Osborne McGraw-Hill

DATA BASE MANAGEMENT SYSTEMS

MS-DOS®: EVALUATING MS-DOS® DATA BASE SOFTWARE

David Kruglinski

Data Base Management Systems MS-DOS®: Evaluating MS-DOS Data Base Software

David Kruglinski

Osborne McGraw-Hill
Berkeley, California

Published by
Osborne **McGraw-Hill**
2600 Tenth Street
Berkeley, California 94710
U.S.A.

For information on translations and book distributors outside of the U.S.A., please write to Osborne **McGraw-Hill** at the above address.

A complete list of trademarks appears on page 457.

**Data Base
Management Systems**

1234567890 DODO 898765

ISBN 0-07-881180-5

Lorraine Aochi, Technical Editor
Cheryl Creager, Composition
Yashi Okita, Cover Design
Cynthia Hudson, Acquisitions Editor
Ellen Silge, Copy Editor

Acknowledgments

Special thanks go to those software publishers who made copies of their packages available for evaluation and who were patient enough to answer this first-time user's questions. Particularly helpful were Mike Nourse of Cosmos and David Kroenke of Microrim. Others contributed sample data for the examples: Phil Fawcett provided the C.I.P show dogs; Jim Buss, the R:base steel certificates; Pam Cleveland, the DataFlex insurance forms; and William Clise, the PC/FOCUS commodity data. At Osborne/McGraw-Hill, the main contributors were Cindy Hudson, Jon Erickson, and Lorraine Aochi.

About The Author

Since writing the original Osborne/McGraw-Hill *Data Base Management Systems* book in 1982, David Kruglinski has produced *The Osborne/McGraw-Hill Guide to PC Communications* and *The Framework Book.* He is currently a sales consultant at the Entré Computer Center in Bellevue, Washington, where he specializes in data base, accounting, and project management software.

Preface

This book is the successor to *Data Base Management Systems: A Guide to Microcomputer Software*, published by Osborne/McGraw-Hill in late 1982. Much has changed in two years. The IBM PC has replaced the eight-bit CP/M system as the principal business microcomputer, and the number of data base products has grown from half a dozen to over two hundred.

The earlier book's scope was limited by the packages that were available then. There was no hierarchical data base for microcomputers, which meant that there was no hierarchical section in the previous book. The dominant relational package used the term "database" in a special way, and it was adopted as a technical term. Now that some of the best mainframe DBMS packages are available for the PC, there is no longer a need for special accommodations. Mainframe terminology is used to describe both high-performance DBMSs and entry-level file managers.

Readers of the earlier book will find some familiar DBMS packages with a few of the old examples included here, but they will also find new material. Rather than try to be an encyclopedia of DBMS products, this book attempts to classify packages, emphasizing a few strong products in each class. The products are chosen because of their quality and popularity or because of some unique feature.

This book includes a chapter on single-file data managers because of their popularity and because they are a good starting point both for understanding data base theory and for organizing data in the office or home. The term FMS (file management system) now applies to single-file packages only; multifile packages have been taken in under the RDBMS (relational data base management system) umbrella.

The objective of this book is the same as its predecessor's: to help the reader decide which data base management product to buy. A thorough but easily comprehensible explanation of data base theory accompanies descriptions of available packages and useful data management examples.

Remember that the microcomputer software world is in a constant state of change. Publishers often add new features to their products, and they sometimes change prices abruptly, usually downward. Be sure to check prices and specifications with your dealer before deciding on a DBMS package.

Contents

1 Introduction To Data Base Software

When microcomputers were first introduced and sold, their owners were expected to program them in BASIC. Then came the WordStar word-processing program and the VisiCalc spreadsheet, and it was obvious that easy-to-use office productivity software would propel the micro to new levels of popularity. Consumers wanted to do useful work with their micros without long hours of programming, and they bought hardware as soon as the necessary software packages were available.

