A Study of Database Management Systems for Microcomputers

Marielle Bergeron

A Major Technical Report

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

March 1985

ABSTRACT

A Study of Database Management Systems for Microcomputers

Marielle Bergeron

Database management systems for microcomputers are the result of an amalgamation of two of the most rapidly expanding technologies in computer science, microcomputers and database management systems. First introduced on the market at the onset of the eighties, these software products have already reached impressive levels with regard to both number of systems and functionality.

This Major Technical Report highlights the fundamental concepts related to the two technologies amalgamated and, more importantly, proceeds to a detailed review, analysis and evaluation of the characteristics and objectives of this new type of software.

Three successive generations of so-called database management systems for microcomputers have already appeared on the market. Each group of software programs is directed at a distinct market segment among various microcomputer users. The subset of functions peculiar to database management systems that is incorporated within a particular type of system increases with each new generation; in a general sense, moreover, the included subset meets the application needs of target users.

To Madeleine,

A STUDY OF DATABASE MANAGEMENT SYSTEMS FOR MICROCOMPUTERS

TABLE OF CONTENTS

INDEX OF FIGURES

I. INTRODUCTION

Database management systems for microcomputers are the result of an amalgamation of two of the most rapidly expanding technologies in computer science, microcomputers and database management systems.

In their thirteen years of existence, microcomputer systems have evolved at a dramatic pace in terms of number of systems, computing power and applications. Microcomputers reached the commercialization stage in the early seventies with the introduction of 4-bit computers used as process controllers and instrumentation monitors, immediately followed by 8-bit general-purpose microcomputers. The 16-bit microcomputer systems appeared in the marketplace toward the end of the last decade while 32-bit computers have become part of the microworld of the eighties.

In view of their low cost, microcomputers encompass an extremely wide range of applications, including, in increasing order of complexity:

- Industrial process controllers;
- Instrumentation monitors,
  e.g. - patient monitor systems in health care;
- Home computers,
  e.g. - control of heating and cooling systems,
  - video games,
  - personal computing (such as: budgets, appointment registers, word processing, etc.);
- Educational tools,

e.g. - didactical applications in schools;

. Small business computers;

. Multicomputer networks.

On the other hand, database management systems have developed as a revolutionary alternative to conventional data management systems. For an organization, these software programs present numerous benefits; one may consider, for instance, the following:

. A global view (schema) of the data (real-world entities and relationships) at the organization level, as well as individual views (sub-schemas) at the user level;

. Data independence that facilitates the development and maintenance of application software independent of environmental changes;

. Powerful languages to define, update and query the data;

. Better control of the data (access control, integrity constraints, concurrency control, data back-up and recovery).

The primary goal pursued in the present paper is to study the software product that resulted from a merger of the two young and most dynamic technologies presented above, i.e., database management systems for microcomputers. First introduced on the market at the onset of the eighties, these software products have already reached impressive levels with regard to both number of systems and functionality. New products

labelled as database management systems keep popping up at a striking pace in the microworld, nowadays more than ever.

Given the broad nature of the subject being addressed in the present paper, the route chosen to attain the previously stated goal adopts an overall management approach rather than a technical approach accessible solely to computer specialists. Thus, the aim will remain throughout the present report, to efficiently cover the widest possible range of areas implied in the study of database management systems for microcomputers, rather than to provide a detailed guide of specifications for these software programs.

Such an approach to the examination of microcomputer database management systems will enable us to recognize certain key points, especially:

- In less than five years of existence, three successive generations of so-called database management systems for microcomputers have already appeared on the market, namely: file management systems, single-user database management systems and multiuser database management systems.

- Each group of software programs is directed at a distinct market segment among various microcomputer users: individuals (file management systems), small businesses (single-user database management systems), and medium and large organizations (multiuser database management systems).

- The subset of functions peculiar to database management systems that is incorporated within a particular type of microcomputer database management system augments with each new generation; in a general sense, moreover, the included subset meets the application needs of target users.

- The selling price of a database management system program for microcomputers stands as good indicator of its category and capabilities: in the vicinity of $250.00 (file management systems), at approximately $500.00 (single-user database management systems), and over $1,000.00 (multiuser database management systems).

If one excludes the current introduction as well as conclusions and recommendations, the present paper is divided into six chapters. The two first chapters define the fundamental concepts related to each of the two fields in computer science involved here, microcomputers (Chapter II) and database management systems (Chapter III). Thus, these first chapters provide the groundwork for the direct examination, in subsequent chapters, of the subject dealt with in this report. Chapter II will first review each subsystem of a microcomputer system which significantly impacts on the development and implementation of a software program such as a database management system, namely: microprocessors (Section 2.1), mass storage devices (Section 2.2), and software tools (Section 2.3); secondly, this chapter will analyze the typical configurations of a microcomputer system taken as a whole

(Section 2.4). Chapter III will follow the evolution process of data management systems from the conventional approach to the file management approach (Section 3.1), to the database management approach (Section 3.2), and then proceed to present a description of the three data models for databases (Section 3.3), as well as a categorization of the main functions and features of a database management system (Section 3.4).

Chapter IV will initiate the amalgamation of the technologies introduced in the preceding discussion by determining what are the objectives of a database management system for microcomputers. Thus, this chapter will develop a categorization of the actual and potential users of microcomputers along with their management information needs: individuals (Section 4.1), small businesses (Section 4.2) and large organizations (Section 4.3).

Chapter V through VII contain the actual study of database management systems within the microworld context. Chapter V will survey database management system products currently commercialized on the microcomputer market, inside the categorization previously put forth for such systems: file management systems (Section 5.1), single-user database management systems (Section 5.2), and multiuser database management systems (Section 5.3). Chapter VI will present an overall evaluation of microcomputer database management systems with regard to the current state of the art in this field, both in terms of functionality (Section 6.1) and efficiency (Section 6.2); furthermore, this chapter will cover trends becoming apparent in these areas (Section 6.3). Chapter VII will establish analysis criteria pertinent to data models (Section 7.1), and

proceed to a comparative analysis of the two principal data models (relational and network) as applied to databases on microcomputers (Section 7.2), a question subject to most lively discussions in the current literature.

The extensive bibliography which provided us with the foundations of the present study of database management systems for microcomputers may be found at the end of this paper.

II. THE FUNDAMENTALS OF MICROCOMPUTER SYSTEMS

As set forth in the introduction, our primary objective in the current chapter consists of defining the basic concepts related to the first of two technologies embodied in a study of database management systems for the microworld, microcomputer systems.

Within this context, the essential question being addressed here, i.e., what is a microcomputer system, will obviously be limited to notions having a significant impact on the implementation of a software program such as a database management system. Nonetheless, in view of precisely the fact that the present study deals with a software program, one must proceed to the examination of a microcomputer system as a whole rather than solely discuss a given system component such as, for instance, microprocessors. Hence, let us undertake to trace the fundamental outlines of microcomputer systems that are relevant tc the study of database management systems for microcomputers by resorting to the three-level hierarchical classification for subsystems of microprocessor-based systems proposed in [INF 78], namely:

- Microprocessors,
- Microcomputers,
- Microcomputer systems.

At the lowest level in such a categorization, microprocessors principally include central processing units (CPU) for arithmetic, logic and control functions, as well as scratch pad memories. At the

intermediate level, microcomputers comprise, in addition to microprocessors, read/write (RAM) and/or read-only (ROM) memories along with input/output interfaces for peripheral control. Lastly, at the highest and most complex level, microcomputer systems include, besides microcomputers as previously defined, monitors, operating systems, and other utility programs referred to as system software (in contrast with application software), as well as peripheral devices. In view of the classification indicated above, the current chapter is divided into four sections, as follows:

- Microprocessors (Section 2.1),
- Mass storage devices (Section 2.2),
- Software tools (Section 2.3),
- Typical configurations of a microcomputer system taken as a whole (Section 2.4).

Section 2.1 will deal with the three main generations of general-purpose microprocessors: 8-bit, 16-bit and 32-bit microprocessors. The chief focus in that section resides in highlighting the following key points with regard to microprocessors:

(1) Each generation of microprocessors has introduced significant improvements over its predecessor, both in terms of power and functionality.

(2) Implemented improvements have reached a level such that the computing power of the newer microprocessors has been demonstrated to be comparable to that of minicomputers, indeed mainframes, in

certain specific areas (such as, for example, maximum direct address range).

The discussion presented in Section 2.2 with respect to the peripheral devices of microcomputer systems, will concern itself solely with disk drives, floppy and hard disks. Indeed, as will be covered in greater detail in Chapter VI of the present report dealing with the overall evaluation of database management systems for microcomputers, the characteristics peculiar to these mass storage devices can significantly impact on the efficiency of a database management system.

Having provided a brief account of the situation of software tools in the microcomputer environment, Section 2.3 will then proceed to examine in greater depth the operating systems for microcomputers, given that one must admittedly consider the particular effect of the latter systems on top of which database management systems actually run. The presentation of the three main generations of operating systems for microcomputers will follow the same approach as that chosen for microprocessors in order to focus on the principal divergences among systems belonging to each generation, namely:

- Single-user single-task operating systems,
- Multiuser multitasking operating systems,
- Networking operating systems.

To conclude the present chapter, Section 2.4 will provide an overview of the typical configurations of microcomputer systems available on the market in terms of the three major system components studied in the

preceding sections: microprocessors, mass storage devices and operating systems.

2.1 Microprocessors

Microprocessors have evolved at a dramatic pace; in just ten years of existence, from 1971 to 1981, no less than four generations of microprocessors have succeeded one another. The 4-bit microprocessors, pioneered by the Intel 4004 in 1971, were immediately followed in 1972 by the 8-bit chips, with the Intel 8008 opening the way. The 16-bit chips made their first appearance with the National Semiconductor PACE in 1974 while a fourth generation, the 32-bit microprocessors pioneered by the Intel iAPX-432, commenced in 1981. Thus, based on the assumption that a generation of microprocessors ends with the introduction of the pioneer chip of the next generation, the 4-bit, 8-bit and 16-bit chip eras have lasted for a period of, respectively, one, two and seven years. One must acknowledge, however, that such an assumption definitely does not hold true. Indeed, one can find simultaneously available in the microworld of the eighties, representatives of each generation of microprocessors; the smaller 4-bit as well as the low end 8-bit chips are used for a wide range of control applications, whereas the 16-bit general-purpose microprocessors still dominate the market.

In this line of thought, Figure 2-1 provides a listing of the main chip models marketed by five leading American manufacturers in the field of microprocessors (Intel, Motorola, National Semiconductor, Texas Instruments and Zilog), where such models are categorized in accordance with the four generations mentioned above. Figure 2-1 was essentially developed through the selection of a subset among microprocessors and

FIGURE 2-1 MICROPROCESSORS AND MICROPROCESSOR MANUFACTURERS

4-bit microprocessors	
Intel	4004 4040
National Semiconductor	COPS: 5782 5799 57140
Texas Instruments	TMS-1000 1100 1200 1300

8-bit microprocessors	
Intel	8008 8080 8041/8741 8048/8748 8035 8085 8088
Motorola	6800 6801 6802/6846 6805 6809
National Semiconductor	ISP8A600 (SC/MP-II)
Zilog	Z-80 Z-8

16-bit microprocessors	
Intel	8086 80286 iAPX-286
Motorola	MC68000 MC68010
National Semiconductor	PACE NS16032
Texas Instruments	TMS9980/9981 TMS9900 SBP9900 TMS9940
Zilog	Z-8000

32-bit microprocessors	
Intel	iAPX-432
Motorola	MC68020
National Semiconductor	NS32032

microprocessor manufacturers listed in [MCG 79] and the integration of the more recent 16-bit and 32-bit microprocessors analyzed in [GUP 83]. Three of the five manufacturers (Intel, Motorola and National Semiconductor) produce microprocessors belonging to each of the three most recent generations, the latter being of particular interest in the context of a study of database management systems for microcomputers.

In view of the widespread commercial distribution of microprocessors manufactured by both Intel and Motorola, these two families of chips were selected to serve as frame of reference for the analysis of microprocessors contained in this chapter. To that purpose, Figures 2-2 and 2-3 outline the principal characteristics peculiar to several chips produced by Intel (Figure 2-2) and Motorola (Figure 2-3). These two figures were developed through the aggregation and amalgamation of the data presented in various sources, mostly in [MOR 82] with respect to Intel models 8008 to 8086, in [GUP 83] for Intel 80286 and iAPX-432, as well as in [HAM 78, AND 82, MOT 79] in connection with, respectively, Motorola models 6800, 6809 and 68000. Although the findings put forth in the current section mainly derive from an analysis of those two specific families, one is permitted to conclude that other families of microprocessors follow suit.

In a general sense, the term microprocessor designates the arithmetic and logic unit (CPU) of a computer scaled down to fit on a silicon chip (sometimes a few chips) holding tens of thousands of transistors, resistors, and similar circuit elements [GUP 83]. However, this term is occasionally used in a broader sense to incorporate single-chip

FIGURE 2-2 INTEL MICROPROCESSORS: 8008 TO iAPX-432

	8008	8080	8085	8086	80286	iAPX-432
Year of introduction	1972	1974	1976	1978	1982	1981
Technology	P-MOS	N-MOS	N-MOS	H-MOS	H-MOS	H-MOS
Number of transistors per chip	2,000	4,500	6,500	29,000	130,000	219,000 (1)
Size of chip (mil^2)	16,800	33,750	37,050	48,400	70,225	100,000 (2)
Density (mil^2 per transistor)	8.4	7.5	5.7	1.7	0.5	1.36
Clock rate	.5-.8MHz	2-3MHz	3-5MHz	5-8MHz	8MHz	8MHz
Data bus width	8-bit (3)	8-bit	8-bit	16-bit (3)	16-bit (3)	32-bit (3)
Address bus width	8-bit (3)	16-bit	16-bit	16-bit (3)	16-bit (3)	32-bit (3)
Direct address range (bytes)	16K	64K	64K	1M	16M	16M
Pin count	18	40	40	40	68	64 (2)
Number of basic instructions	-	-	-	95	121	221
Total number of instructions	66	111	113	133	-	-
Number of condition flags	4	5	5	9	-	-

(1) On three chips
(2) Per chip
(3) Address and data bus multiplexed

FIGURE 2-3 MOTOROLA MICROPROCESSORS: 6800 TO 68000

	6800	6809	68000
Year of introduction	1974	1978	1980
Number of transistors per chip	5,000	15,000	69,000
Density (mil^2 per transistor)	9.1	2.7	1.0
Data bus width	8-bit	8-bit	16-bit
Address bus width	16-bit	16-bit	24-bit
Direct address range (bytes)	64K	64K	16M
Number of pins	40	40	64
Interrupt priority levels	2	3	8
Number of interrupt vectors	4	8	256
Number of data registers	2	1	8
Number of address registers	1	2	7
Number of stack registers	1	2	2
Number of condition flags	6	8	10
Control lines:			
Synchronous	Yes	Yes	Yes
Asynchronous	No	No	Yes
Modes of operation:			
User mode	Yes	Yes	Yes
Supervisor mode	No	No	Yes
Addressing modes:			
Implied	Yes	Yes	Yes
Immediate	Yes	Yes	Yes
Immediate, quick	No	No	Yes
Absolute, short	Yes	No	Yes
Absolute, long	Yes	Yes	Yes
Direct page	No	Yes	No
Data register	No	No	Yes
Address register	No	Yes	Yes
Indexed, no offset	No	Yes	Yes
Indexed, constant offset	Yes	Yes	Yes
Indexed, register offset	No	Yes	No
Indexed, register and constant offset	No	No	Yes
Indexed, post-increment	No	Yes	Yes
Indexed, pre-decrement	No	Yes	Yes
Absolute indirect	No	Yes	No
Indexed indirect	No	Yes	No
PC relative, constant offset	Yes	Yes	Yes
PC relative, register and constant offset	No	No	Yes

computers which group together, on a single chip, the three main components of a computer, namely the CPU, memory, and input/output interface. Thus, one can see that Figure 2-1 refers to the term microprocessor in this broader context; to illustrate, the Intel 8048, which represents an 8-bit single-chip microcomputer, is inserted in the 8-bit microprocessor category.

During their thirteen years of existence, microprocessors have undergone numerous and, more importantly, significant modifications and improvements. Thus, let us present an overview of the evolution process involved in microprocessors by looking at three areas where dazzling improvements were achieved, namely:

- Increased chip density,
- Larger word width,
- More powerful instruction set.

While placing particular emphasis on improvements attained from one generation of chips to the next, the following analysis will compare various models in order to demonstrate two crucial findings with respect to microprocessors. On the one hand, different chips within a given category offer a wide range of computing capabilities, the full extent of which increases in a most noticeable manner with each new generation introduced on the market. Secondly, the high end 16-bit microprocessors (Motorola 68000 and the like) as well as the 32-bit chips actually reach a level in computing power comparable to that of minicomputers.

Increased chip density

Based on the assumption that the number of transistors (or equivalent devices) on a chip would approximately double every year, Gordon Moore predicted that this amount would reach the one million benchmark by the mid-eighties. Although the number of transistors per chip actually advanced at an outstanding pace, Figure 2-2 shows that, in the case of Intel microprocessors, this amount effectively doubled every second year rather than annually. Indeed, the 8008 held 2,000 transistors in 1972 whereas the 80286 introduced in 1982 contained 130,000 devices. Nevertheless, the prediction stated by Moore may yet come fairly close to reality if one considers the exploit recently achieved by Hewlett Packard; in 1982, this company brought into the marketplace a 32-bit CPU microprocessor for its HP-9000 computer system which holds 450,000 devices on a 48,400 mil^2 chip. Thus, taking into account the Hewlett Packard 32-bit chip, the number of transistors per chip has increased by a factor of 200 over a ten-year span, i.e., from 2,000 transistors (Intel 8008) to 450,000 devices (HP 32-bit CPU), whereas the chip density has advanced by a factor of nearly 100 in the same period, i.e., from 10.0 mil^2 per transistor (Motorola 6800) to 0.1 mil^2 per device (HP 32-bit CPU).

Needless to say, those represent very impressive figures. The key point remains, however, to determine the nature and extent of benefits obtained through the inclusion of an ever increasing number of devices and functions on a single chip. Briefly stated, such an integration

results in continuously greater cost savings derived from many factors, among which:

- The elimination of external connections (along with related labor costs);
- A better reliability of microprocessors (ending, as a consequence, in lower maintenance expenses);
- A smaller chip size (resulting in lesser power consumption, reduced cooling requirements as well as lower packaging costs).

Furthermore, the increased number of devices on a chip provides manufacturers with tremendous potential for implementing sophisticated features into the hardware; to illustrate, the Intel iAPX-286 incorporates the memory protection function onto the microprocessor chip itself.

Larger word width

The word width, or word size of a microcomputer represents the paramount single criterion used for the evaluation and selection of microprocessors. This term designates the number of bits of data which can be transfered in parallel. Hence, the word width directly impacts on the processing speed, the maximum memory address range, and the power of the instruction set of a microcomputer.

Prior to undertaking an individual analysis of the three significant effects mentioned above, let us first clarify an important point here.

The categorization of general-purpose microprocessors within the 8-bit, 16-bit and 32-bit generations discussed in the introduction of this chapter, does not always imply a straightforward exercise. This situation is caused by the fact that in certain microprocessors, the processor internal paths (as represented by the address bus) display a word width larger than do the external paths (being represented by the data bus). The Motorola 68000 and the Intel 8088 (on which is based the IBM PC computer system) microprocessors stand as two well known examples of such cases: indeed, the 68000 chip incorporates a 32-bit internal architecture with a 16-bit data bus, whereas the 8088 microprocessor depicts a 16-bit internal architecture with an 8-bit data bus. In the literature, one usually finds both of these models classified under the 16-bit microprocessor category. Within such a context, one needs to specify that the categorization of Intel and Motorola microprocessors represented in Figures 2-2 and 2-3, respectively, was effected on the basis of the word width of the external paths (i.e. the data bus width). Bold characters underline the resulting classification in both figures.

