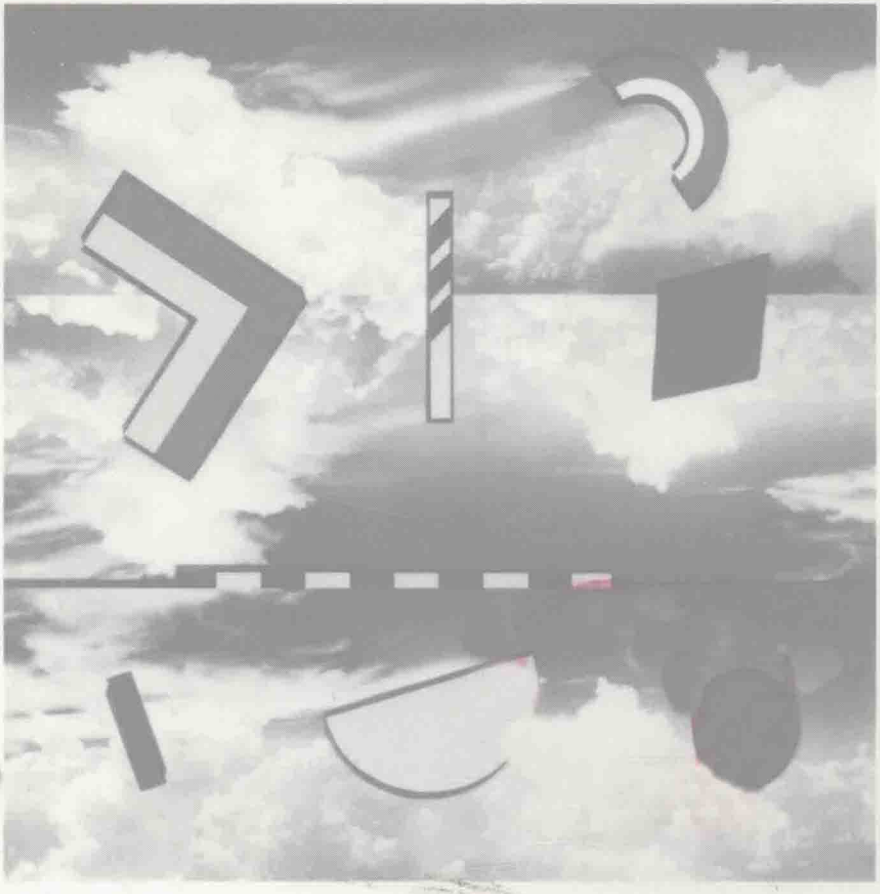
Second Edition

INTRODUCTION TO THE MICROCOMPUTER AND ITS APPLICATIONS

dBASE



CHAO C. CHIEN

PC Version

INTRODUCTION TO THE MICROCOMPUTER AND ITS APPLICATIONS: dBASE®

We recognize that certain terms in the book are trademarks, and we have made every effort to print these throughout the text with the capitalization and punctuation used by the holder of the trademark.

WordStar® is a registered trademark of MicroPro International Corp.
PC-DOS™ is a trademark of IBM Corporation.
WordPerfect® is a registered trademark of WordPerfect Corporation.
Lotus® and 1-2-3® are registered trademarks of Lotus Development Corporation.
dBASE® is a registered trademark of Ashton-Tate.

Richard D. Irwin, Inc., makes no warranties, either expressed or implied, regarding the enclosed computer software package, its merchantability or its fitness for any particular purpose. The exclusion of implied warranties is not permitted by some states. The exclusion may not apply to you. This warranty also provides you with specific legal rights. There may be other rights that you may have which may vary from state to state.