Word processors and spreadsheets are just two of the many categories of microcomputer software that you will see in computer stores or mail-order catalogs. Other major software classifications are accounting, business graphics, project management, communications, computer-aided design (CAD), and *data base management.* Data base software, the subject of this book, enables you to use your microcomputer for record-keeping tasks the same way data-processing professionals use their large mainframe computers. You can keep track of your own business clients the same way that General Motors keeps track of its customers.

There is more variation in data base software than there is in other software categories. Any office word processor will let you write a letter, and any spreadsheet will let you prepare an ordinary income statement. But as you will see, data base software can be as basic as a simple address list or as complex as a complete information system that can automate an entire company. The address list can be set up by office staff in a few minutes, but the more complicated system may require months of development by skilled professionals. The software package that handles the first task may not be powerful enough for the second, and the package that handles the second may be too unwieldy for the first. Choosing data base management software is no easy task because the choice often depends on the job at hand and the personnel available.

A SHORT HISTORY OF DATA BASE DEVELOPMENT

Programmers were writing business software back in the fifties, often using a language called COBOL. Disk drives had not been invented, and 8K (8,192 characters) of memory was considered large. The jobs of business software were big; banks and insurance companies were the first customers. Most systems were *batch-oriented,* requiring data to be punched on tab cards. Batches of cards were fed into the computer so that reports, statements, and invoices could be printed.

As managers warmed up to *data processing* (DP), they demanded more from their systems. IBM's introduction of the 360 computer series made it possible to run two or more jobs simultaneously. Soon video terminals were attached, and *on-line* software was developed. Thus, company staff could enter and retrieve data directly without using punched cards. This put heavy demands on programmers because existing software techniques were geared to batch processing. There was a need for quick response from data stored in complex structures.

Computer manufacturers—IBM in particular—came up with the *data base management system* (DBMS) as a solution to DP problems. The idea was to collect company data into a central information bank accessible to everyone in the company, subject to security requirements. The data was called the *data base,* and the program that manipulated the data was the DBMS. It was a completely new programming methodology.

Since it was unimaginable at the time for ordinary people to design their own computer applications, all DBMS software was oriented to programmers and DP managers. The early DBMS was a computer program that talked only with other computer programs, some of which supervised video terminals. The DBMS was an *intermediary* between an application module, such as an accounts-receivable cash-entry program, and the data base on disk. Using a DBMS created a more organized and tightly integrated system that required less programming and testing than those systems using earlier software technologies.

In the late sixties, programmers attempted to extend the DBMS into a *management information system* (MIS). In an MIS, all of the company's data was put into a vast pool where executives could go fishing for any information they wanted. The information would, of course, instantly reflect any changes resulting from payrolls, sales invoicing, and other business activities. This required a *query processor* to enable executives to ask questions in a computer language as close as possible to English. As it also required extraordinary software expertise and managers who could type, MIS was put on the back burner and classified as an ultimate goal. In the meantime, computer professionals began to apply the data base concept to specific accounting and record-keeping operations.

If you scan the want ads under "data processing," you will notice many jobs for people with CICS and IMS skills. CICS and IMS are acronyms for popular IBM mainframe data base products that interface with COBOL and video terminals. That is the state of the art today in the mainframes, but, surprisingly, the level of sophistication is not as high as in many microcomputer products. It is inertia—the need to protect 20 years' worth of programming investments—that restricts mainframe software development.

CICS and IMS, along with TOTAL, IDMS, ADABAS, and other products from independent software developers, support the bread-and-butter *record-keeping* functions of large corporations. Payroll, order-processing, and inventory applications are usually programmed in COBOL with the help of the DBMS. The emphasis is on on-line access. You place an order by phone, the order taker keys it into the terminal, the shipping papers are produced, and the invoice is printed and mailed.