As previously mentioned, the processor word size has an immediate impact on the maximum direct address range of a microcomputer. With their 16-bit address bus, for instance, the low end Motorola 6800 and 6809 microprocessors may only access a maximum of 64K (2^{16}) bytes of memory. In contrast, the 68000 can directly address up to 16M (2^{24}) bytes of memory with its 24-bit address bus. One realizes that this represents more than 250 times the addressing range of the two preceding models, more than many minicomputers commercially available today and as

much as an IBM/370 computer. Over and above that, the Motorola 68020 may, like mainframes, directly address gigabytes (2^{32}) of memory with with its 32-bit address bus.

With respect to the family of Intel microprocessors, the rise in the direct address range of the 80286 (16M bytes of memory) as compared to that of the 8086 (1M bytes), is mostly attributable to the multiplexing of both the data and address buses rather than to a physical increase in the address bus width. Thus, throughout the evolution of the Intel family, an augmentation in the number of pins per chip accompanied the increase in the word width so as to keep the overhead related to multiplexed buses at an acceptable level; for example, from 18 pins (Intel 8008) to 68 pins (Intel 86286).

More powerful instruction set

A larger word size also implies a more powerful instruction set with greater addressing capabilities. Figure 2-2 corroborates the first part of this assertion: one can see a noticeable increase in the number of instructions offered from one Intel model to the next, ranging from the 66 instructions supplied in the 8-bit 8008 to the 221-instruction set incorporated in the 32-bit iAPX-432. Furthermore, Figure 2-3 confirms the latter part of the relationship put forth above: indeed, the number of available addressing modes goes from six modes in the Motorola 6800 to fourteen for the 68000. One must recognize, however, that an outstanding rise in the number of addressing modes had already

been achieved from the 6800 to the 6809, even though both models belong to the same 8-bit generation of microprocessors. This improvement was obtained by increasing the number of bytes contained in one instruction, which went from a maximum of three bytes in the Motorola 6800 to five bytes in the 6809. An extra byte was inserted in the latter model to enable the specification of additional opcodes over and above those used in the 6800, while the second additional byte, or post-byte, allows for the specification of variations in the indexed and indirect addressing modes.

In this line of thought, significant divergences may exist among numbers of instructions quoted in various articles and/or booklets dealing with a given microprocessor. Thus, one must interpret such figures with due care. In some instances, the number provided refers to basic instructions while in other cases, it denotes basic instructions along with their variations. This observation equally applies to the number of addressing modes supported by a computer.

Besides the three principal areas of improvement discussed above, one can extract additional improvements from a review of Figures 2-2 and 2-3. The clock frequency of the Intel family has increased by a factor of nearly 20 (from .5MHz to 8MHZ). This augmentation is even far greater if one takes into account, for example, the 18MHz average clock rate of the Hewlett Packard 32-bit CPU microprocessor. In addition, more and more sophistication has been incorporated into the interrupt structure of the Motorola family (from a 2-level and 4-vector interrupt in the 6800 to an 8-level and 256-vector interrupt for the 68000); thus, the

high end processors can support a wider variety, and accomodate a larger number of interrupt devices. A supervisor mode of operation was also added to the 68000 for multiprogramming. Furthermore, the CPU includes an increased number of internal registers in order to reduce the number of memory accesses required to perform an operation (from four 8-bit registers in the 6800 to seventeen 32-bit registers for the 68000).

One may pointedly conlude from the present section that microprocessors, although the product of a relatively young technology, have attained, at an outstanding pace and through drastic improvements, a stage of development such that the recent high end 16-bit 68000-like processors as well as the 32-bit chips closely resemble minicomputers in terms of computing power. Furthermore, according to benchmark tests reported in [GUP 83], the overall performance of newer microprocessors actually approaches the effectiveness reached by mainframes. For instance, the Intel 432 processor at 8MHz clock frequency takes 6.375 μs for a 32-bit integer multiply and 27.875 μs for an 80-bit floating-point multiply whereas corresponding figures for the IBM 370/148 computer currently stand at, respectively, 16.0 and 38.5 μs.

2.2 Mass storage devices

One can find a wide variety of disk drives for microcomputers currently available on the market, ranging from 5 1/4-inch single-sided, single-density minifloppy disk drives to 14-inch hard disk drives. Prices may run from $250.00 to $5,000.00 while storage capacities can range from 100K bytes to over 30M bytes.

A lower price tag obviously constitutes the principal advantage associated with floppy disk drives as compared to hard mass storage devices. One must bear in mind, however, that floppies constitute easily damageable devices. Moreover, other disadvantages inherent in floppies may be pinpointed through a comparative review of the characteristics of floppy and hard disk drives outlined in Figure 2-4; such drawbacks include:

- Lower storage capacity,
- Slower transfer rate and rotational speed,
- Higher average access time,
- A maximum of two tracks per cylinder.

Figure 2-4 summarizes and combines the technical specifications provided by Shugart Associates for both types of mass storage devices [SHU 77, SHU 79]. As indicated in the introduction to the current chapter, such characteristics have a significant impact on the efficiency of a database management system, an area that will be covered in greater depth in Chapter VI.

FIGURE 2-4 FLOPPY AND HARD DISK DRIVE SPECIFICATIONS

	8" floopy disk single-density double-sided	8" hard disk non-removable
Capacity (unformatted):		
Bytes per drive	800K	10.67M
Bytes per surface	400K	2.67M
Bytes per track	5.2K	10.40K
Transfer rate	250K bits/sec.	4.32M bits/sec.
Average latency	83ms	9.6ms
Access time:		
Track-to-track	18ms	19ms
Average	91ms	70ms
Rotational speed	360rpm	3,125rpm
Number of cylinders	77	256
Number of tracks	154	1,024
Number of heads	2	4
Drive price	$ 600.00	$ 2,000.00

2.3 Software tools

Software tools for the 8-bit microcomputer systems have matured over the last few years. In today's marketplace, most 8-bit microcomputer systems come fully supported by high-level languages (principally, BASIC and UCSD Pascal), monitors and utilities. Although somewhat lagging, the 16-bit microcomputer are catching up very quickly. Thus, the latter systems will incorporate a wider variety of high-level languages (such as standard Pascal, C, COBOL, LISP, etc.), increased utilities, as well as sophisticated software development tools such as symbolic debuggers. This observation applies even more so to the generation of 32-bit microcomputer systems.

Even if one entirely excludes operating systems dedicated to a specific computer system, an important number of different operating systems can currently be found in the microworld. Figure 2-5 presents sixteen of the best-known operating systems for microcomputers marketed to run on more than one computer system, grouping the selected systems within three generations; thus, the present section will be further divided into three parts, each subsection covering one category of operating systems for microcomputers, namely:

- Single-user single-task operating systems (2.3.1),
- Multiuser multitasking operating systems (2.3.2.),
- Networking systems (2.3.3).

The above classification actually reflects various stages followed

FIGURE 2-5 OPERATING SYSTEMS FOR MICROCOMPUTERS

Single-user single-task operating systems		Multiuser multitasking operating systems	
Digital Research	CP/M-80 CP/M-86	A T & T	UNIX (Xenix)
Microsoft	MS-DOS (PC-DOS)	Digital Research	MP/M-11 MP/M-86 CCP/M-86
SofTech Microsystems	UCSD p-System	Phase One Systems	Oasis
Technical Systems Consultants	FLEX	Pick Systems	Pick
		Technical Systems Consultants	UniFLEX

Networking operating systems	
Digital Research	CP/NET
Microsoft	TurboDOS

through the evolution of operating systems for microcomputers. Indeed, the operating systems first developed for the microenvironment were scaled to match the limited capabilitites of early personal microcomputers, i.e. restricted amounts of memory space, minimal disk storage systems as well as 8-bit microprocessors. As such, those systems were designed as single-user, single-task operating systems. The more powerful 16-bit microcomputers then appeared on the market and, along with them, the multiuser multitasking operating systems. Finally, networking operating systems were recently introduced in the microworld in order to enable a group of local personal microcomputers to share global resources such as a shared database on hard disk. One can observe in Figure 2-5 that Digital Research represents the sole operating system designer producing systems belonging to each of those three generations.

Given the fundamental interrelationship between database management sytems and operating systems, let us pursue further the study of operating systems for microcomputers by presenting certain basic particularities and limitations of operating systems within each of the categories previously defined.

2.3.1 Single-user single-task operating systems

In the context of low end microcomputer systems based on 8-bit microprocessors, CP/M-80 and FLEX developed by, respectively, Digital Research and Technical Systems Consultants constitute the two most commonly used operating systems. Both software programs are written in assembly language. Originally designed to run on the Intel 8080 and the Zilog Z-80 microprocessors, CP/M-80 has gradually established itself as the de-fato standard for 8-bit machines. FLEX, on the other hand, is primarily used on the Motorola 6800 and 6809 processors.

These operating systems solely support a single user in a monoprogramming environment. Furthermore, they also present the following limitations:

- No provision is made in order to prevent unauthorized users getting access to the system (such as user password).

- The underlying file system incorporates no file protection mechanisms aimed at controlling the sharing of files (such as file password, read/write only access, etc.)

- The system supports only two types of files, sequential and direct access files.

- The memory space allocated to an opened file generally represents a single-block buffer, thereby resulting in a disk read every

single time a new block is read from a sequential file. This situation causes two disk reads in a direct file environment.

- In certain operating systems such as UCSD p-System [BOW 80], file blocks are stored on contiguous sectors on disk in order to minimize seek delays and simplify the address mapping process.

- The input/output error trapping and recovery system usually remain rudimentary (for example, in CP/M operating systems [TAY 82]).

Single-user single-task operating systems have also been developed for 16-bit microcomputers. Examples of such systems include CP/M-86 introduced by Digital Research for the Intel 8086 and 8088 microprocessors, as well as MS-DOS designed by Microsoft and marketed by IBM under the name PC-DOS. An analysis of these two operating systems [TAY 82] reveals that they are quite similar to previous versions of CP/M.

2.3.2 Multiuser multitasking operating systems

The principal multiuser multitasking operating systems for microcomputers include the following:

- MP/M and CCP/M-86 (Digital Research);
- UniFLEX (Technical Systems Consultants);
- UNIX (developped at Bell Laboratories) along with its many commercial versions, for instance: Xenix;
- Pick (Pick Systems).

Digital Research offers two versions of MP/M: MP/M-11 for 8-bit microcomputers, and MP/M-86 for 16-bit machines. In both versions, the multiuser multitasking environment is created through the segmentation of the physical memory of a computer according to specifications defined by the user at sysgen. The operating system then assigns a segment to a specific user on request [DAH 82]. To my knowledge, Digital Research stands as the only software manufacturer currently marketing a version of a multiuser multitasking operating system to run on 8-bit machines. In point of fact, writers appear unanimous in asserting that 8-bit microcomputer systems do not provide the computing power necessary to efficiently support a multiuser multitasking environment, even where a very small number of concurrent users (such as two) are involved. Hence, all the other operating systems enumerated above were implemented on 16-bit microcomputer systems.

In 1983, Digital Research introduced a second multiuser multitasking operating system named Concurrent CP/M-86 (or CCP/M-86 in short) for

16-bit machines @KOR 83|. In comparison to MP/M-86, the newer operating system incorporates two additional features: firstly, a multiwindow environment enabling the user to initiate and follow the progression of multiple applications simultaneously running on the system; secondly, priority pre-emptive task scheduling particularly suitable for real-time applications.

In contrast with the multiuser multitasking operating systems described so far, UNIX and Pick were not originally designed for microcomputers; rather, both systems migrated from the minicomputer environment to the microworld. UNIX was first developed for minicomputers by Bell Laboratories in 1969. It represents a portable software product written in C and was mainly used, until recently, on DEC PDP-11 minicomputers. UNIX supports several features of particular interest for a time-sharing system, namely: a virtual paged memory, the notion of process and of interprocess communication, as well as a multilevel file directory. As pointedly indicated by Hughes in @HUG 82|, the development of a time-sharing operating system outdoing UNIX would involve a time-consuming effort such that the commercialization of the product would be delayed for years. In my opinion, the availability of UNIX at the very moment the more powerful 16-bit microcomputer systems appeared in the marketplace, combined with the excellent reputation of this software program among user groups (including universities and other research centers), certainly provide sufficient justification for its popularity in the microworld.

Major drawbacks associated with UNIX have somewhat slowed down the

market penetration achieved by vendors of its various commercial versions. Such disadvantages mostly pertain to the relatively high price of this software product, the amount of disk storage required, as well as generally poor vendor support. AT&T apparently decided to enforce vigorous measures aimed at offsetting at least two of these three elements. Indeed, the February 21, 1983 issue of Business Week reported that AT&T, in addition to slashing the established $750.00 licence fee for UNIX to as little as $100.00, would henceforth provide technical assistance to companies using the UNIX system.

Pick, marketed by Pick Systems, incorporates two noteworthy features in the context of the present study of database management systems for microcomputers: this operating sytem supports a database of the relational type and provides an easy-to-use English-like query language. These added functions, borrowed from database management systems, actually give Pick the makings of a database operating system naturally suited for business and managerial applications [COO 84].

In this line of thought, let us underline here the principal conclusion drawn by Stonebraker, who evaluated UNIX as operating system support for a database management system within the context of the development of INGRES [STO 81]. In this author's opinion, a database management system designer would prefer to select a small efficient operating system comprising only desired services, such as the so-called real-time operating systems, rather than UNIX or similar general-purpose operating systems; the latter systems endeavour to provide all things to all people, at the cost of much higher overhead.

2.3.3 Networking operating systems

Figure 2-5 includes only two operating systems belonging to the generation of networking operating systems, CP/NET designed by Digital Research and TurboDOS developed by Microsoft. The recent introduction of pioneer operating systems of this category accounts for the limited number of systems currently available in the microworld. However, in view of the experience of the two preceding generations, one is permitted to foresee that the market will witness a proliferation of networking operating systems for microcomputers within a short period of time.

For several microcomputer manufacturers and users, networking operating systems often appear as a step forward in the advancement of microcomputing that is more logical than multiuser, multitasking operating systems. Such a perception derives from the fact that this third generation allows an already installed base of single-user microcomputer systems to share global resources as is permitted in a multiuser environment, without the drawbacks peculiar to time-sharing systems.

To conclude the current section dealing mainly with microcomputer operating systems, let us mention that, to date, none of the operating systems discussed previously seems to have established itself as the de-facto standard for the 16-bit microcomputer systems. One will

remember that CP/M-80 is regarded as such a standard for 8-bit microcomputer systems. As a point of fact, numerous articles in the recent literature, [HUG 82, PHI 83] among others, attempt to pinpoint which specific operating system will be recognized as the yardstick for 16-bit machines. Only three operating systems, CP/M-86, MS-DOS and UNIX actually stand out as generally acknowledged true contenders in this race. One can see that the question being addressed here essentially centers on determining whether the 16-bit microcomputer systems primarily represent single-user or multiuser systems.

According to Phillips [PHI 83], the following polarization prevails in the microworld: Intel 8086-based systems predominantly represent single-user machines while Motorola 68000-based microcomputers most frequently constitute multiuser systems. If such polarization subsists in the future, we will be permitted to regard UNIX as the de-facto standard for 16-bit high end multiuser systems. In my opinion, MS-DOS will be recognized as the yardstick for 16-bit single-user low end systems, given principally the dazzling popularity of IBM microcomputer systems.

In view of the computing power provided by the 32-bit microcomputer systems as well as the preceding discussion, one is almost automatically led to categorize all products of this type as multiuser systems. Should this be the case, it remains to be seen if UNIX will also establish itself as the de-facto standard for this microcomputer category. In my opinion, its principal competitors will be recruited much more among operating systems migrated from the minicomputer

environment, such as Pick, than in the class of upgraded versions of operating sytems designed for microcomputers.

2.4 Typical configurations

The three first sections of the current chapter clearly highlight the fact that a microcomputer system for data processing could conceivably combine elements presenting most divergent characteristics and power capacities. The lowest end on a continuum of microcomputer systems could correspond, for instance, to a system composed of an 8-bit microprocessor, somewhere between 24K to 48K bytes of main memory, and one 5 1/4-inch single-sided, single-density minifloppy disk drive with a capacity of 100K bytes, running a single-user, single-task operating system. At the utmost point, one could find a microcomputer system based on a 32-bit microprocessor with millions bytes of main memory, two or more 14-inch hard disk drives with a minimum capacity of 30 megabytes per disk, running a multiuser, multitasking operating system concurrently supporting at least six to eight users.

In practice, however, configurations of microcomputer systems for data processing may actually be categorized in two groups located on both sides of the midpoint on the continuum referred to above. Microcomputer systems belonging to the first class primarily represent single-user systems based on either an 8-bit microprocessor or a low end 16-bit 8086-like microprocessor with 128K to 256K bytes of memory, two 8-inch single-sided, double-density floppy disk drives with a capacity of about one million bytes per disk, CP/M or MS-DOS as operating system, and BASIC as programming language. A complete configuration for such a system, printer and terminal included, could cost somewhere between

$5,000.00 and $10,000.00.

Microcomputer systems within the second group present a configuration which closely resembles that of minicomputer systems. Thus, these products primarily correspond to multiuser systems based on either a high end 16-bit 68000-like or a 32-bit microprocessor with a minimum of 512K bytes of memory, one 30M byte hard disk drive along with either a floppy disk drive or tape drive for back-up, UNIX as operating system, and a set of high-level language translators including, as a minimum, COBOL. Capable of supporting four to six users, the price range of such a configuration, including a printer and a few terminals, could run between $20,000.00 and $35,000.00; hence, approximately five times less than a minicomputer system with an equivalent configuration.

A third category may be added if one elects to segregate among single-user microcomputer systems, those that would be interconnected over a local area communication network in order to share global resources such as a high capacity hard disk, and/or a high speed, high quality printer.

III. THE FUNDAMENTALS OF DATABASE MANAGEMENT SYSTEMS

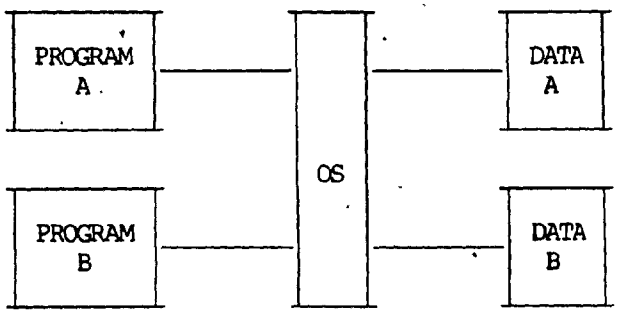
Chapter I has set forth the primary goal being pursued in the present paper, namely to study database management systems for microcomputers. As previously mentioned, this software has evolved from a merger of two young and rapidly expanding fields in computer science, microcomputers and database management systems.

Chapter II concerned itself with tracing the fundamental outlines of one of those technologies, microcomputers. Let us now undertake, in the present chapter, to define the basic concepts related to the second field of applied computer science, databases and database management systems. Thus, the notions introduced in the previous and current chapters will provide the groundwork for the study of database management systems for microcomputers contained in subsequent chapters.