Senior sponsoring editor: Lawrence E. Alexander
Developmental editor: Rebecca J. Johnson
Editorial assistant: Lena Buonanno
Project editor: Joan A. Hopkins
Production manager: Carma W. Fazio
Designer: Keith J. McPherson
Artist: Tom Mallon
Cover photo/illustrator: Emily Medvel
Compositor: Graphic Typesetting Service
Typeface: 10/12 Melior
Printer: Patterson Printing

ISBN 0-256-08820-9 (PC version)
ISBN 0-256-08475-0 (PS/2 version)

Printed in the United States of America

1 2 3 4 5 6 7 8 9 0 P 7 6 5 4 3 2 1 0

To
Mother
With love from us all

INTRODUCTION

This booklet introduces you to the subject of database management and the use of the microcomputer data management program dBASE®. Because of the depth and scope of the subject matter (database management normally is studied in the junior or senior year of a four-year college program), this booklet discusses only the fundamental database concepts. The bulk of the coverage is devoted to using the dBASE programs for file data manipulation.

The dBASE program selected for use is dBASE III Plus (or any compatible program), the most widely used microcomputer data management program version and the basis for many compatible products and language compilers. Although dBASE IV, Version 1.0 is now available, it has not gained wide acceptance by users for both technical and marketing reasons. However, because most of the basic commands are the same in the two versions, you may use this booklet with either program. Nevertheless, the discussions in this booklet are primarily dBASE III Plus oriented, and dBASE IV is mentioned only when necessary.

Because the computer system we will use to run dBASE is the IBM PC and the operating system is PC-DOS, it is assumed that you are already familiar with the basic PC-DOS commands used to manipulate files and work with disk directories. If you are not, you should become familiar with them by reading *Introduction to the Microcomputer and Its Applications* or the separate booklet on PC-DOS, both published by Richard D. Irwin, Inc.

To aid your study of the subject material, a data diskette for your use accompanies this booklet. Because this diskette also comes with *Introduction to the Microcomputer and Its Applications*, you will find that it contains many more files than you need. Ignore the extra files, and use the example and illustration files to perform the exercises as suggested in the text.

INSTRUCTIONS FOR PREPARING THE $5\frac{1}{4}$-INCH DATA DISKETTE

The data diskette has the following directory structure:

\BATCH	For use with PC-DOS and batch files
\WP-FILES	For use with WordPerfect
\WP-LTD	For use with WordPerfect Limited Edition

\WS-FILES	For use with WordStar
\123FILES	For use with 1-2-3
\DB3FILES	For use with dBASE III Plus/IV or the Educational Version
\ADDRESS	For use with dBASE III Plus/IV or the Educational Version
\WORKFILE	Extra work files for practice or exam use
\WORDPERF	For use with WordPerfect
\WP-LTD	For use with WordPerfect Limited Edition
\WORDSTAR	For use with WordStar
\LOTUS123	For use with 1-2-3
\DBASE	For use with dBASE III Plus/IV
\DBASE-ED	For use with dBASE III Plus Educational Version

The $5\frac{1}{4}$-inch diskette is entirely occupied by the files in these directories. To use the diskette, you should first create a working diskette by disk copying the data diskette onto a blank diskette (using the PC-DOS command DISKCOPY A: A:) and then making room on it by removing the unnecessary files and directories. For this booklet, the directories required are \DB3FILES, \ADDRESS, and \WORKFILE\DBASE (or \WORKFILE\DBASE-ED for Educational Version users).

To remove a directory, you must first erase its files. For example, to remove directory \123FILES, use the following PC-DOS commands:

```
ERASE A:\123FILES\*.*
RD A:\123FILES
```

To remove the \WORKFILE subdirectory, first erase all of its files. For example, to remove the \WORKFILE\WORDSTAR subdirectory, use the following PC-DOS command:

```
ERASE A:\WORKFILE\WORDSTAR\*.*
RD A:\WORKFILE\WORDSTAR
```

You are now ready to begin working.

CONTENTS

Chapter 1

Introduction to Database Management

After reading this chapter, you should be able to:

- Describe the database file structure
- Define a database
- Identify three well-known database structures
- Set up and create a dBASE data file
- Perform dBASE data entry and editing operations
- Use the dBASE data review commands

A database is a collection of data prepared for the purpose of producing information. For the microcomputer, it can be a file or a set of related files. Because the maintenance and processing of database data are complicated and demanding tasks, they require special computer programs. These computer programs are known as **data file managers.** The dBASE III Plus and dBASE IV, Release 1.0 programs produced by Ashton-Tate are data file managers designed for use on the IBM PC.

