the growth of novice users, it may be desirable to acquire a system that allows users to increase their role as dialogue initator as their knowledge of the system increases. With such a system, the use of menus would be optional. Thus, the option of interacting through a command language should be available. This would allow querying database files without going through a menu session with the system. Additionally, a nonstifling system should allow the option of menu development. This could be accomplished through a built-in programming language feature. A programming feature would allow users to tailor menus to a specific application.

The size of a software package's existing-user base is a major determinant in the amount of support that will be available for the software. A large user base will mean that the software has been extensively tested in the field and that there is likely to be an established user group. The benefits of a user group were discussed in Section 3.3.

3.5 PRICE

Price is one of the last considerations because it is features as discussed in Section 3.2 that should be scrutinized the most. However, the relatively low cost of microcomputer data management software is a great attraction for potential users. Prices vary from as low as $25 to as high as $3500 for a first-time license. This price range is very attractive when compared with the cost of data management software for minis and mainframes which start in the tens of thousands of dollars.

Another key cost is the cost for new releases (i.e., updates, corrections, and enhancements) and support. The price of new releases ranges between $10 and $995 while support ranges between $15 and $150 per year. These prices are attractive, indeed, when compared to the thousands of dollars that are required for new releases and support for mini and mainframe data management software.

4. AN EVALUATION METHODOLOGY

The evaluation process for microcomputer software, like that for any other software, is a multi-step process. The following is a proposed sequential set of actions for evaluating microcomputer data management software:

- o assess requirements,
- o develop selection criteria,
- o perform feature analysis,
- o derive benchmark test set,
- o design benchmark tests,
- o perform benchmark testing,
- o design evaluation chart, and
- o perform evaluation.

These actions are illustrated in FIGURE 19 and further discussed in the following sections.

4.1 ASSESS REQUIREMENTS

As stated in Section 3.1, assessing the functional requirements of candidate applications will aid in identifying the category of microcomputer data management software that is most appropriate for implementing candidate applications. The assessment will be instrumental in serving as a base for developing the selection criteria.

4.2 DEVELOP SELECTION CRITERIA

The selection criteria should place emphasis on those data management features that are crucial to the success of implementing candidate applications. The selection criteria should take into consideration such desirable features as those discussed in Section 3.2. The criteria should also consider available support, and existing and

projected user bases as discussed in Sections 3.3, and 3.4, respectively.

4.3 PERFORM FEATURE ANALYSIS

After defining the selection criteria, the next step is to perform a feature analysis. This is a two-phase process. The first phase involves identifying potential packages in technical journals and computer related reference sources. The result of this effort, plus use of the selection criteria, should enable the potential purchaser to compile a list of candidate packages that come closest to satisfying the requirements of candidate applications.

An in-depth look at the features of each candidate package is the second phase of the analysis. The user manuals of each package, which normally contain detailed descriptions of the software features, should be carefully studied. A suggested means of acquiring manuals for each package is to visit a computer store or a site that has copies of the desired manuals. This course of action will help minimize cost. However, as stated in Section 3.5, the prices of microcomputer data management packages generally range between $25 and $3500. Such a low cost, as compared to the cost of data management software for minis and mainframes, offers another avenue for obtaining manuals for a candidate package. Thus, a version of each data management software for which manuals are not available could be purchased. This is practical only if the selected package for implementing candidate applications is to be purchased in large volumes. Whichever avenue is used to gain access to the manuals, the in-depth study should cover such questions as outlined in Appendix A.

4.4 DERIVE BENCHMARK TEST SET

Phase two of the feature analysis is necessary to further reduce the set of candidate packages. This reduction process can be achieved through assigning a relative weight (level of importance) and a grade (e.g., 1 to 10) for each selection criterion. The relative weights are subjective with respect to the requirements of candidate applications. The grades should be determined by the number of affirmative responses revealed on a feature critique sheet as shown in Appendix A. An acceptable score range should be specified. Each package should be rated to determine whether it falls within the specified range. Those packages that meet the specifications should be targetted for benchmark testing.