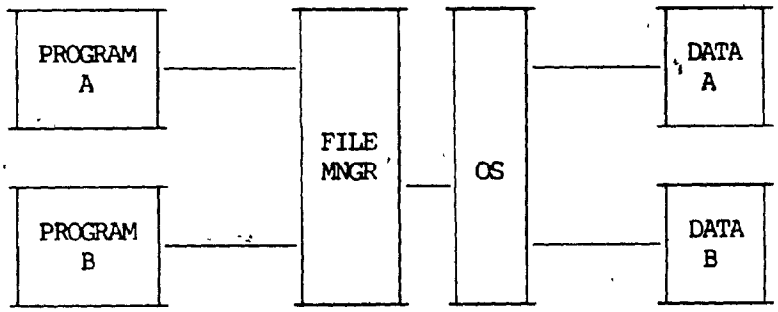
The route chosen to answer the essential question beind addressed here, i.e., what is a database management system, consists of following data management approaches through their evolution towards the concepts involved in databases and database management systems. As discussed in [ELB 82], one can distinguish three stages and two significant breakthroughs in that evolution process. In a first step, data management moved from what has been designated as the conventional approach to the so-called file management systems. Continued advances in this area led to the later move from the file management approach to what is known as database management systems. Figure 3-1 depicts those

FIGURE 3-1 THE EVOLUTION OF DATA MANAGEMENT SYSTEMS

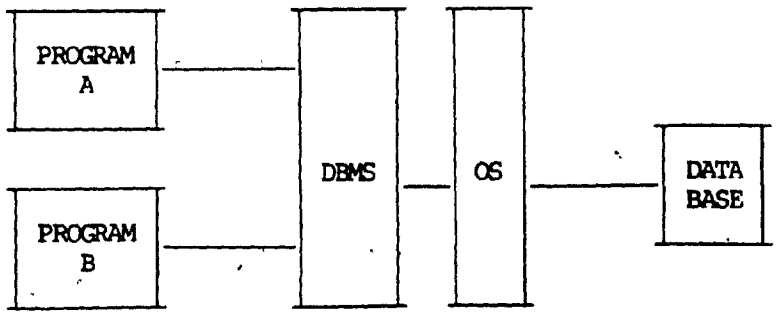
(a) The conventional approach



(b) The file management approach



(c) The database management approach



three stages. The original figure pictured in [ELB 82] has been adapted here in order to highlight the key notions covered in our discussion of the two major steps involved in the evolution process mentioned above, namely:

- The file management approach (Section 3.1),
- The database management approach (Section 3.2).

It is important to point out, at this time, that the move into file management systems was primarily motivated by a pressing need to reduce the programming effort necessary to define and maintain the more complex file structures required to support the generalized switch made in software from batch application programs to on-line applications. Improvements achieved in that first step were later conveyed to database management systems and actually form an integral part of the physical level in the database management approach. The second major move added the concepts of data integration and data sharing at the logical level of database management systems.

Given that the development of data management approaches for microcomputers followed the same path as that described above, basic notions involved in this evolution need to be emphasized. Furthermore, as we will later find out in Chapter V of this report, file management systems for microcomputers have often been, and still are, incorrectly classified and/or labelled as database management systems for microcomputers by software manufacturers and vendors, as well as in advertisements and in articles covering this type of software.

Considering that data models are the crucial means of representation and manipulation of real-world data for computing purposes, Section 3.3 of the present chapter will discuss the three main data models supported by current database management systems, namely:

- The relational model,
- The network model,
- The hierarchical model.

The chief focus in that section will consist of clearly identifying, for each data model, fundamental charateristics that allow for the proper classification of a given data model as being supported by a particular database software.

It is generally acknowledged in the industry, that the network model, while retaining all essential structures provided in the hierarchical model, goes beyond the latter model in allowing a closer representation of complex real-world data. Thus, the drop in popularity of the hierarchical model was reflected in the microworld, where hierarchical-based database management systems are nearly nonexistent. Hence, the emphasis will be placed in the present report on the relational and the network models.

Section 3.4 of this chapter will establish a precise framework within which one can describe the features of database management systems for microcomputers later surveyed in Chapter V of the present report. Features of database management systems for mainframes provide the basis for the framework developed here. Furthermore, this classification of

features will also permit us to proceed, in a systematic manner, to an analysis of the particularities and limitations of database management systems for microcomputers, to be presented in Chapter VI of this paper.

3.1 The file management approach

Figure 3-1 (a) depicts a graphical illustration of the conventional approach to data management. One can see that, under that approach, application programs call directly on the file management module of the operating system through the programming language being used. Given that applications were almost exclusively of the batch type at the time, the simple file structures and access methods included in the operating system provided sufficient support for application programs. Thus, the conventional system was sustained by sequential and direct access files managed by the file management portion of the operating system.

The introduction of on-line applications in the computer world caused a surge in system requirements in terms of file structures and access methods. Thus, in order to improve on the efficiency of systems supporting on-line applications as well as to facilitate the programming of such systems, software products designated as file management systems were, in a first step, developed and commercialized. VSAM, introduced on the market by IBM, is an example of such a product. This type of software was primarily aimed at providing programmers with file structures and access methods complex enough to meet the requirements of on-line applications, thereby allowing for the use of keyed access files such as indexed-sequential files, B-trees, hashed files, etc.. Indeed, keyed access files are paramount to reaching a level of response time acceptable in the on-line environment. However, one must consider that

the programming of those access methods can be extremely time-consuming; furthermore, it usually requires extensive programming abilities.

Figure 3-1 (b) illustrates the file management approach. The two fundamental points to be retained from that figure are the following:

(1) This approach implies the availability of complex file structures and access methods to support on-line requirements. These system requirements are grouped in a set of programs that make up the so-called file management system, which may consist of a group of stand-alone programs.

(2) Each application program possesses its own data, which is stored in one or more data files being created and maintained through the use of the file management system. Thus, the data exclusively serve single applications. The concepts of data integration or data sharing are not applied in any form whatsoever in that approach.

As previously mentioned, file management systems play an important role in the microworld. File managers marketed for microcomputers and, to a large extent, for mainframes as well, are generally designed solely to perform the specialized functions of file definition and record management through the use of a low level system/user interface language, the latter usually corresponding more or less to a set of batch procedure calls. Thus, such file management systems do not, for the most part, incorporate features designed to facilitate the input, the query and/or the reporting of the data contained in data files such

as, for instance, screen handlers, form generators, multicondition query languages or report writers. This area will be covered in greater depth in subsequent chapters (Sections 4.1 and 5.1).

3.2 The database management approach

To attain goals which differed from those pursued in the development of file management systems, the approach to file management evolved, in its second major step, towards database management systems. It should be noted, however, that all former systems have not been replaced by database management systems. Indeed, huge sums of money had been previously invested for the development of application programs based on the file management approach. Nevertheless, it appears evident at this point in time that, with the exception of some specific applications, the industry will gradually phase out file managers to the benefit of database management systems, the completion of this process being simply a matter of time.

Figure 3-1 (c) graphically depicts the database management approach. One can clearly see that the primary objective sought in that approach is to achieve the integration of all the data of an enterprise in what is called a database or databank. Thus, data are no longer serving single application programs. In that context, a database may be defined as "a collection of interrelated data stored together without harmful or unnecessary redundancy to serve multiple applications; the data are stored so that they are independent of programs which use the data; a common and controlled approach is used in adding new data and in modifying and retrieving existing data within the data base. The data is structured so as to provide a foundation for future application development" [MAR 77].

The advantages derived from using a database management approach rather than the conventional (and file management) systems are numerous and generally well understood in the computer world. Let us cover here, for definition and reference purposes, the six following main advantages of a database management system [MAR 77, ELB 82]:

- An integrated view of the data,
- Data independence,
- Controlled data duplication,
- Accuracy and consistency of data,
- Sharing of data,
- Adaptability of ad hoc queries.

An integrated view of the data

The database approach supports an integrated view of all the data of an enterprise, such data being accurately regarded more and more as representing one of the greatest assets of an organization. The approach actually favours a far better understanding and structuring of the data. Furthermore, it creates an environment where tighter and more effective controls can be put in place in order to safeguard this important asset.

Data independence

At the logical level of a database management system, the design of the data may be altered without having to rewrite existing application

programs. At the physical level, storage hardware and physical storage techniques may be changed to improve data storage requirements and/or data access times without causing the rewriting of applications. Thus, the maintenance of existing application programs is made much simpler. This, in turn, results in savings in programming labour costs, a significant advantage when one considers that it is now generally acknowledged in the industry that costs related to the maintenance of existing applications may easily exceed the costs incurred for the development of new application programs.

Controlled data duplication

Given that the data are no longer serving single applications, data are now stored only once, with the exception of cases where technical or economic considerations provide a justification for redundant storage of data.

Accuracy and consistency of data

Data consistency is improved as compared to that attained in the conventional (and file management) systems. Indeed, a database approach to file management inherently avoids having multiple versions of the same data to serve multiple applications being in different stages of updating. This, however, necessitates the installation of concurrency controls to ensure the synchronization of updating transactions. In addition, integrity controls such as range checks are used to detect, where possible, data inaccuracies.

Sharing of data

Under the database management approach, the same data serve multiple applications as well as multiple on-line users, generally at one time (multiprogramming, multiuser mode). This calls for access controls to be established in order to prevent unauthorized access to the data. The need for concurrency controls also comes into play here.

Adaptability for ad hoc queries

In the environment of database management, spontaneous requests for data can be handled without application programs having to be written. This is achieved by means of high-level query and/or report generation languages.

In such a context, a database management system can therefore be defined as a general-purpose set of programs aimed at performing the two following roles:

(1) A data definition language, i.e., to support the definition of the database;

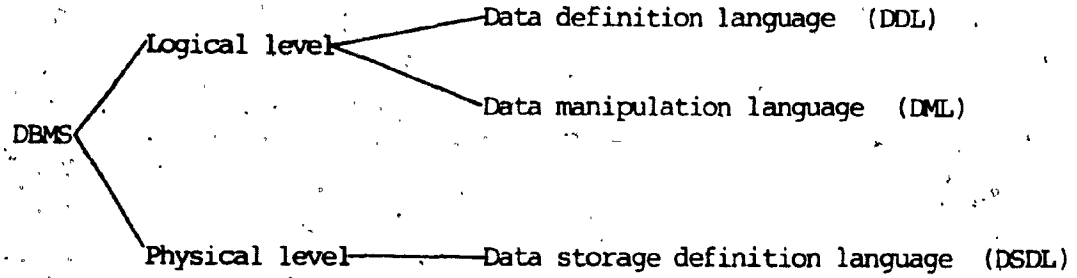
(2) A data manipulation language, i.e., to aid and control the use of the database for adding, modifying and retrieving data.

Needless to say, a database management system must also perform file

management functions associated with the area of the physical storage of data. Thus, a database management system may be conceptually visualized as a two-level software with three supporting data languages [INF 83], as illustrated in Figure 3-2.

At the logical level, one may currently find in database management systems three different data models to support the representation and manipulation of real-world data for computing purposes, namely: the relational model, the network model and the hierarchical model. The next section is devoted to a study of these three data models.

FIGURE 3-2 A DATABASE MANAGEMENT SYSTEM (DBMS)



3.3 The three data models

To provide an overview of the three main data models on which are based the vast majority of current database management systems, we have chosen to follow the methodology put forth by Tsichritzis in his book dealing with data models @TSI 82|. Pursuant to the proposed framework, this section is divided into three parts, each subsection discussing one of the models, i.e., the relational model (3.3.1), the network model (3.3.2) and the hierarchical model (3.3.3). Each subsection is further divided in order to address the three following areas pertaining to a data model:

- Structures,
- Integrity constraints,
- Operations.

Structures

This part will provide a description and analysis of the data structures used by the model in order to represent real-world data. The latter data include both the entities found in the real world and their attributes:

e.g. EMPLOYEE (employee-no, name, address, salary, dept-name, manager-no)

DEPARTMENT (dept-name, floor-no);

as well as the relationships existing between those entites:

e.g. EMPLOYEE.dept-name = DEPARTMENT.dept-name

EMPLOYEE.manager-no = EMPLOYEE.employee-no.

Embodied in such a description will be an attempt to provide and explain the terminology used in reference to a particular model. However, the main focus will remain to highlight the limitations inherent in the data model presented.

Integrity constraints

This area will address the methods incorporated in a model in order to ensure the accuracy of data within the database. As pointedly mentioned by Tsichritzis [TSI 82], the meaning of the data as well as the data themselves must be reflected in the model. For example, a model must permit its users to specify that, in accordance with a policy established by the enterprise, an employee can never be subject to a decrease in salary; thus, an integrity constraint to that effect is incorporated within the database management system:

e.g. ASSERT ON UPDATE OF EMPLOYEE

NEW salary > OLD salary.

In cases where the model does not support the definition and enforcement of such semantic constraints, the responsibility to ensure the accuracy of data within the database automatically lies with the programming of applications. Such an environment inevitably results in a reduced independence (data independence) of the programs in terms of modifying

the meaning of data. Let us suppose, in the preceding example, that in order to assist the enterprise in coping with difficult economic times, a salary decrease of up to, but not exceeding, 10% could be considered in some cases. Such a change in policy would cause all application programs involving employee salaries to be rewritten in order to take into account the new meaning assigned to the data, as opposed to the situation where a data model is designed so that the incorporated integrity constraint could simply be altered:

e.g. MODIFY ASSERT ON UPDATE OF EMPLOYEE

NEW salary $\geq$ 90% OLD salary.

One must consider that controls over the data are also diminished; hence, the coding of a new and/or modified semantic constraint may be omitted for a given application program [INF 82].

Operations

This last part will briefly cover the operations that are allowed by the data model. The crucial point to be retained in terms of operations resides in determining whether or not a given model supports automatic navigation between the entities. Where automatic navigation is incorporated within the database management system, a user only needs to specify the end result desired in a given situation rather than the user having to describe how the system must access the data in order to provide the planned end result [COD 82].

3.3.1 The relational model

The relational model was first introduced by Codd in 1970, in a paper that is now known on a worldwide basis [COD 70]. Thus, that model is nearly ten years younger than the other two data models (network and hierarchical). The relational model has enjoyed a dazzling popularity among software manufacturers for database management systems as well as in universities and other research centers. In my opinion, such a widespread interest is mainly attributable to the conceptual simplicity and picturability of the model. However, given the extensive development delay required to design and commercialize such a product, commercial database management systems for large-scale computers based on the relational model first appeared on the market only in the late seventies. To name a few, here are some of the well-known systems in that category:

- SQL/SD (IBM, formely known as System-R [COD 81, AST 76]),
- INGRES (Relational Technology [STO 76, STO 80]),
- ORACLE (Relational Software Inc.),
- RAPPORT (Logica, United Kingdom).

In 1979, Codd proposed some improvements to the initial relational data model [COD 79]. One of the most significant contributions made in this paper was his discussion on the necessity to incorporate integrity constraints within the model, for the reasons previously stated.

Structures

The key characteristic of the relational model resides in that both the entities and their relationships are represented in a uniform manner by two-dimensional tables (flat files) referred to as RELATIONS where columns (or fields) are designated as the ATTRIBUTES of the relation and rows (or records) are called the TUPLES of the relation. The number of attributes of a relation determines its DEGREE while the number of tuples in that relation defines its CARDINALITY. A DOMAIN represents the complete set of those values which can be used for a given attribute; to illustrate, the set of all positive integers may make up a domain.

A database relation is basically a mathematical relation over a set of domains; as such, a database inherits the properties of a set, namely:

- Duplicate tuples are not allowed. In order to enforce that constraint, a relation must possess a user-defined PRIMARY KEY consisting of one or more concatenated attributes which uniquely identify each tuple in the relation.

- The ordering of the tuples within a relation is not significant; hence, such a property favours the use of set-oriented operations over the relations.

- Columns usually have a distinct name; thus, inthe context of a database relation, the ordering of the columns becomes

meaningless.

The relational model demands that all data be normalized. This basic requirement results in a model where repetitive or multilevel data cannot be represented as such:

e.g. DEPARTMENT (dept-name, floor-no, (employee-no, name, address)).

There are very few limitations inherent in the relational data model, other than the restriction indicated in the preceding paragraph. Hence, as illustrated in Figure 3-3, the following relations can be represented with equal ease by means of key propagation:

(a) One-to-one relationships (1-1),

(b) One-to-many relationships (1-n),

(c) Many-to-many relationships (m-n),

(d) Recursive relationships,

(e) N-ary relationships.

As a consequence of the great flexibility in representing relationships in the model, integrity constraints, as stressed in [COD 79], necessarily play a particularly important role in a relational-based database management system.

Integrity constraints

Large-scale database management systems of the relational type are still at a relatively fundamental stage of development in terms of the

FIGURE 3-3 RELATIONSHIPS IN THE RELATIONAL DATA MODEL

(a) One-to-one relationship between EMPLOYEE and NAME

EMPLOYEE

Employee-#	Name
900580	M. Bergeron
600238	R. Catafago
700615	J. Massy

(b) One-to-many relationship between DEPARTMENT and EMPLOYEE

EMPLOYEE

Employee-#	Name	Dept-name
900580	M. Bergeron	toy
600238	R. Catafago	toy
700615	J. Massy	kitchen

DEPARTMENT

Dept-name	Floor-#
toy	1
kitchen	7

(c) Many-to-many relationship between ARTICLE and KEYWORD

ARTICLE

Article-#	Title
C-826	DBMS for micros
M-432	Intel micros

KEYWORD

Key-id	Description
MI	Microcomputer
DB	Database

HAS-KEY

Article-#	Key-id
C-826	DB
C-826	MI
M-432	MI

(d) Recursive (one-to-many) relationship between MANAGER and EMPLOYEE

EMPLOYEE

Employee-#	Name	Manager-employee-#
900580	M. Bergeron	700615
600238	R. Catafago	700615
700615	J. Massy	400121

FIGURE 3-3 RELATIONSHIPS IN THE RELATIONAL DATA MODEL (CONTINUED)

(e) Tertiary relationship between SUPPLIER, PART and PROJECT

SUPPLIER

Supplier-#	Name
S1	IBM
S2	Juki

PART

Part-#	Type
P1	Disk
P2	Tape

PROJECT

Project-#	Location
J1	UQAM
J2	Concordia

PART-USED

Project -#	Part-#	Supplier-#
J1	P1	S1
J1	P1	S2
J1	P2	S1
J2	P2	S1

specification and enforcement of semantic constraints. System-R [AST 76] constitutes one of the most advanced programs in this specific area. Integrity constraints in a data model may be grouped in three categories according to their complexity level, as follows:

- Field constraints,
- Inter-field constraints,
- Inter-file constraints.

(1) Field constraints:

The simplest constraints, designated as field constraints, enable users to specify in a more precise fashion, the set of allowable values which a given attribute may draw from a specific domain. Such constraints are usually defined when a new relation and its attributes are created. In that category, one can find the constraints most frequently implemented in current database management systems such as minimum, maximum or range checks, UNIQUE or NONNULL fields as well as default values. For example, an enterprise establishes that each employee must have a distinct employee number and that his (her) salary cannot, under any circumstances, exceed $50,000:

```
e.g. CREATE EMPLOYEE
        employee-no   (numeric, 6, UNIQUE)
        salary        (dollar, 7, MAX 50000.00).
```

(2) Inter-field constraints:

The so-called inter-field constraints enable users to set more complex restrictions over values allowed for a specified attribute in connection with the contents of one or more attributes of the relation, itself included. The example presented at the very beginning of this section, concerning the comparaison of an employee's NEW salary with his (her) OLD salary, illustrates well the situation where the restriction on a given attribute pertains to the attribute itself. We will now present an example where the restriction on an attribute is defined in connection with another attribute belonging to the same relation. Let us say that the enterprise is organized so that a department cannot include more than ten employees:

e.g. ASSERT ON INSERTION OF EMPLOYEE

CHECK COUNT dept-name < 10.