Although dBASE IV creates and works on data files with structures totally different from those of its predecessor, dBASE III Plus, and offers significantly more data processing features and commands, it is capable of working on dBASE III Plus data files directly. In other words, dBASE III Plus data can be used with dBASE IV. However, dBASE IV–created data are not acceptable by dBASE III Plus, a characteristic known as **upward compatibility.** However, because this booklet gives basically the same topical coverage of dBASE III Plus and dBASE IV—with the exception of Chapter 4, which is dedicated to the study of the dBASE III Plus assist mode of operations—either program can be used with it. Therefore, unless explicitly stated otherwise, when we use the term *dBASE* we will mean either dBASE III Plus or dBASE IV.

THE CONCEPT OF A STRUCTURED DATA FILE

The unique characteristic of a database file is that its data are *structured*. This means that the file data consist of identifiable units with specific properties. This is because in data processing many operations, such as ordering the data, cannot be performed unless the data are structured. Therefore, before we begin studying data management operations, we will examine the concept of structured data. Figure 1.1 presents a sample structured data file.

FIGURE 1 ■ 1 A music data file

COMPOSER	TITLE	ORCHESTRA	CONDUCTOR
Beethoven	Symphony No. 3	N.Y. Philharmonic	Leonard Bernstein
Beethoven	Symphony No. 5	L.A. Philharmonic	Carlo M. Giulini
Beethoven	Symphony No. 7	New Phil. Orch.	Otto Klemperer
Brahms	Symphony No. 1	Berlin Philharmonic	von Karajan
Brahms	Symphony No. 4	Philadelphia Orch.	Ricardo Muti
Tchaikovsky	Symphony No. 6	Leningrad Phil.	Mravinsky

FIGURE 1 ■ 2
A student grade records data file

STUDENT	S.S. NUMBER	GRADES		
Johnson, Connie	123-66-4538	90	90	95
Jackson, David	453-67-9568	85	80	70
Krieg, Benny	343-77-7454	60	55	70
Larson, Hans	112-87-3429	40	30	55
Masters, Vinve	440-33-8766	85	85	80

The Record

The distinctive characteristic of a set of well-structured data is uniformity. For instance, an instructor keeps a well-defined collection of data about the students in a class. The data collection consists of a number of entries that pertain to each student. Each entry, in turn, consists of set items, such as name, social security number, and test grades (see Figure 1.2). Most important, all entries have the same composition, and none receives preference; in other words, all have the same *ranking*. Such a collection of data is called a *data file* or, more precisely, a **structured data file**, and its components are called **records**. Thus, the instructor maintains a student data file made up of many student data records.

The Field

While a record is a file entry—a finite collection of similar data—it has a definitive composition. The components of a record are called **fields**. Thus, for the student records in Figure 1.2, the name, social security number, and test grades are the data fields, with each record having the same composition. The field, therefore, is the smallest unit of defined file data.

In electronic data processing, a data file is saved in storage and consists of records, which in turn are composed of fields. Fields, of course, are made up of basic characters, or bytes.

THE CONCEPT OF A DATABASE

The instructor in our example may also maintain a separate data file containing the same students' homework grades. Perhaps the homework grades will be averaged and used with the test grades from the other data file to compute the final grades. Therefore, the instructor maintains a set of related files that comprise a **database**—a central

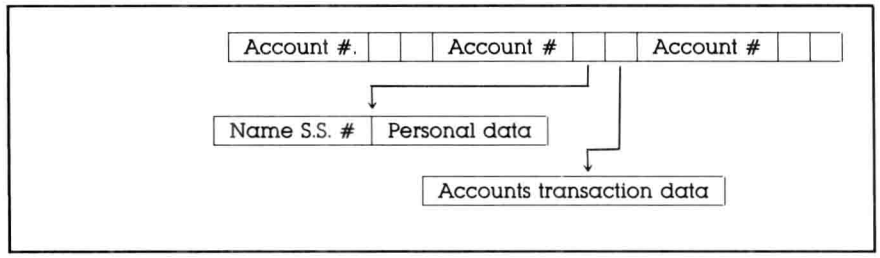


FIGURE 1 ■ 3
A hierarchy database

collection of related data created for specific purposes. Depending on how the data held in the various files will be used, they can be set up according to specific arrangements.