There is another distinct category of mainframe DBMS, however, designed to meet the corporation's analytical needs. CICS stores and retrieves orders well, but it was not designed for answering a question like, "How many cases of walnuts were shipped to Utah during the first four months of 1985?" NOMAD, RAMIS, and FOCUS are among the leading analytical DBMSs. These products allow computer professionals to set up the data on disk, and then others in the company can make queries using a *fourth-generation language* (4GL). Because these DBMSs are not optimized for on-line transactions, data is often loaded into them from a separate CICS-based on-line system or from an outside source.

While DBMSs were becoming entrenched in the mainframe world, the microcomputer revolution was beginning. In the late seventies, small, inexpensive microcomputers like the Apple II and the TRS-80 appeared and were snapped up by office managers. Software developers quickly produced a number of simple, inexpensive electronic filing systems. CP/M became the dominant business operating system, and despite a severe (by today's standards) memory limitation of 64K, some surprisingly capable DBMS packages were developed for those machines. dBASE II was the best-known product in this class.

As new mainframe accounting systems were being based on DBMSs, so were some microcomputer accounting systems. This trend will continue, even though the development process is slow. Because personal-computer software developers can amortize development costs over a large number of units, they can afford to include many features in their programs, making the packages more flexible and easy to use than any custom mainframe program. The DBMS is a natural foundation for those sophisticated accounting systems.

The IBM PC has attracted software created on both eight-bit micros and mainframes. The popularity of this machine and its clones has motivated major software developers to convert their products. Start-up firms have also contributed new products designed with the IBM PC in mind. DBMS programs are in the forefront of IBM PC software, but the new *integrated* programs with strong data management functions should not be ignored.

THE DEFINITION OF A DBMS

What does an IBM 4300 DBMS have in common with an Apple II data management program? How is a mainframe management-information system similar to the Christmas card mailing list on your micro? In all cases, the DBMS is an intermediary between a person and the data in the computer. A person requests and receives information, and that information is stored on disk in an organized way. "List all my friends with birthdays in June" is the equivalent of "List all my customers with overdue balances." Every package described in this book, whether it costs $50 or $15,000, can answer those kinds of questions.

While every package allows simple queries, there are some differences. Some PC DBMSs allow development of sophisticated applications software, particularly accounting and record-keeping systems. The authors of each product have their own unique approaches to setting up and using a data base. Your job is to find the approach that best suits your needs and abilities.

Figure 1-1 shows the old pre-data-base method of programming. Notice that each program is connected directly to the disk files it uses. Figure 1-2 shows the data base approach in which the DBMS controls all access to a single collection of data. In the first case, programs are forced to look at data stored in rigid structures called *files*, and there is no provision for queries. In the second case, the programs (and people) have a *logical view* of the data that is independent of the way the data is *physically* laid out on the disk. The data on disk