(3) Inter-file constraints:

Also designated as referential constraints [INF 82], inter-file restrictions allow for the specification and enforcement of relationships between entities. Hence, they clearly take on particular importance in the relational model. For example, let us assume that any employee must belong to an already existing department:

e.g. ASSERT ON INSERTION OF EMPLOYEE.

CHECK dept-name IN DEPARTMENT.dept-name.

Operations

The relational data model supports set-oriented operations dealing with all the tuples of a relation at a time. Principal operations supported by the model may be grouped in two categories, namely:

(1) The conventional set operations: INTERSECT, UNION, DIFFERENCE;

(2) The relational operations: PROJECT, SELECT, JOIN and DIVIDE.

PROJECT and SELECT represent operations which pertain to only one relation at a time (single file operations) whereas JOIN and DIVIDE combine more than one relation between themselves over common attributes (multiple file operations). PROJECT permits the selection of certain attributes of a relation (vertical partition) whereas SELECT allows the choice of those tuples of a relation which meet the selection criteria specified by a user (horizontal partition). When performing a PROJECT, one would expect the system to automatically check for, and eliminate duplicate tuples, as is being done, for example, by INGRES. However, due to the high amount of overhead involved in such an operation, this constraint, although actually inherent in the relational model, may not always be automatically enforced within a given system; this may be true even in large-scale database management systems of the relational type. In the case of System-R [AST 76], the system searches for, and eliminates duplicate tuples solely upon user-specified request, i.e., by means of PROJECT UNIQUE rather than PROJECT.

To conclude our discussion of the relational data model, let us review the essential conditions which, according to [COD 82], must be met for a database management system to be properly classified as a relational-based system:

(1) The system must incorporate automatic navigation: thus, it must support tables without a user-described navigation link between them.

(2) The system must support set-oriented operations over the relations: hence, it must provide relational processing capability so that the transformations specified by SELECT, PROJECT and JOIN operators of the relational algebra can be specified without resorting to commands for iteraction and recursion.

Therefore, those two conditions will constitute the frame of reference used in Chapter V in connection with an analysis aimed at determining whether or not the labelling of some database management systems on the market as relational-based systems actually corresponds to an accurate classification for this type of software.

3.3.2 The network model

A large number of committees have worked on the study and standardization of the network data model @INF 83|. For instance, one may consider all releases published by the Data Base Task Group (DBTG) pursuant to Conferences on Data Systems Languages (CODASYL) held in 1971, 1973, 1978. The discussion that follows is based primarily on the recommendations published by the Data Base Task Group in 1971.

Let us also remember here that network-based management systems were first commercialized nearly ten years prior to the appearance on the market, towards the end of the last decade, of database management systems supported by the relational model. Indeed, one must bear in mind that huge resources had already been invested by that time, for the development, implementation and maintenance of systems and applications based on the network model.

Therefore, one may currently find numerous commercial database management systems based on the network model. Here are the most popular systems available on the market that fall in that category:

- IDMS (Cullinane),
- TOTAL (Cincom Systems Inc.),
- System 2000 (Intel),
- DMS 1000 (Sperry Univac).

Structures

Within the network model, the entities are not represented in the same manner as are the relationships existing between those entities. Thus, the entities are expressed by means of two-dimensional flat files (tables) referred to as RECORD TYPES whereas their relationships are represented by SET TYPES. A record type fundamentally differs from a database relation with regard to two basic properties, namely:

(1) The ordering of records within the record type is significant.

(2) Duplicate records are allowed within a given record type.

Therefore, the notion of a user-defined primary key found in the relational model (to enforce the elimination of duplicate tuples) becomes meaningless in the environment created by the network model.

As far as set types are concerned, the network model limits them to binary one-to-many relationships between an OWNER record type and a MEMBER record type, where the owner and member record types must correspond to distinct entities. Hence, as illustrated in Figure 3-4, only the following relationships can be represented directly:

(a) One-to-one relationships,

(b) Binary one-to-many-relationships.

Within that context, one can easily see that more complex relationships may not be expressed without resorting to what is referred to as dummy

record types. Figure 3-4 depicts how dummy record types come into play in the representation of the following relationships:

(c) Many-to-many relationships;

(d) Recursive relationships;

(e) N-ary relationships.

On the other hand, the network model, in contrast with the relational data model, supports the definition of repetitive groups as well as multilevel data:

e.g. RECORD DEPARTMENT

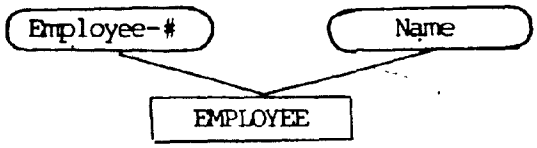
dept-name	character (30)
floor-no	numeric (1)
employee-info	OCCURS MULTIPLE TIMES
employee-no	numeric (6)
name	character (30)
address	HAS SUBSTRUCTURE
street	character (30)
city	character (30)
post-code	numeric (6).

Integrity constraints

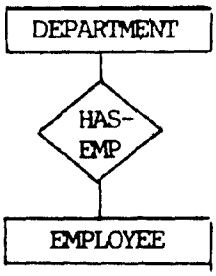
At the logical level of a network model, the data definition language (DDL) put forth by the DBTG group includes statemen s aimed at describing semantic constraints that must be enforced by the system, such as:

FIGURE 3-4 RELATIONSHIPS IN THE NETWORK MODEL

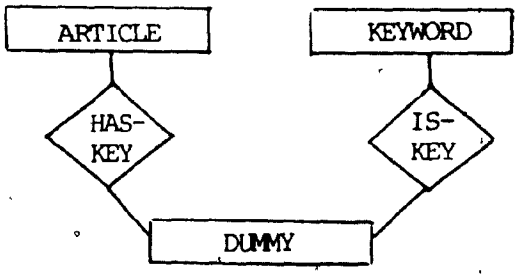
(a) One-to-one relationship between EMPLOYEE and NAME



(b) One-to-many relationship between DEPARTMENT and EMPLOYEE



(c) Many-to-many relationship between ARTICLE and KEYWORD



(d) Recursive (one-to-many) relationship between EMPLOYEE and MANAGER

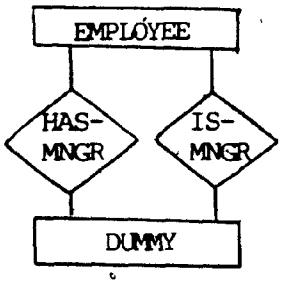
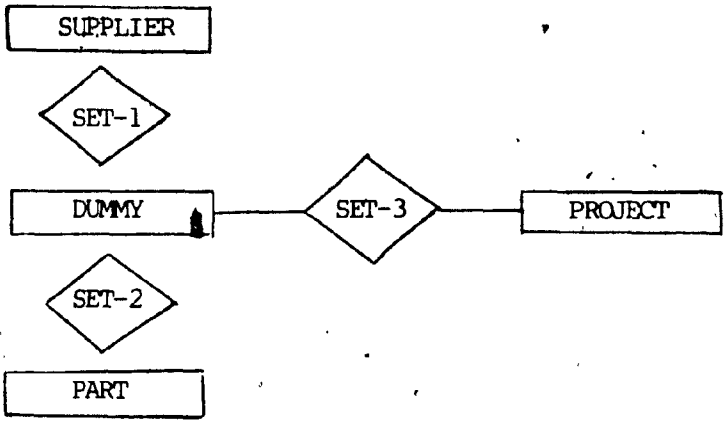


FIGURE 3-4 RELATIONSHIPS IN THE NETWORK MODEL (CONTINUED)

(e) Tertiary relationship between SUPPLIER, PART and PROJECT



Legend: Data item

Record type

Set type

(1) Field constraints:

e.g. DUPLICATES ARE NOT ALLOWED;

(2) Inter-file constraints:

e.g. INSERTION IS AUTOMATIC

SET SELECTION THRU CURRENT OF set type;

(3) User-defined constraints:

e.g. ON conditions

where conditions represent any set of user-defined procedures invoked before or after the use of commands such as STORE.

Operations

The data manipulation language (DML) of a network model as defined by the DBTG group is a one-record-at-a-time oriented language. Thus, the notion designated as current run-unit, omnipresent in such a language, applies to the record types and set types as well as to records or sets:

e.g. FIND NEXT record type RECORD IN CURRENT set name SET,

FIND CURRENT OF record type RECORD,

GET record type.

The crucial point to be retained in terms of operations within a system based on the network data model resides in that a user needs to describe the access paths through the records and sets to be followed by the

system in order to obtain the end result desired in a given situation:

e.g. SET OWNER record type RECORD THRU set name SET.

Hence, a network-based system does not support automatic navigation: record types interrelate by means of a user-defined navigation link between them. One can see that having to initiate complex access paths through the database, using a data manipulation language (DML) consisting of a set of one-record-at-a-time procedure calls may easily become a far too demanding task for nontechnical users. In respect to such a limitation inherent in the network model, software products have been designed and commercialized to enable a user with limited knowledge of the network navigational paths to access the database through the use of an English-like system/user interface. This area will be covered in greater depth in Chapters V through VII.

In view of the fundamental differences in characteristics peculiar to the relational and network models, researchers, manufacturers as well as users have made repeated and laborious attempts to assess which type of model actually surpasses the other. This question still remains subject to lively discussions in the current literature. Given the importance of the issue involved here, let us defer its discussion to a separate chapter of this report. Thus, Chapter VII will present our evaluation of the two data models as applied, in particular, to the microworld.

3.3.3 The hierarchical model

The hierarchical model actually stands as the forefather of the two models previously discussed. IMS, developed by IBM, constitutes the best known database management system based on that type of model. The widespread use of IMS in the industry as well as the very large sums which were allocated to the development of application programs based on that system, warrant a mention of the hierarchical model here. Indeed, the data model of the hierarchical type continues to enjoy a status somewhat comparable to that of the relational and network models even though the former model, as may be seen in the description of its structures presented below, can actually be regarded as a subset of the network model, with more restrictive modeling capabilities. Hence, our discussion of the hierarchical model will remain brief, addressing only the structures involved in it; the integrity constraints and operations in the model merely follow suit.

Structures

The hierarchical data model applies the same notions as those found in the network model in terms of the divergent representation of the entities and the relationships existing between those entities. Thus, one concludes to the same fundamental limitations inherent in the model as those previously identified in the preceding subsection (3.3.2), which may be summarized as follows:

(1) Set types are limited to the representation of two types of relationships, namely:

- Binary relationships;
- One-to-many relationships.

(2) The OWNER and MEMBER record types must correspond to distinct record types.

Over and above that, the hierarchical model presents an additional significant limitation in that a given record type can only be assigned one, and only one owner record type. In such a context, a hierarchical model can therefore be conceptually visualized as a tree structure.

3.4 The functions of a database management system

The survey, analysis and evaluation of a software program calls for a two-step process; firstly, one must compare what a program actually does perform as opposed to what it should be doing and, in a second step, formulate an opinion on the manner in which the program does perform. The first part involves a study of the program functions whereas the second step derives from an investigation of the methods being used to implement those functions; such methods are hereafter referred to under the term features of a database management system.

Given that the above mentioned approach was actually adopted in Chapters V and VI of this report dealing with, respectively, a survey and an overall critical analysis of database management systems for microcomputers, the objective sought in the current section is to define and, more importantly, to classify the functions and features of a database management system in the most schematic manner feasible in such a context.

With regard to the functions of a database management system, Kim introduced in a survey dealing with relational database systems [KIM 79], the following listing of nine functions which should be performed by a database management system of the relational type. As will be later demonstrated, such a listing equally applies to all types of database management systems.

(1) An interface for a high-level non-procedural data language which provides the following capabilities for both application programmers and nontechnical users: query, data manipulation, data definition and data control facilities.

(2) Efficient file structures in which to store the database along with efficient access methods to the stored database.

(3) An efficient optimizer to help meet the response-time requirements of terminal users.

(4) User views and snapshots of the stored database.

(5) Integrity control aimed at the validation of semantic constraints on the database during data manipulation, as well as the rejection of offending data manipulation statements.

(6) Concurrency control to achieve a synchronization of simultaneous updates to a database shared by multiple users.

(7) Selective access control to ensure that access privileges attributed to a specific database user are effectively unacessible to other users.

(8) Recovery from both soft and hard crashes.

(9) A report generator for a highly stylized display of the results of

interactions against the database and application-oriented computational facilities such as statistical analysis.

The following tenth function was added, in [INF 82], to the already impressive listing presented by Kim:

(10) Adequate tools for the database administrator to find out what is going on in order to change and improve database design and/or storage when necessary.

On the premise that current database management systems based on the network or hierarchical data model call, for the most part, for the user to describe the required navigational set of instructions to the system, functions (1) and (3) listed above represent the only functions referring mainly to a database management system of the relational type. However, those functions also apply in cases where some important navigation decisions rest within recently commercialized database management systems of the network type, through the use of a relational-like high-level language overriding the lower level data manipulation language (DML).

In order to develop the framework most suitable for a subsequent survey and analysis of database management systems for microcomputers, the ten functions listed in the preceding pages have been first grouped under five areas, and then divided into fourteen categories. Figure 3-5 illustrates such a classification.

FIGURE 3-5 FUNCTIONS OF A DATABASE MANAGEMENT SYSTEM

Areas	FUNCTIONS	Features
General	USER INTERFACE	Command entry method (menu, typed out) Host programming language interface Build-in programming language Macro commands Command error trapping On-line help command Documentation
Data definition	DATA DESCRIPTION	Create schema Create views or subschemas Specify field types
	DATA STORAGE DESCRIPTION	Specify secondary indexes Specify file structures Specify data clusters
	DATA RESTRUCTURING	Add fields Delete fields Modify field type or size Restructuring mode (static, dynamic)
	DATA REORGANIZATION	Modify secondary indexes Modify file structures Modify data clusters Reorganization mode (dynamic, automatic)
Data manipulation	DATA ENTRY	Add records Screen generator
	DATA UPDATE	One record updates Automatic multiple-record updates System triggered updates Full-screen editing

FIGURE 3-5 FUNCTIONS OF A DATABASE MANAGEMENT SYSTEM (CONTINUED)

Areas	FUNCTIONS	Features
Data control	DATA RETRIEVAL/QUERY	Query language type Snapshots of the stored database Multiple field conditions Boolean operators (AND, OR, IN) Relational operators (=, <, >) Single file operations Multiple file operations
	REPORT GENERATOR	Formatting facilities Mathematical and statistical facilities
	INTEGRITY CONSTRAINTS	Field constraints Inter-field constraints Inter-file constraints
	ACCESS CONTROL	Data granularity (file record) Access privileges
	CONCURRENCY CONTROL	Transaction definition (command, user-spec) Lock granularity (file, record) Deadlock detection and recovery
	DATA BACK-UP AND RECOVERY	System back-up Transaction commitment Transaction logging
Data storage	PHYSICAL STORAGE	File structures (B-tree, hash, ISAM) Access methods (sequential, direct)

With respect to the features of a database management system, the classification of available features provided under the third column in Figure 3-5 was developed through the aggregation and amalgamation of various categorizations proposed in the recent literature, mostly in certain articles dealing with surveys of database management systems [BAR 81, BON 84, DIE 81, KRA 81, KRU 83].

Rather than to proceed to a description of each of the forty-nine features included in Figure 3-5, a rather lengthy and potentially redundant exercice in view of subsequent chapters of the present report, we have chosen to defer such an endeavour to Chapter V, where the classification of system functions and features introduced in this figure will serve as the frame of reference for a survey and critical analysis of database management systems for microcomputers.

To conclude, let us underline the fact that the incorporation of some of the functions enumerated above to the large-scale commercial database management systems is, indeed, a very recent occurrence. In this line of thought, the following cases involving well-known database management systems can be mentioned [COD 81]:

DYNAMIC DATA RESTRUCTURING:

The EXPAND TABLE command of System-R which enables users to add a new attribute to an existing relation without having to perform a batch off-load of the database.

DYNAMIC DATA REORGANIZATION:

The MODIFY command of INGRES which permits changes in the storage structure of a base relation or a secondary index without having to off-load the database.

CONCURRENCY CONTROL:

The BEGIN-TRANSACTION and END-TRANSACTION delimiters of System-R allowing the user to specify the starting and ending points of an atomic or indivisible transaction which does not have to be limited to a single database command.

Furthermore, certain functions still remain under-developed to date, even for the largest database management systems currently available on the market; this is the case principally in areas concerning the incorporation of INTEGRITY CONSTRAINTS and AUTOMATIC DATA REORGANIZATION within database management systems.

IV. THE OBJECTIVES OF DATABASE MANAGEMENT SYSTEMS FOR MICROCOMPUTERS

The preceding chapters endeavoured to answer the two most fundamental questions in a study of database management systems for microcomputers: what is a microcomputer (Chapter II), and what are a database and a database management system (Chapter III). Let us now initiate the amalgamation of these two technologies in computer science and attempt to determine, in this chapter, what are the objectives of a database management system for microcomputers. Three basic issues are raised by such a question: we must first identify the actual and potential users of microcomputers; then proceed to determine the needs among microcomputer users for a database management system; and finally assess how a database management system for microcomputers meets identified needs.

Within this context, microcomputer users not in need of data processing capabilities must be ignored. Our discussion therefore completely excludes, at the onset, the three following groups of microcomputer uses:

- Industrial process monitors,
- Recreational uses (video-games),
- Educational uses (didactical products).

Benhaset and Goldstein introduced a first categorization of minicomputer users in [BEN 77], distinguishing two major user groups: the small businesses and the large organizations. Both these categories apply to

the microworld of the eighties; let us remember here that the high end microcomputers being sold today can indeed be compared to minicomputers in terms of available computing power.

According to Benhaset, a small business essentially pursues the same basic goals in acquiring a minicomputer as those sought by a large organization using a larger-scale computer; the fact that the former selects a minicomputer derives from the computer needs and, more importantly, the financial resources of a small business being limited, in relative terms.

For the large organization, interconnected mini or microcomputers may represent an attractive alternative, or adjunct, to a large central computer system for several well-known reasons such as, for example:

- Interconnected small computers result in lower total hardware costs, transmission costs, etc., thereby becoming a less expensive way to process data.

- A network of small computers allows more flexibility for future growth; hence, adding new nodes to an existing network of small computers is easier and cheaper than switching to an upgraded version of a very large central computer system.

- The environment created by interconnected small computers makes it easier to cope with system (component) failures.

. A network of small computers naturally fits the geographical distribution of large organizations.

To the two minicomputer user categories which also apply to today's microworld, one must add a third category of microcomputer users, composed of individuals, the latter group being users of microcomputers only. The availability of low cost microcomputer systems enabled manufacturers and vendors to penetrate two new segments of the market with these machines, namely:

(1) The home computer market, for uses other than recreation (such as: budget keeping, word processing, appointment keeping, etc.);

(2) The market segment comprised of the employees of large organizations, for individual professional uses (such as: word processing, electronic mail, worksheets, graphs, etc.).

Microcomputers used in those two segments of the market are often designated as workstations. Having pinpointed the identity of the users of microcomputers for data processing purposes, let us now consider what are the actual needs among those users in terms of database management systems and applications. This question will be addressed for each of the three microcomputer user groups discussed above:

. For individuals (Section 4.1),

. For small businesses (Section 4.2),

. For large organizations (Section 4.3).

4.1 For individuals

Data processing needs among individual users often remain relatively simple and are generally translated into the implementation of a manual card index (cardex) system on a computer medium. The main purposes sought by users in the installation of such a system can be summarized as follows:

(1) To generate reports in various formats with ease through the use of different selection and sort fields;

(2) To obtain on-line information rapidly and with minimum effort;

(3) To facilitate data manipulation, i.e., the input of new data and the elimination of obsolete data;

(4) To save on physical storage space by converting paper files to computer files.