4.5 DESIGN BENCHMARK TESTS

The benchmark testing should take into consideration an appropriate set of variables that will be essential for implementing candidate applications. Each variable should be isolated as much as possible so that the effects of that variable, and only that variable, are evaluated. The nature of the single-user environment of the microcomputer lends itself nicely to this isolation process.

The benchmark design involves establishing the environment in which testing is to be performed, and developing a set of tests. The benchmark design is a four-step procedure that defines the variables associated with [Beni84]:

- o system configuration,
- o test data,
- o benchmark workload, and
- o experimental design.

System Configuration - As for any system, the major variables associated with system configuration are hardware and software. The critical hardware variables are the amounts of internal memory and external storage as well as the type of external storage medium (hard or floppy disk). Internal memory influences the degree to which swapping is performed; while, external storage constrains the size of database files and the space available for results generated by operations on the database files. The type of external medium affects access time. A hard disk can be accessed faster than a floppy disk; however, the performance of a floppy disk may be adequate for the candidate applications. The software variables include such factors as maximum record length, the number of allowable indexes per database file, the length of an indexed field, and any other access performance controls.

Test Data - The variables associated with test data include the amount of test data per database file and the amount as well as type of indexing on the database files.

Benchmark Workload - The variables associated with the benchmark workload cover both qualitative and quantitative aspects [Beni84]. Qualitative aspects refer to the level of difficulty of queries (i.e., simple retrieval on a single

database file or complex queries involving many files), the mode of operation for a given command (i.e., operates on only two files at a time), and the mode of interaction with the data management system (i.e., menus, command language or batch files). While, quantitatively, there are such factors as the percentage of time that each type of database action is performed, and the average amount of data returned to a user per transaction. Thus, if tests are designed to cover the qualitative and quantitative aspects of a data management package, test results will help determine how well the package performs under various workloads.

Experimental Design - The experimental design phase involves looking at such parameters as the candidate test packages, the size of the database files, the number of indexes, and the complexity of queries. For example, the testing should be general enough to cover a common set of features across the variety of candidate packages, yet specific enough to reveal the strength and weaknesses of each package. The database files should contain enough records to realistically reflect candidate applications. If response time to queries is critical to the applications, the indexing capabilities of each package should be thoroughly tested. Queries should vary in level of complexity in order to test the strength of a package as well as the adaptability of that package to applications. An example of a set of tests that can be used to test a variety of microcomputer data management packages is shown in Appendix B.

4.6 PERFORM BENCHMARK TESTING

Execution of the benchmark tests may require different techniques for different data management packages. That is, each candidate package will have its particular design and limitations. For this reason, the benchmark tests should be tailored to accommodate the design constraints of each package. For example, some of the packages may be able to handle selecting and sorting in a single statement while others will require separate statements (i.e., one for selecting and another to sort the selected result). An example of this is shown in FIGURE 23.

Action

Resulting Table

Employee

SELECT ALL
FROM Employee

Empno	Name	Job	Sal	Deptno
3451	Brown	Manager	2500.00	30
3454	Jenkins	Programmer	1500.00	20
3459	Dawkins	Technician	1200.00	20
5125	Smith	Researcher	2000.00	10
5145	Jones	Chemist	2400.00	40
2435	Wilkins	Clerk	800.00	30

SELECT ALL
FROM Employee
WHERE Sal > 1200;

Empno	Name	Job	Sal	Deptno
3451	Brown	Manager	2500.00	30
3454	Jenkins	Programmer	1500.00	20
5125	Smith	Researcher	2000.00	10
5145	Jones	Chemist	2400.00	40

SORT result_table
ON Sal
IN DESCENDING ORDER;

Empno	Name	Job	Sal	Deptno
3454	Jenkins	Programmer	1500.00	20
5125	Smith	Researcher	2000.00	10
5145	Jones	Chemist	2400.00	40
3451	Brown	Manager	2500.00	30

Figure 23: Sample Benchmark Tests

4.7 DESIGN EVALUATION CHART

The composite score for each candidate package from the benchmarking should be recorded as one of the entries for an evaluation chart. The design of this evaluation chart should include, in addition to the benchmark entry, entries for the weighted scores derived from phase two of the feature analysis as discussed in Section 4.4. FIGURE 24 illustrates an evaluation chart.