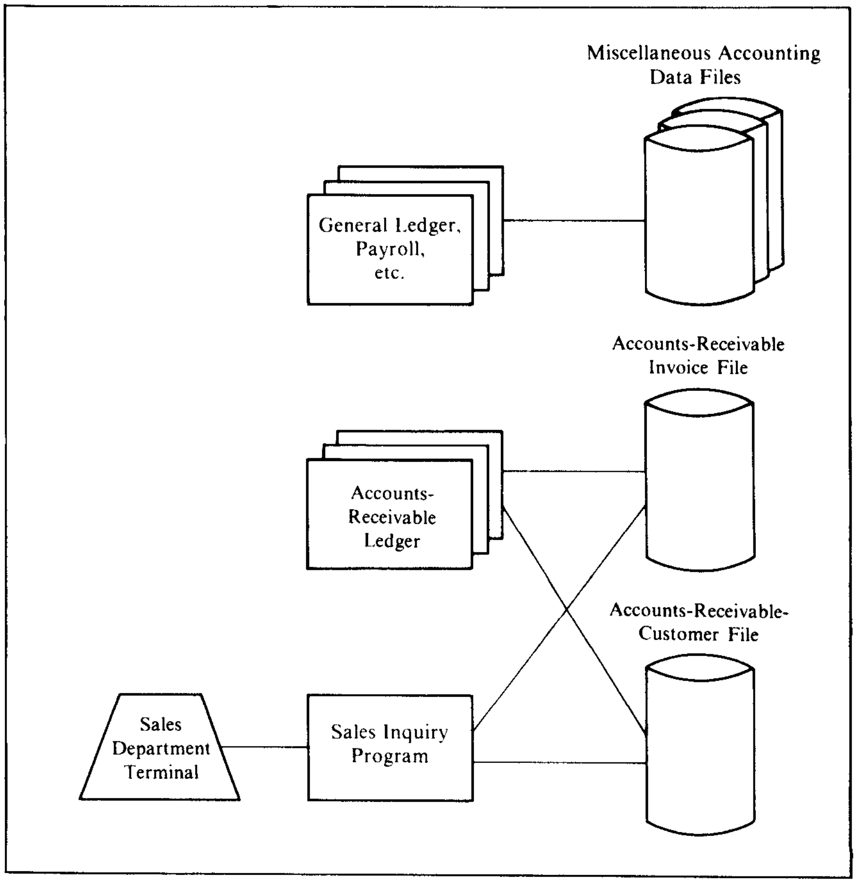

Miscellaneous Accounting Data Files
General Ledger, Payroll, etc.
Accounts-Receivable Invoice File
Accounts-Receivable Ledger
Accounts-Receivable-Customer File
Sales Department Terminal
Sales Inquiry Program

FIGURE 1-1. File-oriented programming before data base

can be changed or supplemented without requiring program changes, and it can be viewed in different ways.

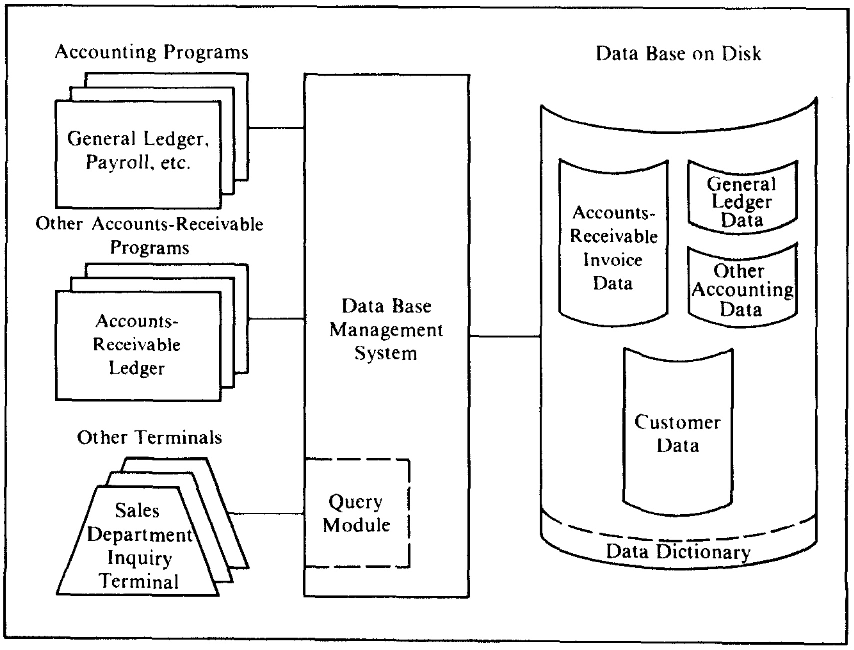
Remember that the DBMS is the computer program you buy and that the data base is your data after the DBMS has stored it on disk. You may or may not have to write your own programs, depending on which DBMS package you select.

DBMS BENEFITS

Benefits other than elegance must justify the expense of purchasing and installing a DBMS. You should consider the following benefits while evaluating each prospective DBMS package. These gains apply more to the sophisticated DBMS than to the simple file managers.