A data management system would perfectly suit the needs of those users by providing for the following functions:

(1) File management for on-line inquiry,
(2) Report generator,
(3) Interactive data definition and manipulation language,
(4) Query language.

Any system capable of manually finding its way through a cardex stands as a perfect candidate application for data management systems of this type. For instance, to name only a few:

- A stamp collection catalog,
- An appointment register,
- An inventory system,
- A bibliography,
- Any list of individuals, goods, etc.

Without such a data management system, a microcomputer owner would be required to either program the desired application or purchase a package. A vast majority of individual users may not be tempted by either alternative. As far as the first option is concerned, one must consider that this group is composed mostly of nontechnical users. When this is the case, the time an individual would have to invest in order to learn a programming language and to perform the design, coding, debugging and testing of his (her) application programs, appears to be out of all proportion to his (her) needs. Furthermore, the individual must acquire a translator (interpreter or compiler) in order to program in a level language higher than BASIC, the latter being generally supported on microcomputers. The financial resources of individual users must also be taken into account here. To illustrate, a potential user may be rather surprised to find out, by means of an advertisement dealing with new products published in the February 1984 issued of IEEE Micro, that a Pascal-286 translator for iAPX-286 (a 16-bit microprocessor) has been announced by Intel, at the price of $3,900.00.

With respect to the second alternative, an individual user must start on the hunt for application packages not necessarily available on the market. Where desired packages can be found, the user has to bear the disadvantages associated with such products, namely:

- Packages can never meet the specific needs of a particular user;
- These packages generally do not allow for future maintenance;
- Packages are expensive (to my knowledge, the average cost per microcomputer application is equivalent to approximately 10% of the initial hardware equipment cost).

One can currently find on the market numerous data management systems that incorporate the functions required to meet the needs of most individuals. Often marketed under the label database management systems, such systems actually correspond more closely to file management systems than to database management systems, in accordance with the descriptions provided in Chapter III (Sections 3.1 and 3.2, respectively). Nevertheless, they inherit certain characteristics of database management systems in providing the following functions:

- A high-level interactive query language,
- A high-level data definition and maintenance language,
- A report generator language, etc.

4.2 For small businesses

The number of users making up the small business group was estimated by Wickham in [WIC 78] at over five and a half million, in the United States alone. This figure includes approximately three million small business organizations, with a number of employees ranging anywhere from 1 to 100, as well as two and a half million self-employed individuals. For instance, a lawyer operating as a sole practitioner falls under the latter category.

One must consider here that, in many cases, a microcomputer may be the only stand-alone computer system a small business can afford. As such, the microcomputer must provide the small organization with basically the same facilities as those made available to a larger enterprise by a larger computer. Thus, the major divergences between these two user groups reside in the volume of data having to be stored and processed by the computer, as well as the number of users having to get access to the system. In such a context, a database management system designed for small businesses should allow for the same functions as those provided by a large-scale database management system to serve the same purposes.

However, one must recognize the impact of a limited number of employees getting access to the system. In some instances, a sole employeee is authorized to use the system. In other cases, more than one employee is empowered to make use of a system, but the organizational environment

does not provide sufficient grounds to warrant a simultaneous access to the system by authorized users. In both these situations, a relatively inexpensive single-user database management system adequately meets user needs. As will be covered in Section 5.2, numerous single-user database management systems are currently available on the market.

In cases where circumstances surrounding a small business give justification for the concurrent use of the system by authorized employees, the organization must acquire a multiuser database management system. Such a system provides for more than one employee to simultaneously get access to it. As will be seen in Section 5.3, relatively few multiuser database management systems for microcomputers have, at this point in time, reached the commercial stage. Section 5.3 will not only deal with a survey of software products of this type currently marketed, it will also emphasize the various configurations peculiar to a multiuser database management system, namely:

- A time-sharing database management system,
- A database server,
- A distributed database system over a local area network.

Applications required by the small business user group mostly pertain to their accounting systems. Needless to say, specific user needs in terms of the full range of accounting applications vary widely from one small organization to the next. Major applications falling within such a range are listed below, in decreasing order of frequency of use:

- Cash disbursement system,
- Cash receipt system,

- General ledger system,
- Payroll system,
- Accounts receivable system,
- Accounts payable system,
- Inventory system,
- Cash flow system,
- Fixed assets system.

Over and above accounting-related programs, more refined applications have been developed to fill the needs of business users. Found mostly in larger organizations due to the financial limitations inherent to small businesses, such applications are nonetheless also being used by the latter group, although at extremely divergent levels of complexity and/or frequency. Here is a partial listing of potential applications:

- Order systems,
- Personnel systems,
- Project control systems,
- Management supporting systems (such as: budgets, forecasts, cost/benefit analysis, etc.),
- Production control systems.

4.3 For large organizations

For a large organization, interconnected mini or microcomputers often appear as an attractive alternative, or adjunct to a large central computer system, for reasons which were discussed in our introduction to this chapter. The interconnection of two or more computers for a large enterprise, within the environment of databases, may follow at least two different approaches which, in the literature, are being most often designated as follows:

(1) Back-end database systems,

(2) Geographically distributed database systems over a long-distance communication network.

Figure 4-1 graphically illustrates the notions applied in a back-end database system. One can see that the database management functions are off-loaded from the mainframe host computer to a smaller back-end database machine dedicated to such functions. The host computer retains solely the function pertaining to the interface with users/application programs. All database requests are directly passed on, for processing, to the back-end database system which in turn, transmits processing results to the host computer for distribution to the request initiators.

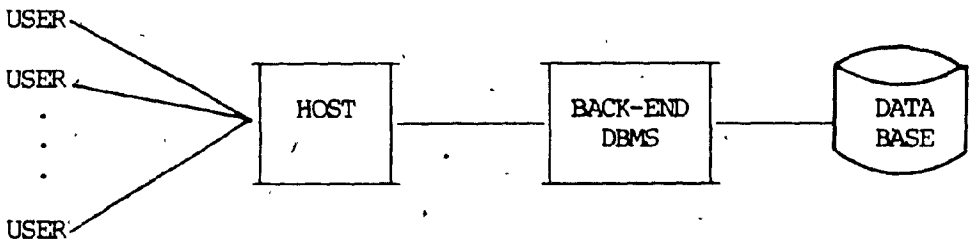
Maryanski mentioned, in a survey of back-end database systems published in 1980 [MAR 80], that to his knowledge, only experimental prototypes

of back-end database systems existed (no such system having yet been marketed) at the time. However, at least two systems of this type have been subsequently introduced on the market, namely:

- ADABAS (Software AG),
- iDBP (Intel).

The Intel database processor iDBP 86/440 takes on particular interest in the present study of database management systems for microcomputers since this product is based on the iAPX-286 16-bit microprocessor developed by Intel.

FIGURE 4-1 BACK-END DATABASE SYSTEMS



A back-end database system is primarily aimed at that market segment composed of large organizations with heavy data management activities wishing to off-load such functions to a dedicated back-end database machine in order to free the host computer, thereby allowing the latter to devote its own time to less specialized tasks.

The major advantages of a back-end database system, according to

Maryanski, may be summarized as follows:

- The implementation of such a system requires less expensive hardware than would be necessitated by an equivalent mainframe update.

- The system allows for upgrades in smaller steps.

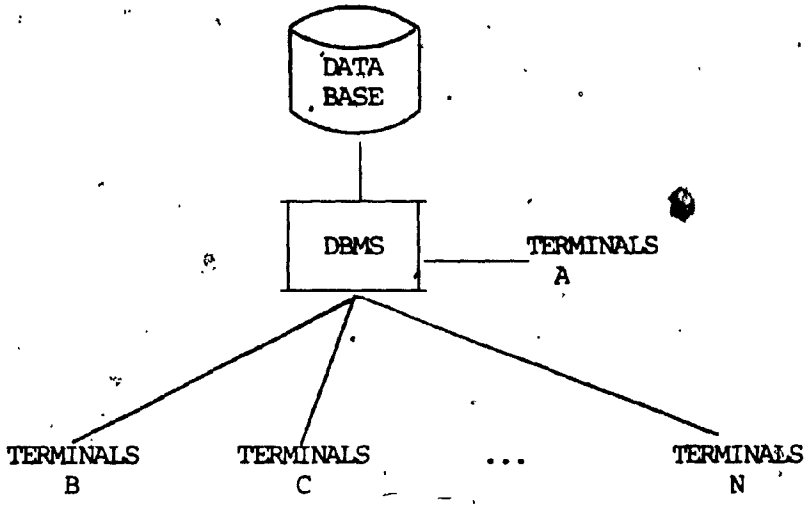
- The incorporation of a back-end database system may result in overall system performance gains.

- A back-end database system constitutes a first step towards distributed data processing and distributed databases.

The second type of configuration involving more than one computer for large organizations, distributed database systems, are particularly advantageous to enterprises serving geographically distributed users. Figure 4-2 depicts the basic concepts being applied in the distributed database approach. As may be seen from the first two graphs, such an approach replaces a centralized database management system with remote distributed access (illustrated in Figure 4-2 (a)) by a set of smaller independent nodes that are interconnected over a long-distance communication network (pictured in Figure 4-2 (b)). In such an environment, both the database and the database management system program are distributed over at least two computers. That characteristic constitutes the fundamental basis on which one can distinguish a distributed database system from what may be designated

FIGURE 4-2 DISTRIBUTED DATABASE SYSTEMS

(a) Distributed access approach



(b) Distributed database approach

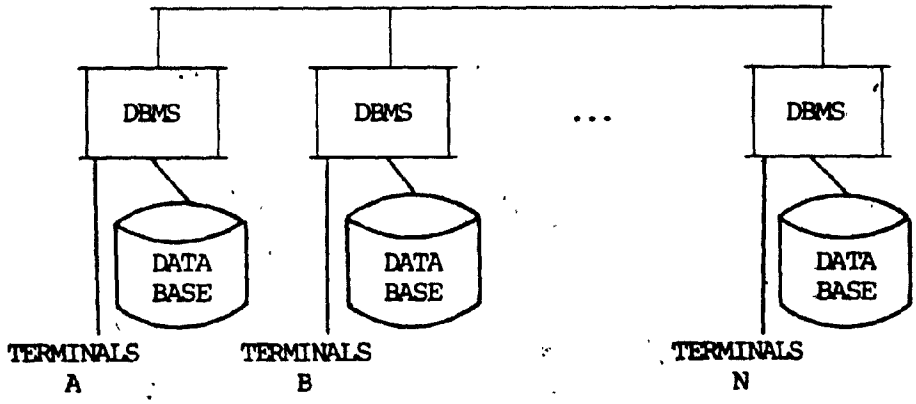
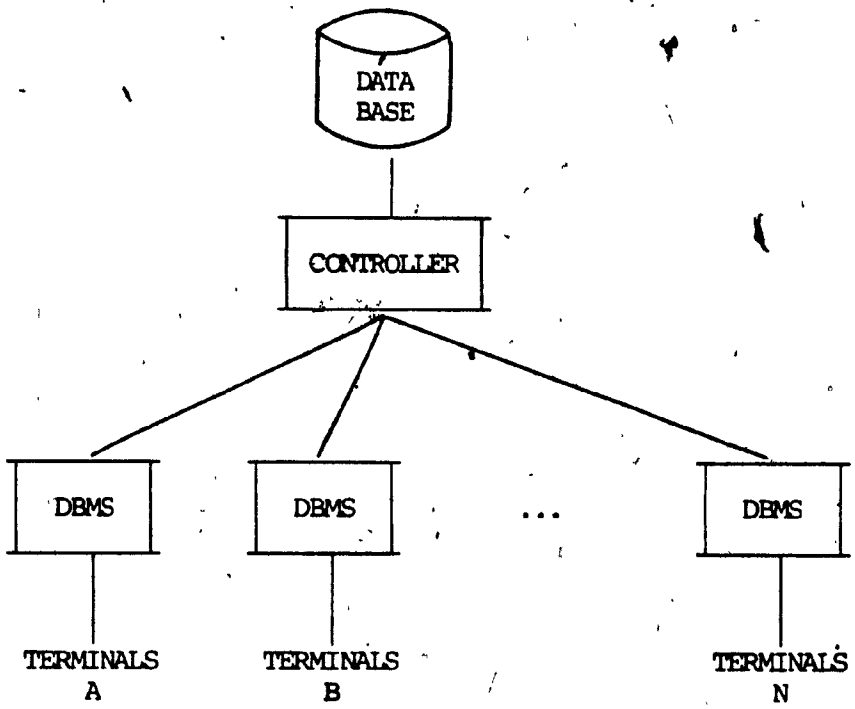


FIGURE 4-2 DISTRIBUTED DATABASE SYSTEMS (CONTINUED)

(c) Distributed database processing approach



as a distributed database processing system. In the latter case, as illustrated in Figure 4-2 (c), only the database management system program is distributed over two or more computers; hence, the database remains centralized at a single node.

The advantages of a distributed database system over a conventional centralized database system derive from five main improvements carried out in those systems, namely:

- An improved access time to the data pursuant to the principle of data locality. The designers of Distributed INGRES, for example, have set forth as a fundamental hypothesis, that at least 95% of all accesses are to local data [STO 79].

- A decrease in communication costs which may, in turn, potentially translate into a decrease in total on-going operation costs also attributable to the principle of data locality.

- A decentralized control of the data, resulting in greater independence for current users of the data.

- An increase in the total system reliability in that a failure of a component of the system does not necessarily imply a total system failure.

- A greater flexibility for future growth (decline) in that the system favours an incremental upward (downward) scaling of total

database capacity.

There are, however, certain problems associated with geographically distributed database systems over a long-distance communication network. Such drawbacks are essentially the same as those found in distributed database systems over a local area network, which will be discussed in Section 5.3 dealing with multiuser database management systems for microcomputers.

To conclude this particular subject, it should be noted that, to my knowledge, no operational (versus experimental) geographically distributed database management systems where the independent nodes consist of microcomputers have yet reached the stage of being commercially available.

Having determined, in the current chapter, what are the objectives of database management systems for microcomputers, a consideration of those objectives becomes fundamental in all subsequent chapters of this report. Thus, one must always keep in mind the actual needs peculiar to each of the three distinct microcomputer user groups (individuals, small businesses and large organizations) in terms of database management systems and applications.

V. A SURVEY OF DATABASE MANAGEMENT SYSTEMS FOR MICROCOMPUTERS

We have now reached the point where the complete framework within which we have selected to perform our study of database management systems for microcomputers has been set up. Indeed, Chapters II and III enabled us to establish, describe and analyze, in a manner structured so as to meet our stated goal, the fundamentals of microcomputers and database management systems, respectively. Furthermore, Chapter IV permitted us to determine the objectives of database management systems for microcomputers through a categorization of the current as well as potential uses for such tools. We are therefore in a position to proceed to the actual study of database management systems within the microworld context. Such a study will comprise the three following major steps:

- A survey of database management systems currently available on the microcomputer market (Chapter V);

- An overall evaluation of database management systems for the microworld with regard to both the current state of the art, as well as future trends in that field (Chapter VI);

- A comparative analysis of the two principal data models (relational and network) as applied to databases on microcomputers (Chapter VII).

In this line of thought, the present chapter consists of a survey of database management system products currently available on the microcomputer market. At the very onset, our research in this area led us to distinguish three principal types among database management systems marketed in the microworld, namely:

- File management systems,
- Single-user database management systems,
- Multiuser database management systems.

Given that each category of database management systems mentioned above presents very distinct fundamental characteristics, the present chapter is divided into three sections, each one being devoted to the study of a given type of microcomputer database management system.

In order to deal with each of the three system groupings in a consistent and uniform manner, all three types of database management systems for microcomputers will be presented and examined in terms of the categorization of the functions and features of a database management system which was developed in Section 3.4. Hence, such an approach will provide us with a comparative basis for the study of the different system products conducted in the current chapter. Furthermore, that same comparative basis will be used in Chapter VI, where we will proceed to an overall evaluation of the database management products being surveyed here.

5.1 File management systems

Given the ever-increasing number of software products labelled as database management systems for microcomputers introduced on the market over the past five-year period, one may now find a few surveys of systems carrying such a label being reported on in the current literature. Reading through some of those surveys, one realizes that the vast majority of programs sold as database management systems for microcomputers actually represent file management systems rather than database management systems. A great number of those systems can access only one data file at the time: thus, all the records that can be used at a time belong to just one file, and that file constitutes the database. As explained in Chapter III, such systems should therefore be described as file management systems. Indeed, the fundamental characteristic of a database management system is the ability of the software to support a data model to define multiple entities and relationships between entities, thereby enabling a user to access more than one data file at a time.

The incorrect classification of file managers for microcomputers as database management systems can be repeatedly found among software manufacturers and vendors, as well as in advertisements and in articles covering this type of software. The November 1981 issue of BYTE Magazine [BAR 81], for example, presented a survey on database management systems for microcomputers, where twenty different programs marketed as database management systems at the time were analyzed. A

close review of that paper reveals that sixteen of the twenty systems discussed therein, actually represented file managers rather than database management systems. Another example can be found in an article published in 1982 [DOR 82], where all five database management systems discussed by the writer turn out to be file management systems. Furthermore, the confusion remains present in today's literature. For instance, eight systems described as database management systems were reviewed in the June to August 1984 issues of Microcomputing Magazine [BRY 84] although, in view of the fundamental distinction mentioned above, only four out of the eight systems could accurately be classified as database management systems.

Nevertheless, one can perceive in the 1984 literature, the emergence of an attempt to clarify the varioustypes of database management systems for microcomputers. To illustrate, the October 1984 issue of BYTE Magazine specifically devoted to databases presented, as an introduction to this subject, an article [KRA 84] which differentiates and describes seven types of databases for microcomputers, namely:

- File management systems,
- Relational databases,
- Hierarchical databases,
- Network databases,
- Free-format databases,
- Multiuser databases,
- Database machines.

Furthermore, the catalog of database management systems published in

that same issue [BON 84] listed forty-seven systems currently available on the market, grouping nineteen of them as file manager programs, twenty-two as relational database programs and six as multiuser system programs.

Let us now present a brief summary of findings of some of the published surveys, restricting ourselves, for the purpose of the current section, to systems which can access only one data file at a time, i.e., to file management systems.

Prices of file managers for microcomputers currently range from $25.00 to $600.00. Given the extremely rapid growth in the sales volume of these systems over their five years of existence, one would have expected the significant economies of scale realized by manufacturers and vendors to be reflected by a drop in selling prices. However, the average price of a file management system for microcomputers has actually increased from a level of $150.00 in 1981 to approximately $300.00 in 1984. This increase is mostly attributable to refinements introduced in the software itself during that period. Thus, file managers being sold today generally provide a greater number of features than their predecessors. This trend is reflected, for instance, in the number of programs available at prices below the $100.00 mark: 56% of the sixteen file management systems surveyed in 1981 belonged in that price range while only 21% of the nineteen file managers included in the 1984 survey were available at those low prices.

When comparing the survey published in late 1984 [BON 84] to the paper

presented three years earlier in the same magazine [BAR 81], one is surprised to notice that none of the nineteen file management systems shown in the 1984 survey actually appeared on the listing of sixteen file managers dating back to 1981. Thus, although different sampling methods account for this finding to some degree, one may be allowed to conclude that file management systems for microcomputers have had, at least to date, a very short life span. This rapid turnover can possibly be attributed to the agressive competition going on among microcomputer software manufacturers. Indeed, not only have upgraded versions of existing systems continuously been introduced in the marketplace over the past five years, new packages are also popping up almost every day.