4.8 PERFORM THE EVALUATION

After the evaluation chart has been designed, it should be completed for each package that was benchmarked. The evaluation chart will serve as a score card for each package. The candidate package with the highest score should be the best selection.

Criteria	Weight	Candidate 1		Candidate 2		Candidate 3		Candidate 4	
		Grade	Grade X Weight	Grade	Grade X Weight	Grade	Grade X Weight	Grade	Grade X Weight
B1	.045	8	.360	5	.225	8	.360	4	.180
B2	.045	7	.315	6	.270	8	.360	4	.180
C1	.015	8	.120	7	.105	8	.120	9	.135
C2	.020	9	.180	8	.160	9	.180	9	.180
C3	.045	5	.225	8	.360	8	.360	9	.405
C4	.045	7	.315	9	.405	7	.315	9	.405
C5	.001	0	.000	0	.000	0	.000	0	.000
D1	.001	9	.009	0	.000	9	.009	0	.000
D2	.020	3	.060	8	.160	8	.160	9	.180
D3	.025	9	.225	5	.125	9	.225	9	.225
D4	.020	5	.100	0	.000	5	.100	9	.180
D5	.025	8	.200	8	.200	8	.200	9	.225
D6	.025	0	.000	8	.200	0	.000	9	.225
D7	.015	0	.000	0	.000	0	.000	9	.135
E1	.045	10	.450	5	.225	10	.450	6	.270
F1	.015	8	.120	6	.090	8	.120	0	.000
F2	.045	0	.000	9	.405	0	.000	9	.405
F3	.045	0	.000	9	.405	0	.000	9	.405
G1	.025	3	.075	8	.200	5	.125	7	.175
G2	.020	9	.180	8	.160	9	.180	6	.120
G3	.035	9	.315	8	.280	9	.315	10	.350
G4	.020	6	.120	9	.180	7	.140	10	.200
G5	.035	5	.175	7	.245	7	.245	10	.350
G6	.035	8	.280	8	.280	8	.280	10	.350
G7	.045	10	.450	5	.225	8	.360	6	.270
H1	.008	0	.000	0	.000	0	.000	8	.064
I1	.015	0	.000	0	.000	0	.000	5	.075
I2	.015	8	.120	7	.105	8	.120	8	.120
Price	.050	8	.400	7	.350	5	.250	5	.250
Testing	.200	8	1.600	8	1.600	9	1.800	9	1.800
		Scores	6.394		6.960		6.774		7.859

Grades range from 0 to 10

Figure 24: Feature Evaluation Chart

5. FUTURE DIRECTIONS AND CONSIDERATIONS

In the future, microcomputer data management software will be instrumental in realizing true distributed database management. Recent developments in building micro-mainframe links have led developments in this direction. Historically, the amount of storage space on the microcomputer has limited developing on them applications which use large data files. This limitation has kept larger databases on mainframes. However, technical developments have increased the amount of available storage on the microcomputer making it feasible to develop micro-based applications that in the past were not possible. Thus, "outside in" development of distributed systems now appears to be the general trend, rather than "inside out" (i.e., centralized on mainframes).

Several methods have been considered for linking mainframe to microcomputer database [Free84]. An initial method used software that made the microcomputer look like a terminal to the mainframe. The terminal emulator method requires the user to know how to exploit the mainframe database management system. Thus, the user has to know the access language that is used to manipulate data on the mainframe through its resident database management system. Another drawback of this technique is the fact that the user cannot easily bring information from the mainframe to the microcomputer for processing. That is, the terminal emulator method without file transfer capability is just that--"a terminal".

A second method integrates the terminal emulation and file transfer capabilities. This method offers various facilities that allow data to be formatted appropriately for target environments. Thus the user can access data from the mainframe DBMS and download the data into file format on the microcomputer where it can be manipulated. However, there is still the drawback of having to know the data access language for the mainframe DBMS.