Accurate Data Modeling

Your company's information can be organized in a natural way. Perhaps you have vouchers grouped by vendor, or orders related to both customers and products. Each DBMS supports one or more specific data structures that may or may not apply to your situation. If you have a single application in mind for your DBMS, you can match the system to your requirements; but if you are obtaining a DBMS for



Accounting Programs
General Ledger, Payroll, etc.
Other Accounts-Receivable Programs
Accounts-Receivable Ledger
Other Terminals
Sales Department Inquiry Terminal
Data Base Management System
Query Module
Data Base on Disk
Accounts-Receivable Invoice Data
General Ledger Data
Other Accounting Data
Customer Data
Data Dictionary

FIGURE 1-2. The data base environment

general purposes, your job is not so easy. Chapters 2 and 3 will help you understand the strengths and weaknesses of some very different data structures.

Simple Data Organization

The structure of the data in a DBMS should be as simple as possible. The simpler the structure of the data base, the easier it is to manipulate the data.

Timely Response to Queries

Data base queries should be quickly formulated and executed, an improvement over the old days when a new report took weeks to create and had to be scheduled for overnight execution. Repetitive tasks such as customer-account inquiries should be completed in seconds, and more complex retrievals in minutes. Much of the data base design effort goes into ensuring the efficiency of often-used queries.

Cost Reduction

Disk storage is becoming cheaper, and labor is becoming more expensive. Most DBMS packages reduce the need for both, but don't look for a dramatic decrease in the amount of required disk space. The real DBMS cost saving is in programming time. With a DBMS, solving simple problems takes minutes or hours rather than days, often eliminating the need for a computer professional's involvement. Complex applications require only one programmer instead of a team of programmers. Reducing the cost of a modest set of business programs from $10,000 to $5,000 (including the cost of a DBMS) might enable a small company to enjoy the benefits of tailored software once reserved for large corporations. Don't kid yourself, though. A DBMS won't let you do a one-evening knock-off of an inventory system that took three years to develop on a mainframe.

Efficient Data Storage

Redundant data requires little disk space but has great potential for errors. Suppose you are processing orders for customers. If you store the customer's address in each order record, you will have to find and update each order if the customer moves. The DBMS environment makes it easy to store the customer's address only once, internally linked to each associated order.

Safeguards for Data Integrity

A data base is often composed of a number of files, records, data items, and interconnections. If something goes wrong—a power failure, for instance—all or part of the data base may be unreadable. The DBMS should detect this condition to prevent further catastrophic degradation of the data.

One integrity check is a *checksum,* the abbreviated sum of the numeric values of all the bytes in the data base disk files. The DBMS updates the checksum figure every time it changes the data base, and it frequently verifies the stored checksum against the checksum it recomputes. If there is a discrepancy, the DBMS can inform the operator that the data base is no longer usable. The fewer disk files involved in the data base, the better this method works.

If the DBMS can keep a checksum, it can also record the number of times the data base was logged onto. Thus, it is possible to see if anyone has been using the data base since you last logged off and to detect two sequential log-ons without a log-off, an obvious error condition.

The DBMS can also control integrity at a lower level. If a data base has a place for the price of an inventory item, for example, it would not make sense for the price to be less than zero or for the price to be XYZ or any other nonnumeric value. The DBMS watches for inconsistent data, preventing its entry and, if it should slip in, alerting the user to its presence.

Easy Access to Data

As a DBMS user you can ask a wide variety of questions about the data without an in-depth knowledge of its structure. Questions such as "What parts have a value greater than $50?" or "What is the total value of inventory?" are commonly asked by users who are not computer professionals. Most data base management systems have a *query language* for asking these questions. This language should be simple to learn and use.

When you query the data base, you usually get one screen of data. However, you may sometimes need a report—a nicely formatted