We will review the major functions provided in the most advanced file management systems designed for the microcomputer environment. To assist in this task, Figure 5-1 summarizes the features and characteristics of the five most expensive file managers for microcomputers surveyed in [BAR 81]. The packages selected for a close review and analysis are the following:

- TIM (Innovative Software),
- The Microconductor (Microcomputer Technology Inc.),
- Jinsam 2.0 (Jini Micro-Systems Inc.),
- Profile II (Radio Shack),
- Micro Manager (Des Moines Computer).

As mentioned in the introduction to the current chapter, the comparative features used in the survey reported in [BAR 81] have been rearranged

FIGURE 5-1 FUNCTIONS AND FEATURES OF FILE MANAGEMENT SYSTEMS

FUNCTIONS and features	TIM	The Micro-conductor	Jinsam 2.0	Profile II	Micro Manager
VENDOR	Innovative Software	Microcomputer Technology Inc.	Jini Micro-Systems Inc.	Radio Shack	Des Moines Computer
GENERAL FEATURES					
Runs on	Any computer with CP/M	Apple,TRS-80	Commodore PET	TRS-80	Ohio Scientific
5-inch Disks	Yes	Yes	Yes	-	-
8-inch Disks	Yes	-	-	Yes	Yes
Hard Disks	Yes	-	-	-	Yes
Minimum Memory (in bytes)	48K	32K	24K	32K	48K
Written in	BASIC	-	BASIC	Assembler	-
Price	$400.	$400.	$195.	$179.	$249.
Extension Packages/Price	Wordstar	-	Mathpack/$40 Statpack/$50 Wordpack/$50 Label Gen/$50	Scripsit/$300	-
USER INTERFACE					
Error Recovery	Yes	?	Yes	?	?
Command Entry Method	Menu	Menu & Type	Menu	Menu	Menu
Programming Language Interface	Yes	No	Pkg Avail	No	No
Help Command	Yes	Yes	Yes	No	No
Macro Commands Available	No	Yes	No	No	No

FIGURE 5-1 FUNCTIONS AND FEATURES OF FILE MANAGEMENT SYSTEMS (CONTINUED)

FUNCTIONS and features	TIM	The Micro-conductor	Jinsam 2.0	Profile II	Micro Manager
DATA DESCRIPTION					
Field Types Defined:					
Alpha	No	Yes	No	Yes	No
Numeric	Yes	No	No	Yes	No
Date	Yes	No	No	No	No
Dollar	Yes	Yes	No	No	No
Variable Length	No	No	No	No	No
Calculated	Yes	No	No	No	No
Decimal	No	No	No	Yes	No
Maximum Number of Fields	24	20	255	-	75
DATA RESTRUCTURING					
Revision Allowed	Yes	No	Yes	No	No
DATA ENTRY					
User-designed Screens	No	No	No	Yes	No
DATA UPDATE					
Automatic Multirecord Updates	No	No	No	No	Yes
System Triggered Updates	Yes	Yes	No	No	No
Predefined Updates	No	Yes	No	No	No
Update Audit Trail	No	No	No	No	No
DATA RETRIEVAL/QUERY					
Multi Field Conditions	Yes	Yes	Yes	Yes	No
Complex Nested Conditions	Yes	No	Yes	No	No
Range Conditions	Yes	No	Yes	Yes	No
Substring	Yes	Yes	Yes	Yes	Yes
Null Value	Yes	No	No	No	No
Predefined Queries	Yes	No	Yes	Yes	No
Modifiable Queries	Yes	No	No	No	No
Subfiles (snapshots)	Yes	No	No	No	No

FIGURE 5-1 FUNCTIONS AND FEATURES OF FILE MANAGEMENT SYSTEMS (CONTINUED)

FUNCTIONS and features	TIM	The Micro-conductor	Jinsam 2.0	Profile II	Micro Manager
REPORT GENERATOR					
Can be written to file	No	Yes	No	No	No
Format Predifined	Yes	Yes	Yes	Yes	No
Title	Yes	Yes	Yes	Yes	Yes
Heading	Yes	Yes	Yes	Yes	Yes
Footing	No	No	No	No	No
Column Widths	Yes	No	Yes	No	Yes
Text Inserts	Yes	No	No	No	No
Margin Justification	Yes	No	No	No	No
Margin	Yes	No	Yes	Yes	Yes
Sample	Yes	Yes	Yes	No	No
# Lines Skipped	Yes	No	Yes	No	Yes
# Char. Between Records	Yes	No	Yes	No	Yes
# Records Across Page	Yes	No	Yes	Yes	No
# Lines per record	Yes	No	No	Yes	Yes
Summary Breakpoints	Yes	Yes	No	Yes	No
Sums	Yes	Yes	Pkg Avail	Yes	No
Means	Yes	No	Pkg Avail	No	No
Minimums	Yes	Yes	Pkg Avail	No	No
Maximums	Yes	Yes	Pkg Avail	No	No
Roots	No	No	Pkg Avail	No	No
Weights	No	No	Pkg Avail	No	No
Count	Yes	No	Yes	No	No
Statistics	Yes	No	Pkg Avail	No	No
Average	Yes	No	Pkg Avail	No	No
User Defined	Yes	No	No	No	No
Two Columns Operations	Yes	No	Pkg Avail	No	No
Record Numbering	?	No	Yes	Yes	No
ACCESS CONTROL					
Levels of Security	1	No	3	4	1

to fit the categorization of database management system functions which was developed in Section 3.4. In accordance with such a classification, the selected file management systems performed, somehow or other, the following functions of a database management system: USER INTERFACE, DATA DESCRIPTION, DATA RESTRUCTURING, DATA ENTRY, DATA UPDATE, DATA RETRIEVAL/QUERY, REPORT GENERATOR and, finally, ACCESS CONTROL. Following a brief discussion of some general features of the five file management systems enumerated above, we will therefore proceed to analyze all five systems in terms of such functions.

GENERAL FEATURES:

General features include hardware requirements, prices, available extension packages and language in which the software is written.

Direct access disks are required for all five systems; thus, these packages cannot be used on low price home computer systems supporting cassettes only. Two of the five systems (TIM and Micro Manager) support hard disks while only one package, TIM, provides an interface to a host programming language. One can therefore see that file management systems for microcomputers were clearly not designed for individual requirements involving fairly complex applications, nor were they aimed at business applications.

Minimum memory requirements for the selected systems vary from 24K to 48K bytes while their prices range from $200.00 to $400.00, excluding

extension packages. Available extension packages include mainly word processors as well as mathematical and statistical packages.

USER INTERFACE:

Only two of the five systems, TIM and Jinsam 2.0, provide error recovery functions to deal with hardware failures, such as loss of power, as well as error trapping facilities to prevent a system from crashing when a user attempts to enter invalid data. As mentioned in [BAR 81], the user manuals for the other three packages (question marks in Figure 5-1) leave the issue of error recovery and error trapping in doubt. One is therefore led to draw the conclusion that system reliability has not been a primary concern for designers and manufacturers of file management systems for microcomputers. Unfortunately, at the current stage of software development, this observation generally applies to a large number of programs directed to the microcomputer environment.

The programs are mainly menu-driven systems intended for nontechnical individuals. As previously mentioned, only TIM offers an interface to a host programming language forprogram development; Jinsam 2.0 offers an interface package on disk. Three packages (TIM, The Microconductor and Jinsam 2.0) provide an on-line help command, in response to which the system produces a list of commands the user might need. Lastly, macro-commands to store sequences of repeatedly used instructions are available on only one system, The Microconductor, the latter being the sole system capable of supporting typed out commands in addition to

menu-driven instructions.

DATA DESCRIPTION:

Jinsam 2.0 and Micro Manager both exclusively support alphanumeric fields. This severe limitation explains their lack of mathematical functions in generating reports, as can be seen in Figure 5-1. TIM is the only system which allows for the definition of virtual fields. The latter expression is used to designate fields whose values are automatically calculated by the system, as specified by the user, from the contents of other fields whenever data are entered or modified in those source fields. Furthermore, it should be noted that none of the programs allow data fields to vary in length.

Four of the five systems specify a maximum number of fields allowed. This number varies significantly from 20 in The Microconductor to 255 for Jinsam 2.0.

DATA RESTRUCTURING:

Two programs (TIM and Jinsam 2.0) surprisingly allow for field type definitions to be revised after the data have been entered without causing the loss of the previously entered data. One would have expected such a feature to be available in larger-scale software only.

DATA ENTRY:

Only one of the five systems, Profile II, provides for user-designed screens, thereby permitting a user to position the fields of a record at any location desired and/or to suppress specific fields from view. This program also allows characters or lines to divide or highlight portions of the screen and supports reversed lettering. The rarity of user-designed screens in file management systems currently available on the market is very unfortunate when one considers such a feature may be very attractive for any user wishing to match screen layouts with supporting input (paper) documents.

DATA UPDATE:

Jinsam 2.0 and Profile II offer no record update facilities other than letting the user update one record at a time. Only one system (Micro Manager) provides a method for simultaneously updating all records that meet specified criteria, a feature designated as automatic multirecord updates in Figure 5-1. TIM and The Microconductor support system-triggered updates; however, it should be noted that these cannot be considered as automatic multirecord updates since, pursuant to a display of all records that meet specified criteria, the user must actually change each record one at a time. None of the systems provide for an audit trail of updates.

DATA RETRIEVAL/QUERY:

All but one system, Micro Manager, allow a user to request that more than one field meet given criteria while only two programs, TIM and Jinsam 2.0, support complex nested conditions on multiple fields joined by AND and OR operators. Although three systems (TIM, Jinsam 2.0 and Profile II) permit a user to search for a record falling within a specific range (range conditions), TIM is the only program which allows for the request of all records with null values in a particular field. Such a null value feature is interesting in that it can be used, for example, to determine which records of a file have not yet been updated and/or to print only records that contain data in a particular field. All systems support substring searches of a desired field value. In three of the five programs (TIM, Jinsam 2.0 and Profile II), predefined queries can be named and stored for later use while only one of those systems, TIM, allows for predefined queries to be subsequently modified. TIM represents also the only program which provides a subfile (or snapshot) feature, thereby permitting the creation of a file from another file based on user-specified selection criteria.

REPORT GENERATOR:

In Figure 5-1, one can see that TIM offers most of the facilities for generating reports one would expect to find in a stand-alone report generator for larger-scale computers. The four other systems are indeed

not comparable to TIM: they offer fewer facilities to define report formats in a precise fashion and are very limited in terms of calculating mathematical or statistical results derived from existing records. Jinsam 2.0 offers mathematical and statistical facilities in extension packages.

ACCESS CONTROL:

With the exception of The Microconductor, all programs provide some type of data security mechanisms to restrict file access to authorized users only.

The above presentation leads us to conclude that TIM goes far beyond the other four selected systems. Given the relatively limited environment of a file management system for microcomputers, that program indeed puts together quite an impressive number of features, in addition to having its own built-in text editor that provides some word-processing facilities without resorting to a word-processing extension package. Within this context, one is tempted to say that users "get their money's worth". Nonetheless, the following important features should be incorporated into TIM to render it the perfect file management system directed to noncomputer users of single-use, single-task microcomputers:

. User-designed screens for data entry,

. Typed out and macro commands,

. Variable length fields,

- An increased maximum number of fields per file,
- Automatic multirecord updates,
- Transaction logging and recovery,
- The facility to write reports to a file.

5.2 Single-user database management systems

In 1979, Maryanski and Slonim reported in [MAR 79] that only two vendors, International Data Base System and General Robotics, claimed to be marketing a database management system for microcomputers at the time, even though the two writers indicated that vendors had expressed considerable interest in software of this type. Based on the growth pattern of database management systems for minicomputers and mainframes, and on the assumption that "history repeats itself", they predicted that several database management systems for microcomputers would surface within the next few years. There were obviously right. By 1981, a few more of these software products were already reported on, including:

- dBASE II (Ashton-Tate),
- Condor 3 (Condor Computer Corp.),
- MDBS III (Micro Data Base System Corp.),
- FMS-80 (Systems Plus),
- Selector IV (Micro AP),
- TITAN (developed at the University of Montreal),
- Micro-SEED (International Data Base System).

In 1984, the catalog of database management programs for microcomputers published in the November issue of BYTE Magazine [BON 84] included forty-seven different programs, among which twenty-two were classified as single-user database management systems and six were actually labelled as multiuser programs.

In the present section, we embark upon a survey of single-user database management systems currently available on the microcomputer market. At the onset, an important point should be noted: all file management systems (covered in Section 5.1) have been automatically excluded from this category, as also were all multiuser database management systems, the latter being dealt with in Section 5.3. Let us also remember here that, within the context of this report, the distinguishing characteristic of a file management system, in contrast with a database management system, resides in its managing only one file at a time. In this same line of thought, the fundamental peculiarity of a multiuser system program lies in its capability to support more than one user at a time, as opposed to the environment involving a single-user database management system.

A particularly interesting point stands out in a close review of single-user database management systems included in the catalog published in 1984 [BON 84]; indeed, all twenty-two system programs listed therein are based on the relational data model. For reference purposes, that list of products (and vendors) was reproduced in Figure 5-2, along with certain general features pertaining to such programs, namely: minimum memory requirements, supportive operating system and selling price. The notation "*" was used to highlight, in Figure 5-2, those software products that are most frequently subject to review and/or analysis in the literature.

The list, presented on the preceding page, of seven single-user database

FIGURE 5-2 RELATIONAL SINGLE-USER DATABASE MANAGEMENT SYSTEMS

Product Name	Vendor	Min RAM	Operating Systems	Price
10-Base	Fox Research	192K	MS-DOS 2.0	$495
Citation	Eagle Enterprises	96K	PC-DOS,	$185
		58K	CP/M-80	
CodeWriter	CodeWriter Corp.	?	Apple II+, IIe, PCjr IBM PC standard OS's	$249
Condor 3 *	Condor Computer	80K	PC-DOS, MS-DOS	$650
		64K	CP/M-80,-86,MP/M-80	
DataEase *	Software Solutions	?	PC-DOS, MS-DOS	$600
DATAVU PLUS	Thinker's Soft	128K	CP/M-86, PC-, MS-DOS	$250
DB3	Tarbell Electronics	56K	CP/M, -86, CCP/M-86	$249
dBASE II *	Ashton-Tate	128K	CP/M-86, PC-, MS-DOS	$495
		48K	CP/M-80	
FMS-80 *	FMS Software Ltd.	64K	CP/M,-86, PC-,MS-DOS TurboDOS,MP/M,MmmOST	$395
Fast File	ICCS	128K	PC-DOS	$350
filePro	The Small Computer	64K	CP/M	$199
filePro 16	The Small Computer	256K	PC-DOS	$495
Knowledgman *	Micro Data Base Systems	192K	CP/M-86, PC-, MS-DOS	$500
LAN:DATASTORE	Software Connections	192K	MS-,PC-DOS, p-System	$450
power-base 2	PowerBase Systems	256K	PC-DOS, MS-DOS	$395
R:base 4000 *	MicroRIM Inc.	256K	PC-, MS-DOS, CP/M-80	$495
RL-1 Data Base	ABW Corp.	128K	PC-DOS, MS-DOS	$495
Revelation *	Cosmos	320K	PC-DOS, MS-DOS, Pick	$950
SALVO	Software Automation	128K	MS-DOS, PC-DOS	$496
		64K	CP/M	
SAVVY PC	Excalibur Technology	64K	SAVVY	$395
Sensible Solution	O'Hanion Computer	128K	CP/M,MS-DOS,TurboDOS, MP/M,CPC/OS,MmmOST	$695
Smart Data Mgr	Innovative Software	192K	PC-DOS, MS-DOS	$495

management systems for microcomputers reported on in 1981 includes five system programs based on the relational data model and only two network-based systems, namely:

. MDBS III (Micro Data Base System Corp.),

. Micro-SEED (International Data Base System).

At this point in time, I am not aware of any other database management system program for microcomputers based on the network model. Furthermore, the sole hierarchical database management system I came across in my research is a system program developed by Micro Data Base System Corp., named HDBS, which actually represents the forefather of the current MDBS and appears to have been withdrawn from the market. A well-established compagny in the area of database management systems for microcomputers, Micro Data Base System Corp. is today concentrating its selling efforts on its two other products which, very popular in the microworld, are marketed under the following names:

. Knowledgman (relational-based),

. MDBS III (network-based).

Let us now review the general features of single-user database management system programs for microcomputers included in Figure 5-2. One can see that the average selling price of such software products runs in the vicinity of $450.00, which represents a 50% increase over the average price of the microcomputer file management systems surveyed in the preceding section. Minimum memory requirements for single-user database management systems range from a low of 48K to a high of 320K bytes. This represents a very significant increase over those of file

management systems where a minimum memory of 48K represented the high point of the minimum memory requirement range. Not surprisingly, the single-user database management system programs included in the 1984 catalog are principally based on the CP/M, MS-DOS and PC-DOS operating systems, the latter representing the single-user operating systems by far the most widely used in the microworld. However, certain database management systems offer program versions supported by other operation systems such as Pick (Revelation) and p-System (LAN:DATASTORE).

In order to proceed to a detailed review and analysis of the single-user database management systems for microcomputers, three system programs were selected among those currently marketed, namely:

- dBASE II (Ashton-Tate),
- Condor 3 (Condor Computer Corp.),
- MDBS III (Micro Data Base System Corp.).

The choice of dBASE II and Condor 3, both of which represent relational-based database management systems, took mainly into consideration the fact that those two products rank among the oldest, best known and most widely used software products of the relational type. With regard to network-based database management systems, MDBS III, rather than Micro-SEED, will be covered in detail in the present section: MDBS supports numerous functions and represents a system program for microcomputers that is comparable to a much larger scale database management system. As a consequence of its outstanding capabilities, MDBS has always enjoyed extensive coverage in the literature.

Our study of each of the three selected database management sytems will be presented within the categorization of the fourteen functions of a database management system discussed in Section 3.4 and summarized in Figure 3-5. One realizes, however, that the CONCURRENCY CONTROL function will be totally excluded here since such a function does not have any relevance in a single-user environment, where database management systems fundamentally do not support more than one user simultaneously getting access to the system. Assuming, however, that two or more individuals may be authorizedto use the system at one time or another, the ACCESS CONTROL function remains pertinent to the present discussion. Thus, following an introduction to the three specific software products identified above, these system programs will be examined within the framework illustrated in Figure 3-5, as follows: USER INTERFACE, DATA DESCRIPTION, DATA STORAGE DESCRIPTION, DATA RESTRUCTURING, DATA REORGANIZATION, DATA ENTRY, DATA UPDATE, DATA RETRIEVAL/QUERY, REPORT GENERATOR, INTEGRITY CONSTRAINTS, ACCESS CONTROL, DATA BACK-UP AND RECOVERY and, lastly, PHYSICAL STORAGE.

Figure 5-3 summarizes major features of the three database management systems within such a categorization. This figure aggregates information extracted from various sources, mostly from [ABB 82, BAR 81, HOL 82, KRU 83].