Currently, the most advanced developments are in the area called integrated software links. This method allows the user to send queries from the microcomputer environments to the mainframe DBMS, and the response is automatically returned to the microcomputer environment in a format usable by the microcomputer DBMS. This does not allow updating of the mainframe DBMS through the link with the microcomputer. Future developments will allow this updating. Also, microcomputer data management software will become more a reflection of mainframe DBMSs. This will really make possible transparent links between microcomputer data management software and mainframe DBMSs.

6. CONCLUSIONS

Microcomputer data management software will never replace mainframe database managements systems; however, they can be complementary in many ways. A user should keep this perspective in mind when selecting microcomputer data management software. With this in mind, the following actions should be performed when selecting microcomputer data management packages:

- o assess application requirements,
- o develop selection criteria, and
- o evaluate candidate packages.

Assess Application Requirements - A clear understanding of the requirements of an application should be achieved as a first step in the selection process. Applications requiring large volumes of data should not be expected to perform as efficiently on microcomputers as on mainframes unless adequate processing power and storage are available.

Develop Selection criteria - After identifying the needs of an application, the assessment should be used as a basis for developing an appropriate set of selection criteria. The selection criteria should include desired plus needed data management features.

Evaluate Data Management Packages - A weighted grading scheme should be used to evaluate those features that are both necessary and desirable for candidate applications. The grading of features for each data management package should be performed after studying manuals and literature about a package. The subjective grading of features should narrow the scope of candidate packages and identify a benchmark test set. Thus, the final evaluation score is derived by including benchmark scores with the subjective feature scores. All the packages that score within an acceptable range are good choices for implementing candidate applications.

7. References

[Beni84] Benigni D. R., A Guide to Performance Evaluation of Database Systems, NBS Special Publication 500-118, National Bureau of Standards, December, 1984.

[Data84] A Buyer's Guide to Data Base Management Systems, Datapro (Minicomputers), Datapro Research Corporation, Delran, NJ 08075, May 1984, page 70E-010-61a.

[Date77] Date C. J., An Introduction to Database Systems, Addison-Wesley Publishing Company, Reading MA, 1977.

[Ever83] Everst G. C., Database Management Systems for Microcomputers: Are They Surpassing DBMSs for Mainframe Computers, School of Management, University of Minnesota, November 1983.

[Fips110] Guideline for Choosing a Data Management Approach, Federal Information Processing Standards (FIPS) Publication 110, National Bureau of Standards, December 1984.

[Free84] Freedman D. H., Tapping the Corporate Database, High Technology, April 1984, pages 26-29.

[Gall84] Gallagher L. J. and Draper J. M., Guide on Data Models in the Selection and Use of Database Management Systems, NBS Special Publication 500-108, National Bureau of Standards, January, 1984, pages 51-59.

[Gutt84] Guttman M. K., Micro Data Managers Get Mainframe Power, Computers & Electronics, October 1984, page 67.

[Jame84] James B., Network vs. Relational: Fit the Solution to the Need, Computerworld, September 24, 1984, page SR/38.

[Soft84] Software Evaluations: A Discussion of the Methodology used for Microcomputer Software Evaluation, Data Decisions, Filing Sequence 805, February 1984.

[Tsic77] Tsichritzis D. C. and Lochovsky F. H., Data Base Management Systems, Academic Press, Inc., 111 Fifth Avenue, New York, New York 10003, 1977.