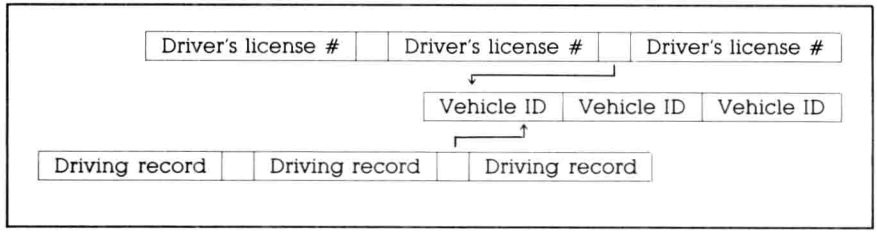
Hierarchy Database

Keeping large amounts of data in the same file can be cumbersome, especially when there are many data fields to maintain. Usually only a portion of the data set is used at any one time, that is, for a particular application. Thus, data can be segregated into groups but related so that the user can move from one set to another if necessary.

For instance, suppose a bank customer wants to review his checking account data at one time and his loan history at another. The two sets of data can be maintained separately but identified and accessed through the customer's account number. Thus, the account number serves as the customer's reference for the various data files.

The account numbers and data references of the bank customer data—technically called **pointers**—can be collected into one file and used as the entry point to the account data, which in turn may contain pointers to refer to other, more detailed data. This data organization forms a **hierarchy database**; that is, data stored in various files are made accessible through a set referral system (see Figure 1.3). In other words, a hierarchy database possesses a data-ranking structure: The lower-ranking file data are accessible only through references from the higher-ranking files. In our bank example, the customer's account number will lead to the pertinent data records, but neither the data in the checking account file nor the data in the loan history file will trace back to the account number. The files through which the lower-ranking files are accessed are called the **parents**, and the lower-ranking files are called the **children**. In a hierarchy database, a parent may have more than one child, but a child cannot have more than one parent.

FIGURE 1 ■ 4
A simple network database



Network Database

To facilitate tracing of related data, the various files can be arranged such that the data in one file will lead to related data in another file, and vice versa. This data structure is called a **network database** (here the word *network* does not refer to telecommunication). In a network database, the referral data (pointers) are incorporated into the various file records so that they can lead to other, related data records (see Figure 1.4). Therefore, in a simple network database a child may have multiple parents. But in a complex network database there is no defined parent-child relationship, since each file can be the parent and child of another file simultaneously.

Relational Database

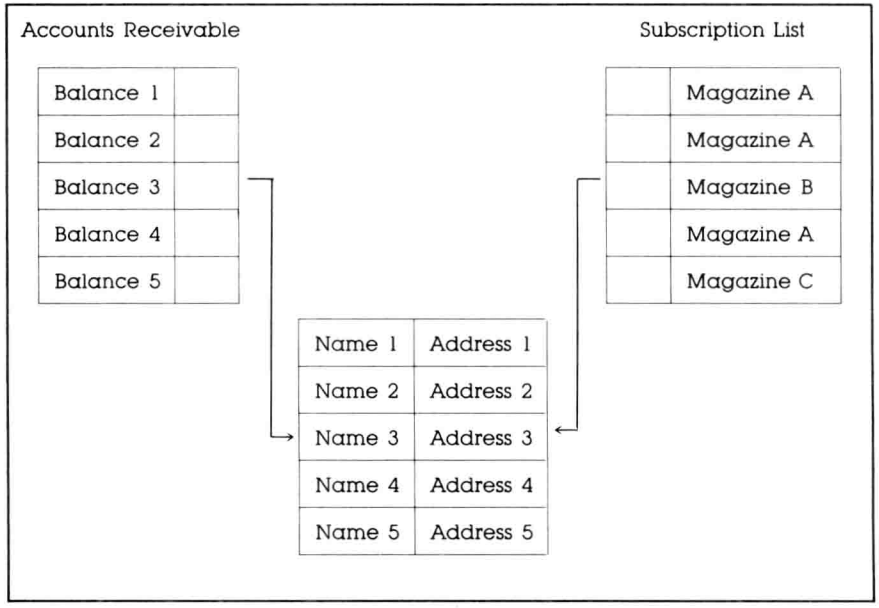
In a **relational database** (see Figure 1.5), the related data are constructed into a matrix table when they are used, which facilitates their processing. Due to the tremendous flexibility such data relationship tables offer, the data can be related in different ways to suit the needs of specific applications. The relational database structure is the most recent approach to database design and is favored by many large computer applications. Its construction, however, is necessarily complex.