FIGURE 5-3 FUNCTIONS AND FEATURES OF SINGLE-USER DATABASE MANAGEMENT SYSTEMS

FUNCTIONS and features	dBASE II	Condor 3	MDBS III
USER INTERFACE			
Menu-driven	No	Reports	No
Command-driven	Yes	Yes	IDML (1)
Established language interface	No	No	Yes
Built-in language	Yes	Limited	No
Maximum number of opened files	2	1	Unlimited
Macro commands	Yes	Yes	Yes
On-line help command	Yes	Yes	Yes
DATA DESCRIPTION			
Maximum characters per field	254	127	65,535
Maximum characters per record	1,000	1,024	65,535
Maximum fields per record	32	127	255
Data types - Alphanumeric	Yes	Yes	Yes
Alphabetic	No	Yes	No
Real	No	No	Yes
Integer	Yes	Yes	Yes
Decimal	Yes	No	Yes
Dollar	No	Yes	No
Logical (Y/N)	Yes	No	Yes
Date	No	Yes	Yes
Time	No	No	Yes
User views	No	No	?
DATA STORAGE DESCRIPTION			
Maximum number of key fields	7	8	N/A
Data clustering	No	No	Yes
DATA RESTRUCTURING			
Add fields	Yes	Yes	DRS (1)
Delete fields	Yes	Yes	DRS (1)
Change field size	Yes	ASCII dump	DRS (1)
DATA REORGANIZATION	Limited	Limited	DRS (1)
DATA ENTRY			
Maximum records per file	65,535	32,767	Unlimited
Screen generator	No	Yes	SCREEN (3)
DATA UPDATE			
Automatic multiple updates	Yes	Yes	DML (2)
System triggered updates	Yes	Yes	DML (2)
Full-screen editing	Yes	Yes	SCREEN (3)

FIGURE 5-3 FUNCTIONS AND FEATURES OF SINGLE-USER DATABASE MANAGEMENT SYSTEMS (CONTINUED)

FUNCTIONS and features	dBASE II	Condor 3	MDBS III
DATA RETRIEVAL/QUERY			
Snapshots	Yes	Yes	QRS (1)
Complex nested conditions	Yes	Yes	QRS (1)
REPORT GENERATOR			
On-screen display format	No	Yes	No
Multilevel subtotals	One level	Yes	REPORT (1)
Multifield computations	One level	Yes	REPORT (1)
INTEGRITY CONSTRAINTS			
UNIQUE field	No	No	Yes
Range constraints	No	Yes	Yes
Default values	No	Yes	No
Referential constraints	No	No	Yes
User-defined constraints	No	No	Yes
ACCESS CONTROL			
User password	No	No	Yes
Multiaccess privileges	No	No	Yes
Granularity - file	No	No	Yes
field	No	No	Yes
record	No	No	No
DATA BACK-UP AND RECOVERY			
Transaction commitment	No	No	RTL (1)
Transaction logging	No	No	RTL (1)
Transaction recovery	No	No	RTL (1)
PHYSICAL STORAGE			
File structure - sequential	Yes	Yes	Yes
B+tree	Yes	Yes	Yes
hashing	No	No	Yes
Access method - sequential	Yes	Yes	Yes
direct	Yes	No	Yes
indexed	Yes	Yes	Yes
pointer	No	No	Yes

(1) Extension package available.
(2) Must be user programmed with Data Manipulation Language (DML).
(3) Must be user programmed using extension package available.

dBASE II

dBASE II, developed by Ahston-Tate, is one of the oldest and most popular relational-based database management systems commercially sold on the microworld market. In that position, it is often used in the literature as a standard against which newcomers are evaluated.

Two main versions of this software product are currently available, dBASE II and dBASE III; the latter represents an extended version of dBASE II which was developed for the IBM AT microcomputer system and introduced only recently on the market. The following discussion deals mostly with dBASE II. However, where applicable, the principal enhancements incorporated in the newer version will be highlighted.

dBASE II is written in 8080 machine language. A minimum memory of 48K bytes is required to run the database management system on a 8080, 8085 or Z-80 8-bit microcomputer under a CP/M-80 operating system. The minimum memory requirement is increased to the level of 128K bytes for 16-bit machines under a CP/M-86 or PC-DOS operating system.

Condor 3

Condor 3, introduced by Condor Computer Corp., also represents one of the oldest and best-known database management systems for microcomputers. As was the case with dBASE II, this software product is based on the relational data model; one can find in the literature,

however, some criticism directed at the limited relational capabilities provided by Condor 3. One could say that the problem centers principally around the following significant restriction in its query language: all SELECT, PROJECT, JOIN, SORT and alike commands must be invoked separately and explicitly. Moreover, resulting data are physically stored in temporary relations.

Given that, in many regards, the functions and features of Condor 3 resemble those provided by dBASE II very closely, the following discussion will highlight only areas of Condor 3 where these two database management systems diverge.

MDBS III

MDBS III, developed by Micro Data Base System Corp., is generally accepted in the literature as being "the" best database management system for microcomputers currently available on the market. Nonetheless, the actual use of this software product appears to be relatively limited in the microworld. Such a situation can be explained principally in terms of the two following factors: the deficient initial market penetration strategy adopted by Micro Data Base System Corp. and, more importantly, the much higher price tag attached to MDBS III as compared to other software products marketed as database management systems for microcomputers.

As illustrated in Figure 5-4, the basic price for the minimum version of

FIGURE 5-4 MDBS III MODULES AND PRICES

		8-Bit CP/M version (single-user)	16-bit UNIX/ Xenix version (multiuser)
MINIMUM SYSTEM:			
1. Data Definition Language	(DDL)	$ 2,250 (1)	$ 6,675 (2)
2. Data Manipulation Language	(DML)		
EXTENSION PACKAGES:			
3. Query Retrieval System	(QRS)	1,125	2,500
4. Recovery and Transaction Logging Module	(RTL)	2,225	3,350
5. Interactive Data Manipulation Language	(IDML)	900	2,000
6. Design Modification Unit	(DMU)	1,150	1,700
7. Data Restructure System	(DRS)	N/A	N/A
8. Screen Handler	SCREEN	N/A	N/A
9. Report Generator	REPORT	N/A	N/A
	Total	$ 7,650	$ 16,225

(1) With one programming language interface;
added programming language interface: single-user $ 450
multiuser $ 725.

(2) Multiuser prices vary depending on operating systems and number of users.

the system currently stands at $2,250.00; this selling price exclusively covers the two basic modules, the Data Manipulation Language (DML), to run on a single-user Zilog Z-80 microcomputer-based system. The cost actually exceeds $7,500.00 for a single-user version comprising four of the seven expansion packages for which quotations were available. For a multiuser version of MDBS III, the corresponding price runs beyond $16,000.00, and to over $70,000.00 for a VAX/VMS version.

MDBS III, written in assembly language, is offered over an impressive range of microprocessors, including: the Zilog Z-80 and Z-8000, the Intel 8080, 8085 and 8086, as well as the Motorola 68000.

USER INTERFACE:

dBASE II is a two-level system: an interactive data storage and retrieval package, and an application development package. At the first level, the user interface is driven by typed out commands, as opposed to being menu-driven as was the case for most file management systems for microcomputers.

At the second level, dBASE II provides an interpreted and structured built-in language for the programming of applications, rather than an interface with a host programming language. The built-in language includes all commands related to the interactive data storage and retrieval package, to which condition, loop and input/output statements have been added. The major advantage of the dBASE II language, as

compared to a conventional programming language, resides in an improved ease-of-use for nontechnical individuals. However, the dBASE II language is less powerful; as a point of fact, this is generally regarded as being a major drawback.

An extremely important restriction was included in dBASE II: at the application developement level, only two relations with a single index per relation can be simultaneously opened for access purposes. Such a constraint severely limits the use of dBASE II as a development tool for relatively complex applications involving several entities. In dBASE III, the restriction has been lessened by setting it at ten files rather than two.

At the application level, Condor 3 provides for neither an interface to an established programming language, nor a built-in language to the application programmer. However, a user can create macro commands to store a sequence of instructions. Given that the set of commands in this software product includes some conditional and input/ouput commands, simple applications can be developed through the use of such a feature. Nevertheless, Condor 3 definitely does not support the development of complex applications since this system program does not comprise any branching or individual record access commands. Over and above that, only one file may be opened at a time.

All MDBS III modules are of the batch type except the Query Retrieval System module (QRS) and the Interactive Data Manipulation Language (IDML). The latter module, which supports an extensive use of macro

commands, is intended to simplify the data entry and update process. One of the most interesting characteristics of MDBS resides in that the system supports an imposing range of interfaces to established programming languages, including: Pascal, PL/1, COBOL, FORTRAN, C, BASIC, compiled BASIC and ASSEMBLER.

DATA DESCRIPTION:

The CREATE command of dBASE II allows the user to dynamically create new relations by specifying the relation name, along with the names, types and sizes of its attributes. dBASE II supports only a limited set of data types, including the following: CHARACTER, INTEGER, floating-point DECIMAL and LOGICAL (Y/N). All fields are of fixed length. Lastly, dBASE II does not support the definition of user views.

Contrary to dBASE II, Condor 3 does not support the LOGICAL (Y/N) data type, although it permits the Julian DATE field type. It allows, however, only for fixed-point decimals to store monetary figures of DOLLAR; one can see this puts a severe limit on the type of nonbusiness applications which can be implemented using Condor 3.

DATA DESCRIPTION represents the principal function rendering MDBS III so powerful a system even when compared to larger-scale database management systems of the network type. Indeed, not only does MDBS support a fairly extensive set of data types, it more importantly makes it possible, through the Data Definition Language (DDL) of MDBS III, to

directly specify the following relationships between record types [HOL 82]:

. One-to-one relationships	i.e.,	TYPE	1:1
. One-to-many relationships	i.e.,	TYPE	1:n
. Many-to-many relationships	i.e.,	TYPE	n:m
. Recursive one-to-many relationships	i.e.,	TYPE	1:n
		OWNER	"abc"
		MEMBER	"abc"
. Recursive many-to-many relationships	i.e.,	TYPE	n:m
		OWNER	"abc"
		MEMBER	"abc".

Contrary to dBASE II and Condor 3, MDBS III supports variable-length records. The limitations of the three system programs with regard to maximum number of fields per record, maximum characters per field as well as maximum characters per record are summarized in Figure 5-3.

DATA STORAGE DESCRIPTION:

dBASE II allows the definition of up to seven multifield secondary indexes per relation for fast keyed access purposes. However, the system automatically updates only one of those indexes when a new record is added through an APPEND command. The index to be updated is specified by the user through a USE command:

e.g. USE EMPLOYEE INDEX employee-no.

The other indexes must be updated through a re-INDEX command as a

pointer file.

Condor 3 also supports the definition of secondary indexes for fast keyed access. An index key, however, can be only made up of a single attribute.

Data clusters, or areas, may only be specified in MDBS III in order to ensure that related files are physically stored close to each other on disk.

DATA RESTRUCTURING:

dBASE II allows the user to dynamically add and/or delete fields and change the field size of a relation through a MODIFY STRUCTURE command. However, the use of this command is not straightforward, mainly because the system automatically purges the records of the modified relation. Hence, the user must first off-load the records of a relation into another relation, and then re-load the records in the modified relation. The reloading process is done by matching field names, thereby making it possible for the user to add and/or delete fields:

```
e.g. USE EMPLOYEE
     COPY TO NEW-EMPLOYEE STRUCTURE
     USE NEW EMPLOYEE
     MODIFY STRUCTURE
        (*full-screen editing*)
        employee-no     (numeric,    6)
```

```
name            (character, 30)
manager-no      (numeric,    6)
APPEND FROM EMPLOYEE.
```

Condor 3 also allows the user to add, delete and rearrange fields. However, in order to change the size of a field, the user must first dump the relation contents into an ASCII file, and then load it back from such a file once the relation structure has been modified.

On the other hand, the Data Restructure System (DRS) extension package of MDBS III permits database restructuring without having to dump and reload the database.

DATA REORGANIZATION:

For both dBASE II and Condor 3, as will be later seen in our discussion of the PHYSICAL STORAGE function, the user does not have any control over the file structures. However, a user can dynamically create new secondary indexes to improve the access time to the data. Both systems permit read and write operations from any ASCII file.

DATA ENTRY:

In dBASE II, new records may be added through the use of either the INSERT or the APPEND commands. The use of the INSERT command results in

the new records being inserted at the beginning of the file, whereas with the APPEND command, records are added atthe end of the file. INSERT allows full-screen editing of up to nineteen records per screen with scroll up and down through all newly created records of the relation. In my opinion, this represents a very interesting characteristic of dBASE II, which actually has no equivalent in larger-scale database management systems. Its drawback, however, is that the system will automatically maintain secondary indexes of a relation only if the additional tuples are added through the APPEND command.

Condor 3 allows the user to define on-screen form layout for data entry and update, by simply positioning the cursor where desired on the screen and by typing the attribute names and underscores for the field length.

For DATA ENTRY, DATA UPDATE and DATA RETRIEVAL, MDBS III supports two versions of its Data Manipulation Language (DML), a batch DML version and an interactive IDML version. IDML, which basically includes the same commands as DML, additionally allows such commands to be specified as macro commands. Futhermore, IDML includes an extensive on-line HELP command.

Both versions consist of one-record-at-a-time Data Manipulation Languages with user-specified navigation. Available commands may be grouped into the five following categories [KRU 83]:

. Navigational commands to move among sets within the schema;

- Create, delete and modify commands to add or delete records and change individual data items;
- Connect and disconnect commands used when the DDL specifications do not call for automatic insertion;
- Boolean commands to permit the creation of new sets from logical operations on existing sets;
- Miscellaneous commands including database open and close, recovery control and utility operations.

DATA UPDATE:

Different commands are available in dBASE II to modify existing records of a relation, namely: EDIT, CHANGE, BROWSE and REPLACE. Although both EDIT and CHANGE allow only for one-record-at-a-time updates, CHANGE permits system triggered updates:

```
e.g. USE EMPLOYEE
     EDIT 8          (*direct acces to a single record*)
     CHANGE NEXT 20 (*automatic display of 20 records, one by one*).
```

On the other hand, BROWSE supports full-screen editing of existing records while REPLACE automatically updates all records specified by user-defined selection criteria.

DATA RETRIEVAL/QUERY:

dBASE II primarily provide two commands to retrieve records from a single relation, DISPLAY and FIND. DISPLAY allows the user to specify complex multifield selection condition (equivalent to SELECT) together with implicit PROJECT, but without implicit SORT capabilities.

To illustrate, a request to display, in employee number order, the employee number and the name of all employees with either the name "smith" or the employee number "700615", who work in the toys department could be specified in the following manner in dBASE II:

```
e.g. USE EMPLOYEE
     SORT ON employee-no TO SORT-EMPLOYEE
     USE SORT-EMPLOYEE
     DISPLAY FOR (name="smith".or.employee-no="700615").and.
                 dept-name="toy"
          employee-no name.
```

To provide a basis for comparison, the above request could be specified as follows in the large-scale System-R database management system:

```
e.g. SELECT employee-no, name
     FROM EMPLOYEE
     WHERE (name="smith" OR employee-no="700615") AND
          dept-name="toy"
     ORDER BY employee-no.
```

The DISPLAY command in dBASE II presents a significant restriction in

terms of processing speed since it only accesses files sequentially or directly by means of the record number. To overcome such a limitation, the user must invoke the FIND command which directly locates a record based on the value of a key field through the use of supporting secondary indexes. The user must, however, specify to the system which secondary index to use. For example, the following request would cause the display in employee number order, of the first ten employees with an employee number equal to, or greater than 700615:

```
e.g. USE EMPLOYEE INDEX EMPLOYEE-NO
     FIND "700615"
     DISPLAY NEXT 10 employee-no, name.
```

Note also that dBASE II does not check for duplicate tuples when performing a PROJECT.

For multiple file operations, dBASE II includes two commands that must be invoked explicitly: JOIN and UPDATE. Therefore, multirelation requests cannot be specified in a single command as is the case in large-scale database management systems. The UPDATE command is particularly suitable for master file updates by transaction-files.

As previously mentioned, all relational commands of Condor 3 must be explicitly invoked, as opposed to dBASE II where the DISPLAY command implicitly includes both SELECT and PROJECT commands. For example, the same query as that illustrated for dBASE II could be translated into the following commands in the query language of Condor 3:

```
e.g. SELECT EMPLOYEE WHERE (name="smith" OR employee-no="700615")
```

```
                    AND dept-name="toy"
        PROJECT RESULT BY employee-no, name
        LIST RESULT BY employee-no.
```

MDBS III offers an extension package named Query Retrieval System (QRS) for on-line retrieval. QRS supports a high-level non-procedural English-like language to retrieve set-oriented data from the database. With the exception of the navigational command THRU, this language closely ressembles the relational language provided in dBASE II. To illustrate this similarity, the above query would be expressed as follows using QRS commands, if one assumes that a set type named SYS-EMP is available for direct access to the EMPLOYEE record type:

```
e.g. SET OF 60
     LIST employee-no, name
     FOR (name="smith" OR employee-no="700615") AND dept-name="toy"
     THRU SYS-EMP.
```

The SET command in MDBS III allows the user to specify the display format of numeric fields in terms of number of digits before and after the decimal point. However, one is surprised to observe that the SET command applies unconditionnally to all numeric fields, without any regard to a particular mix of integer and decimal fields.

REPORT GENERATOR:

Condor 3 supports a more powerful report generator than does dBASE II,

and perform multilevel subtotals and subheadings. Furthermore, Condor 3 presents an interesting particularity in that it allows the user to specify on-screen display layouts.

The Query Retrieval System (QRS) extension package of MDBS III also performs this function, although to a minimal extent. To produce stylized report formats, the user must either program them or acquire the add-on report generator module.

INTEGRITY CONSTRAINTS:

dBASE II does not support the definition of any integrity constraints, not even the simplest constraints designated as field constraints which were described in Section 3.3 of the present report (range checks, UNIQUE field, default values, etc.).

Condor 3 provides for the specification of two integrity constraints within the category referred to above, range checks and default values.

Semantic constraints usually specified through a network-based Data Definition Language are provided in MDBS III, namely:

- DUPLICATES ARE NOT ALLOWED,
- INSERTION AUTOMATIC,
- Range checks.

ACCESS CONTROL:

dBASE II and Condor 3 are certainly not secure systems; indeed, these software products do not incorporate any provision whatsoever in order to limit access to the database. This problem was apparently resolved in the newer version of dBASE (dBASE III).

MDBS III provides security at the user level. However, the add-on Design Modification Unit (DMU) module is required to set and change user names and passwords in the system. Passwords are used to restrict the access to the database at the record type, set type and/or data item levels. Access privileges include read and write. The following example illustrates the incorporation of access controls into the schema definition:

```
e.g. ***********At the user level*************
     USER "anyone" with "pass"
     READ  ACCESS IS (a)
     WRITE ACCESS IS (b)
     ***********At the record level***********
     RECORD employee IN ANY AREA CALC KEY (employee-no)
     READ  ACCESS IS (a)
     WRITE ACCESS IS (a)
     ITEM employee-no  INT 6   RANGE 0 to 999999
     ITEM name         STR 30.
```

DATA BACK-UP AND RECOVERY:

dBASE II and Condor 3 do not provide any facilities to log transactions for recovery from either soft or hard crashes.

The add-on Recovery and Transaction Logging (RTL) module of MDBS III is particularly interesting to mention in connection with error recovery and back-up. This system indeed makes error recovery possible at two levels. At the first level, called page-image processing, the system prevents a complex transaction sequence from going to disk before it is completed. Transaction delimiters are user-specified through the use of commands such as LGCPLX and LOGENX to log the starting point and ending point of complex transactions, respectively. At the second level, referred to as transaction logging, the system maintains a log file on disk to record all transactions which change the database. Furthermore, the module also includes a utility program to restore the database from the most recent back-up.

PHYSICAL STORAGE:

In dBASE II, one CP/M file is used to store each base relation and each secondary index. The structure definition (data dictionary) of a relation or index is included at the beginning of the file. dBASE II supports one, and only one file structure for each class of file. Base relations are stored in heaps with sequential and direct accesses while secondary indexes are stored in B+trees with indexed-sequential access.

Although Condor 3 supports the same file structures as those included in dBASE II, it solely provides exclusively for sequential and indexed-sequential access methods. Unlike that of dBASE, the structure definition (data dictionary) of a relation is stored, in Condor 3, independently of the relation in a distinct CP/M file with .DEF extension.