APPENDIX A: Example of a Feature Critique Sheet

A. General Information

A1 - Package name: _____

B. Physical Requirements

B1 - Internal memory: _____ K bytes
B2 - External memory: _____ K bytes

C. Logical Constraints

C1 - Maximum characters per attribute (field): _____
C2 - Maximum fields per record: _____
C3 - Maximum records per database file: _____
C4 - Maximum database files per database: _____
C5 - Database files (tables) are assigned to designated area: _____ (yes or no)

D. Data Definition

D1 - Defined through screen forms. _____ (yes or no)
D2 - Screen editing is available while defining data definitions. _____ (yes or no)
D3 - The set of available data types. (check all)
number: ____
character: ____
logical: ____
text: ____
date: ____
dollar: ____
memo: ____
Other: ____ (specify.) _____
D4 - A forms design facility is available. ____ (yes or no)
D5 - Indexing can be specified. ____ (yes or no)
If yes, indicate maximum number of fields allowed: _____
Is indexing optional or manditory: _____
Check any of the following performance controls (in addition to indexing) that are available.
clustering: ____
triggers: ____
accelerators: ____
Other: ____ (specify) _____
D6 - Is integrity checking of data values available: ____ (yes or no)
If yes, check all of the following that apply.
range checking: ____

list checking: ____
table lookup: ____
Other: ____ (specify) _____

D7 - Is security specifications available: ____ (yes or no)
If yes, check all of the following that apply.
at the database level (passwords): ____
at the table (database file) level: ____
at the record level: ____
at the attribute (field) level: ____
Other: ____ (specify) _____

E. Data Reorganization

E1 - Is data reorganization possible: ____ (yes or no)
If yes, check all of the following that apply.
Data must be unloaded (by the user) to a file before reorganizing data definitions: ____
Data is unloaded (by the system) to a temporary file after reorganization is specified by the user and then the data is loaded (by the system) to the modified definition: ____
Other: ____ (specify)_____
_____
_____
_____

F. Data Loading and Unloading

F1 - Indicate which of the following features is available upon keyboard entry.
Screen editing over the entire set of record attribute values: ____
Single attribute prompting without screen editing capabilities: ____
Other: ____ (specify)_____
_____

F2 - Can data be loaded (read) from a data file: ____ (yes or no)
If yes, indicate all acceptable forms of data files.
DIF: ____
An ASCII file with attribute values separated according to specification: ____
Binary form: ____
Other: ____ (specify)_____

F3 - Can data be unloaded (written) to a file: ____ (yes or no)
If yes, indicate all available forms that data can be written in.
DIF: ____
An ASCII file with attribute values separated according to specification: ____
Binary form: ____
Other: ____ (specify)_____

G. Data Accessing

G1 - Indicate the methods availabe for conversing with the system. (Check all)
Through a command language: ____
Through menus: ____
Through a built-in programming-like language: ____
Through a programming language: ____
Through user-defined screens: ____
Other: ____ (specify)_____

G2 - Indicate the methods available for updating. (Check all)
Through a form of screen editing (a single record at a time): ____
Through a form of screen editing (over several records at a time): ____
Through a non-screen edit mood (using some form of query capability that allow attribute value replacement): ____
Other: ____ (specify)_____

G3 - Indicate the logic operators available for searching. (Check all)
AND: ____
OR: ____
NOT: ____
Other: ____ (specify)_____

G4 - Indicate the operators that can operate simultaneously on multiple tables. (Check all)
JOIN: ____
MATCH: ____
COMPARE: ____
UNION: ____
INTERSECT: ____
Other: ____ (specify)_____

G5 - Indicate the features available for conditional searching. (Check all)
Wild cards are allowed: ____
A non-indexed field can be used as the search key: ____
Searches can be performed on any data type: ____
Other: ____ (specify) _____

G6 - Can sorts be performed: ____ (yes or no)
If yes, indicate the maximum number of fields that can be sorted simultaneously: _____

G7 - Indicate the features available for report generating. (Check all)
Free form placement of headings: ____
Free form placement of titles: ____
Supports footnoting: ____
Supports totalling: ____
Supports subtotals: ____
Supports formatting of output values: ____
Supports page breaks: ____
Other: ____ (specify)_____

H. Communication

H1 - Does the system allow communicating with another computing environment: _____ (yes or no)

I. Documentation and Help Facilities

I1 - Are help facilities available: _____ (yes or no)
If yes, can help be requestable from any functional level: _____ (yes or no)