DATA FILE STRUCTURE

Except for the broad databases maintained by the large computers, such as the mainframe computer, which may use an entire hard disk or even several for storing database data, practically all database structures begin with files. We will now discuss the essence of a **data file structure**.

It is important to understand that in computer operations all items stored as a group are called *files*, such as program files and data files. The structure of such files consists of the disk's tracks and sectors. These divisions are called the **physical file structures**. The data file struc-

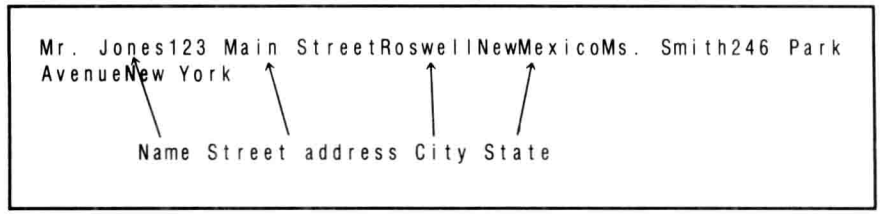
FIGURE 1 ■ 5
A relational database



tures with which we are concerned, however, are not the physical structures of the stored files but the **logical file structures**.

A **database file** is made up of uniform fields and records. For example, if a field is defined as numeric, all the records in the same file will contain the same kind of numeric data. Likewise, if a field is defined as alphanumeric, all the file records will contain the same type of alphanumeric data. In particular, if a **fixed-length record format** is used, all the records will have the same number of characters. The records and fields are defined with set lengths to facilitate future data changing and updating.

Consider the following stored data arrangement:



Now suppose Mr. Jones has moved. His new address is 999 Country Club Road, Roswell, New Mexico. If the new address is written to the file, the data will look like this:

```
Mr. Jones999 Country Club RoadRoswellNew Mexicoth246 Park
AvenueNew York
```

Ms. Smith's data have been written over, because Mr. Jones's new address is longer than the previous one. To preserve Ms. Smith's data—and those for other names in the file—the entire file must be rewritten to make room for Mr. Jones. However, rewriting a file each time you change a single data item is inefficient. One way to circumvent this is to define all the data records as occupying the same storage space. That way, any data changes will fit into the original slots without your having to rewrite the entire file. The ability to retrieve and store any data record without disturbing other data records in the same file is known as *direct accessing* or *random accessing*. A data file that uses the fixed-length record format is called a **direct-access file** or **random-access file.** dBASE data files are examples of such files. Therefore, before you can enter data into a file for later use, you must define the file structure by stating:

1. How many fields each record will contain.
2. What **data type** they are.
3. How many characters (called the **field length** or **field width**) each field will be allowed to hold.

This file data definition process is called *structuring a file*. After the file structure is defined, the database manager will know what the data will look like and therefore how to put them in the file.

To illustrate how to define a file structure using dBASE, we will design a data file for maintaining a set of product data for a retail store. Each record in this file will have nine fields: part number, part description, per-item cost, supplier, address, telephone number, date last ordered, a field indicating whether the part needs reordering, and a field for entering remarks. We will now start up dBASE and create this file.

STARTING dBASE

There are several ways to start up the dBASE program. In all cases, the process involves two directories: the directory in which dBASE resides and the directory in which the data files will be stored. The two directories, however, may be the same.