I2 - Indicate the availability of manuals and documentation. (Check all)
Tutorial: ____
User: ____
Reference card:____
Other:____ (specify)_____

APPENDIX B: Example of Benchmark Tests

Searching a single table

```
T1-1  select all
          from agencyd;

T1-2  select all
          from agencyd
          where agency='BD';

T1-3  select all
          from agencyd
          where agency='BD' or agency='AF';

T1-4  select all
          from agencyd
          where (agency='BD' or agency='AF')
          and subelemt='07';

T1-5  select all
          from agency_desc
          where ((agency='BD' or agency='AF')
          and subelemt='07')
          or subelemt='24';
```

Selecting and Projecting through a single statement

```
T2-1  select agency,subelemt
          from agencyd;

T2-2  select agency,subelemt
          from agencyd
          where agency='BD';

T2-3  select agency,subelemt
          from agencyd
          wherec agency='BD' or agency='AF';

T2-4  select agency,subelemt
          from agencyd
          where (agency='BD' or agency='AF')
          and subelemt='07';

T2-5  select agency,subelemt
          from agency_desc
          where ((agency='BD' or agency='AF')
          and subelemt='07'))
          or subelemt='24';
```

Sorting a table

```
T3-1  sort jobhist by servdate

T3-2  sort jobhist by jhssn,servdate
```

Loading a table

```
T4  load jobhist from jobhist.dat
```

Unloading a table

```
T5  unload jobhist to jobhist.dat
```

Updating a record

```
T6  update acadisc=0507
           in educat
           where essn=201235532
```

Deleting a record

```
T7  delete educat
           where essn=201235532
```

Inserting a record

```
T8  insert record
           into educat
           where essn=201235532
                 educlvl=13
                 degrdate=78
                 acadisc=0506
```

Joining two tables

```
T9-1  join educat,jobhist
          where essn=jhssn;
```

```
T9-2  join educat,jobhist
          where essn=jhssn
             and acadisc='0506';

T9-3  join educat,jobhist
          where essn=jhssn
          and (acadisc='0506' or acadisc='0101');

T9-4  join educat,jobhist
          where essn=jhssn
          and (acadisc='0506' or acadisc='0101')
          and servdate>780101

T9-5  join educat,jobhist
          where essn=jhssn
          and (((acadisc='0506' or acadisc='0101')
          and servdate>780101
          and educlvl>21)
          or state='31')
```

Joining and Projecting through a single statement

```
T10-1  select essn,jhagency,educlvl
          from educat,jobhist
          where essn=jhssn;

T10-2  select essn,jhagency,educlvl
          from educat,jobhist
          where essn=jhssn
             and acadisc='0506';

T10-3  select essn,jhagency,educlvl
          from educat,jobhist
          where essn=jhssn
          and (acadisc='0506' or acadisc='0101');

T10-4  select essn,jhagency,educlvl
          from educat,jobhist
          where essn=jhssn
          and (acadisc='0506' or acadisc='0101')
          and servdate>780101

T10-5  select essn,jhagency,educlvl
          from educat,jobhist
          where essn=jhssn
          and (((acadisc='0506' or acadisc='0101')
          and servdate>780101
          and educlvl>21)
          or state='31')
```

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBS/SP-500/131	2. Performing Organ. Report No.	3. Publication Date October 1985

4. TITLE AND SUBTITLE

Computer Science and Technology:
Guide for Selecting Microcomputer Data Management Software

5. AUTHOR(S)

Charles L. Sheppard

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234
Gaithersburg, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Same as item 6.

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 85-600598

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

The purpose of this guide is to provide information that will aid in developing criteria that can be used to select the microcomputer data management software that is most appropriate for a specific database application. Emphasis is placed on identifying the spectrum of data management software for microcomputers and the tasks involved in using a set of criteria to evaluate and select microcomputer data management software.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

application; assessment; benchmark; evaluation; microcomputers; multi-file; selection; single-file

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

66

15. Price

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NBS Technical Publications

Periodical

Journal of Research—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396).
NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.
Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*
Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Bureau of Standards
Gaithersburg, MD 20899