MDBS III uses virtual memory where the page size and number of pages are user-defined. MDBS III sets are stored in multilevel balanced trees, pointer arrays for FIFO and LIFO sets, or hashed tables when the CAL KEY option is set. MDBS III therefore supports both indexed-sequential and hashed keyed access methods.

5.3 Multiuser database management systems

Software designated as multiuser database systems for microcomputers began to reach the market approximately two years ago. Figure 5-5 presents, indicating vendors, minimum memory requirements and prices, the six software products that were classified as multiuser database management systems in the catalog of microcomputer database management systems published in the October 1984 issue of BYTE Magazine [BON 84]. In a way, this remains a relatively small number of system programs when compared with the nineteen file management systems and the twenty two single-user database management systems included in that same catalog. However, one could immediately add to the list provided in Figure 5-5, MDBS III (Micro Data Base System Corp.) which, as discussed in the preceding section and shown on Figure 5-4, also comprises multiuser versions based on UNIX-like operating systems.

A rapid review of price quotations reported in Figures 5-4 and 5-5 readily highlights marked discrepancies in the selling price of multiuser database management systems currently available on the microworld market. Indeed, significant variances exist in both the following respects:

(1) In terms of the cost of two different software products within the multiuser category; to illustrate, $500.00 for filePro 16 in contrast with $7,500.00 for MDBS III.

FIGURE 5-5 MULTIUSER DATABASE MANAGEMENT SYSTEMS

Product	Vendor name	Minimum Memory	Operating Systems	Price
filePro 16	The Small Computer Co	256K	XENIX marketed by Radio Shack UNIX system	$ 495 995
Informa	Unlimited Processing	64K 256K	CP/M-80, DPCOS, MmmOST, TurboDOS DPCOS-86, CP/M-86, MP/M-86, MS-DOX 2.X 3COM, PCNet, Davong ,Novell, Corvus	795 995 1,495
LAN:DATASTORE	Software Connections	192K	MS-DOS, PC-DOS, p-System -- up to 5 users -- up to 16 users	945 1,945
Optimum Version 7.03	Uveon Computer System	64K 128K	CP/M-80, -86, MP/M-86 PC-DOS, TurboDOS, MmmOST	595
PDBase	IOTC Inc.	64K	p-System, PC-DOS, MS-DOS CCOS (concurrent)	245 795
Unify	Unify Corp.	256K	UNIX, XENIX and other UNIX implementations VAX 780 (for minicomputers)	1,495 14,500

(2) In terms of the price tag for two different versions of the same product; for instance, a microcomputer implementation of Unify costs $1,000.00 whereas a version of the same program to run on a VAX 780 minicomputer sells for $15,000.00. Even if one disregards minicomputer versions, noted discrepancies will remain significant; for example, the selling price of LAN:DATASTORE depends on the number of users supported by a system, i.e., from $1,000.00 for a maximum of five users to $2,000.00 for a maximum of sixteen users.

The average selling price of multiuser database systems for microcomputers is far greater (in the vicinity of a 100% price increase) than for the single-user software products examined previously.

In my opinion, such noticeable differences are mainly attributable to the type of multiuser configuration provided by a particular database management system as well as the number of users allowed by the system. Thus, prior to proceeding any further in the current section, let us first clarify what is being designated as a multiuser database management system, and determine existing possibilities in relation to the configuration of such a system.

One will remember that the fundamental peculiarity of a multiuser system program lies in its capability to support more than one user simultaneously getting access to the system. In view of the fact that certain products are inaccurately being classified as multiuser

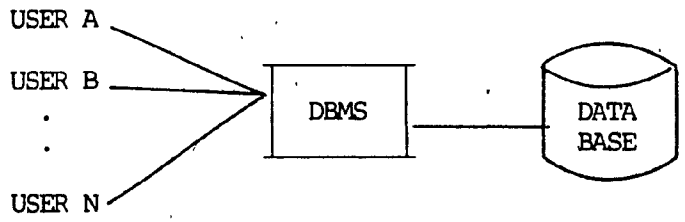
database management systems, this distinguishing characteristic in comparison with single-user software programs must be defined with greater precision here. Indeed, the mere fact that multiple users can simultaneously get access to a computer, and even to a database management system, does not mean for all that the database management program and, more importantly, the database is actually being shared by the multiple users.

Within this context, there is no cause for using the multiuser designation in situations where a copy of the database management program resides in several independent segments of the memory (for example, under MP/M operating system) and the database, particular to each user, does not integrate shared data of an enterprise. Neither is there justification for such a multiuser reference in cases where the database management program is shared (i.e., re-entrant code), but the database is not. Nevertheless, one may find certain manufacturers and/or vendors of database management systems for microcomputers who market products of the two types described above under the label multiuser database management systems, without further information and at very low prices, relatively speaking. In my opinion, such software products would be more appropriately categorized as multiple single-user database management systems running on the same computer.

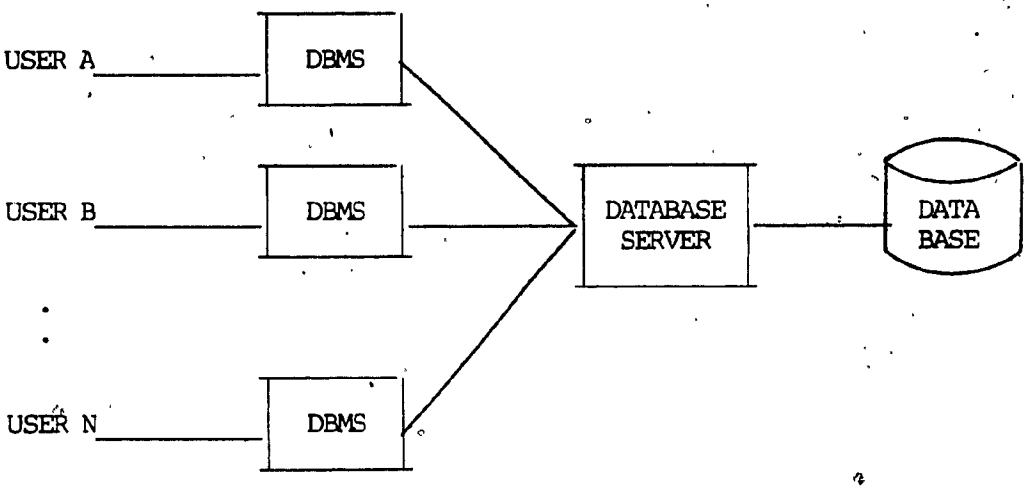
As illustrated in Figure 5-6, a genuine multiuser database management system environment could be created by means of one of three existing configurations; thus, the present section is hereafter divided into three parts, each subsection covering a specific configuration of

FIGURE 5-6 CONFIGURATIONS OF MULTIUSER DATABASE MANAGEMENT SYSTEMS

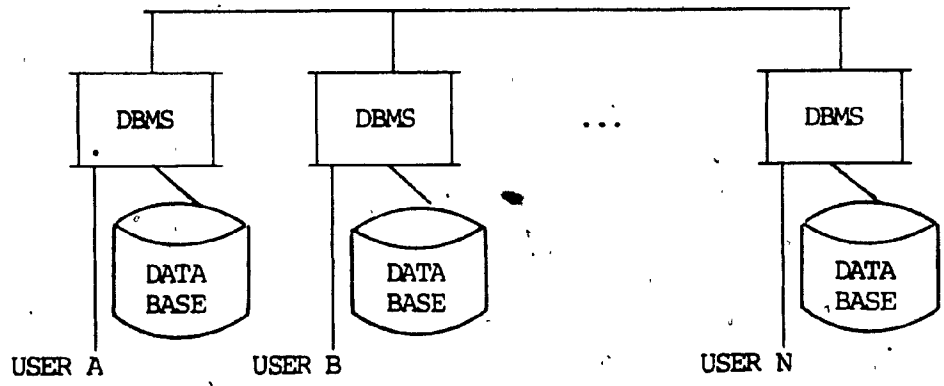
(a) Time-sharing configuration



(b) Database server configuration



(c) Distributed database configuration



multiuser database management systems for microcomputers, namely:

- The time-sharing configuration (5.3.1),
- The database server configuration (5.3.2),
- The distributed database configuration (5.3.3).

As will be covered in greater depth in the following discussion, one can distinguish the three types of system configurations mentioned above principally by their respective capability to support multiple computer and database sites.

5.3.1 The time-sharing configuration

Figure 5-6 (a) graphically depicts the time-sharing configuration, which represents by far the most widely used structure for database management systems on minicomputers or mainframes. This includes, among others, all the large-scale database management systems presented in Section 3.3 dealing with the study of data models, namely: INGRES, SQL-SD, IDMS, TOTAL, IMS, etc..

The major difficulties associated with the inclusion of concurrent access to the database by multiple users, in comparison with a system where only a single user may get access to the database at one time, are generally acknowledged in the computer world. Let us outline here the four following main problems of a multiuser system [ROT 78]:

- The database management system must incorporate concurrency controls to synchronize multiple concurrent updates of the database.

- The database management system should support the definition and maintenance of different views of the database for different users.

- Selective access controls are required to grant different access privileges to different users.

. The data granularity of the system for concurrency control and access control (LOCK) should be small to allow for a high level of concurrency.

5.3.2 The database server configuration

One can see that the database server configuration illustrated in Figure 5-6 (b) closely resembles, at least pictorially, the back-end database system organization with multiple host computers which was depicted in Figure 4-1. The principal divergence between these two approaches lies in the degree of off-loading of the functions of a database management system onto a back-end computer. With regard to a back-end database system, as explained in Section 4.3, all database management functions are off-loaded from the host computer to a back-end database machine, with the host computer retaining solely the function pertaining to the interface with users/application programs. Exactly the inverse situation occurs in a database server configuration. In the latter case, all database management functions indeed remain the responsibility of the host computer with the exception of the function relating to the control of access to the database itself, which is off-loaded to a back-end computer called a database server. The terms back-end controller or file server may also be found in the literature in reference to this back-end machine.

In a microcomputer environment, host computers could represent complete single-user microcomputer systems with their own mass storage devices, possibly floppy disk drives. The shared database could reside on a common hard disk controlled by a dedicated microcomputer-based database server. These multiple microcomputers would then be interconnected over a local area communication network.

The main advantages of such a configuration as compared to a time-sharing structure may be summarized as follows:

. The database server configuration allows for sharing of an integrated database by multiple microcomputer users while retaining the advantages associated with the use of a dedicated computer, namely:
  - Full user control of their computer environment (i.e., users can work with their machine when they wish to);
  - Total user independence with regard to the applications they desire to implement on their computer.

. A dedicated database server allows for the sharing of an expensive hard disk drive among several relatively inexpensive microcomputers.

. The overall performance of the system is improved through the use of the capacity of parallelism of the multiple microcomputers.

The principal problem resulting from such an approach to database management resides in the potential build-up of a bottleneck in the database server since requests originating from several host microcomputers must be processed through this back-end machine at one time.

To my knowledge, there is no operational microcomputer-based database

server currently available on the market. In this area, one must recognize that a database machine such as the iDBP Intel Database Processor described in Section 4.3, represents more than a database server since it also performs database operations such as SELECT, PROJECT and JOIN. With regard to experimental file servers, Mitchell described two machines of this type in [MIT 82]: XDFS (Xerox Distributed File System), built at the Xerox Palo Alto Research Center, and CFS (Cambridge File Server), developed at the Cambridge University Computer Laboratory. The design of XDFS was really oriented toward the control of a database, i.e., as a database server.

5.3.3 The distributed database configuration

Figure 5-6 (c) depicts the distributed database configuration. Although this expression immediately reminds us of the system which was designated as a geographically distributed database system over a long-distance communication network in Section 4.3, it should be noted that the notion of geographical distribution does not apply to the configuration being studied here. Thus, the latter structure is organized around a local area communication network rather than a long-distance one. Nonetheless, the basic problems associated with such an architecture remain essentially the same in both cases.

The following crucial point may be observed by comparing parts (b) and (c) of Figure 5-6: contrary to the database server approach, both the database management program and the database are distributed over multiple computers in the distributed environment. Thus, although this fundamental distinction is not always clearly identified in the literature, only the configuration pictured in Figure 5-6 (c) actually fits the definition of a distributed databasesystem. A system based on the approach illustrated in part (b) would be more adequately categorized as a distributed processing database system or, briefly stated, a multicomputer database system.

The degree of importance attached to the problems associated with this type of structure mainly depends on the method used for distribution of the data over multiple computer installations or sites [ROT 78]. If

data replication over the various sites is not allowed (i.e., a data item is stored at a single site) and if only batch updates of the data dictionary are supported (i.e., the communication network is closed in order to update the data dictionary replicated at each site), the distributed database configuration is then similar to the time-sharing organization in terms of complexity, except that in the former case, an inter-computer communication protocol must be established. Moreover, additional decisions must be made regarding, for instance, the processing of requests that cannot be satisfied locally, i.e., will such requests be forwarded to the remote site for execution or should the remote site send a copy of the data for local execution.

If either one of the two restrictions indicated in the preceding paragraph is not incorporated in the system, the distributed database configuration then supports data replication over multiple sites. As a consequence, extremely complex problems such as synchronization of updates of multiple remote copies of the data, transaction back-up and recovery of multiple remote sites, etc., come into play here. Hence, such areas remain to date at the stage of experimental research.

In this context, one may be permitted to conclude that only the first of the two situations described above, i.e., where replicate data and dynamic data dictionary updates are not allowed, appears theoretically feasible today, using multiple microcomputers. To my knowledge, however, no commercial system of this type has yet been introduced in the microworld.

This completes the survey of database management system products currently available in the microcomputer market, within the categorization previously established for such systems (file management, single-user, and multiuser database management systems), thereby enabling us to proceed to the next chapter which will present our overall evaluation of microcomputer database management systems both in terms of the current state of the art, and future trends emerging in this rapidly expanding field in computer science.

VI. AN OVERALL EVALUATION OF DATABASE MANAGEMENT SYSTEMS FOR MICROCOMPUTERS

The primary objective sought in the current chapter is to proceed to an overall analysis and evaluation of the database management systems for microcomputers surveyed in the preceding chapter, in order to determine the extent to which these software programs meet the application needs of their actual and potential users. In this context, database management systems will be assessed in two different, but complementary, aspects.

In a first step, Section 6.1 will evaluate the functional completeness of database management systems designed for the microworld, and will summarize major conclusions drawn from a comparison of the functions actually incorporated in the reviewed microcomputer database management systems (i.e., what functions these programs actually do perform), with the extensive list of functions of a database management system developed in Section 3.4 and categorized in Figure 3-5 (i.e., what functions such systems should be doing). Secondly, Section 6.2 will investigate the performance of database management systems for microcomputers in terms of storage requirements and processing time, and formulate an opinion on the manner in which such systems do perform.

Section 6.3 will then extend the overall examination presented here into the future by looking at the direction of three major trends in database management systems for tomorrow's microworld. These general tendencies become apparent pursuant to the survey, analysis and evaluation of prior

and current microcomputer database management systems endeavoured in this paper.

Clearly defining the scope of the critical analysis we now embark upon is of paramount importance. In this context, the discussion contained in this chapter completely excludes the file management systems surveyed in Section 5.2. Reasons for such an exclusion are two-fold; on the one hand, the usefulness and completeness of those database management systems were investigated and evaluated throughout, and summarized at the end of, the section indicated above. On the other hand, and more importantly, the following analysis focuses on database management systems fundamentally intended for small businesses.

Over and above that, instead of dealing with specific database management programs for microcomputers, the evaluation presented hereafter will rather encompass observations applicable to the majority of microcomputer database management systems. As a consequence, the overall analysis contained in Sections 6.1 and 6.2 will especially address database management systems of the relational-type, and almost inherently exclude certain outstanding microcomputer database management systems such as, for instance, network-based MDBS III, although the most noticeable particularities of such systems will be briefly underlined where applicable.

6.1 A functional evaluation

On the basis of the survey comprised in Chapter V, and within the framework of functions established in Section 3.4, we are now in a position to group the function-related limitations identified in database management systems for microcomputers under three general themes, namely:

- Low-level query languages,
- Inexistent data controls,
- Limited programming languages.

Assuming that a small business either possesses, or calls on, the technical resources necessary to use its database management system for fairly complex applications, these weaknesses are so significant that they may actually jeopardize to a certain extent, as will be demonstrated below, the usefulness of the program for the organization.

On the other hand, database management systems for microcomputers present certain particularities of special interest in comparison to larger-scale database systems. Mostly attributable to precisely the more limited resources available in a small business environment, these characteristics aim at simplifying the use of a database management system. Here again, one may group the function-related particularities pinpointed in such a software program for microcomputers under three main themes, as follows:

- A large set of field types,

. Text processing capabilities,

. Application development tools.

Within this context, the current section will evaluate each of the major limitations and particularities of microcomputer database management systems, and conclude by assessing the overall significance of those elements in relation to target users and their management information needs, i.e., small businesses.

Low-level query languages

Although one may describe the query languages supported by database management systems for microcomputers as interactive, non procedural, and English-like languages, they remain at a much lower level than the query languages of larger-scale database management systems such as SEQUEL (System-R), QUEL (INGRES), or QBE. For instance, most relational microcomputer database management systems require separate and explicit invocation of the relational commands SELECT, PROJECT and JOIN. Such an observation also applies to support commands such as SORT or TABULATE sum, average, etc.. To illustrate this situation, let us initiate a request to retrieve, in alphabetical order, the names of all employees over 40 years of age working on the first floor and who earn more than their manager. This request would be expressed by the following two-variable declaration and three-command lines in the QUEL language (INGRES):

e.g. RANGE OF E, M IS EMPLOYEE

```
RANGE OF D IS DEPARTMENT
RETRIEVE (E.name)
WHERE E.salary      >  M.salary       AND
      E.manager-no  =  M.employee-no  AND
      E.dept-name   =  D.dept-name    AND
      D.floor-no    =  1              AND
      E.age         >  40
ORDER BY (E.name).
```

The following sequence of commands (or an equivalent sequence) would be required to specify the same query in the Condor 3 language:

```
e.g. PROJECT EMPLOYEE BY employee-no, salary
     WRITE RESULT TO "ASCII-filename"
     DEFINE MANAGER
          manager-no      (numeric, 6)
          manager-salary  (dollar,  7)
     READ MANAGER FROM "ASCII-filename"
     JOIN EMPLOYEE MANAGER BY manager-no
     JOIN RESULT DEPARTMENT BY dept-name
     SELECT RESULT WHERE salary    >  manager-salary  AND
                         floor-no  =  1               AND
                         age       >  40
     PROJECT RESULT BY name
     SORT RESULT BY name.
```

Thus, nine commands are needed to express the same request in Condor 3 language. One can see that the first four commands actually stand as the

equivalent of the first variable declaration statement in INGRES. Indeed, one must resort to an ASCII file in order to create the relation MANAGER, such requirement resulting from the three following limitations inherent in Condor 3:

- The matching fields of two relations participating in a JOIN command must have the same name.

- Field names of an existing relation cannot be modified through a REORG command.

- Field names are matched when records from one relation are attached to another relation through an APPEND command.

Considered individually, each of these restrictions may first appear relatively minor. On a cumulative basis, however, such limitations can render the expression of queries a significantly more complex task for the user. In connection with the last five command lines in Condor 3, this set of instructions correspond fairly well to the last three commands in INGRES, with the following exception: the two JOIN operations are expressed explicitly rather than, more naturally, within the WHERE or SELECT clause.

Moreover, in order to achieve fast indexed access to a particular record or set of records of a relation based on their key value, the user must specify to the system which secondary index to use. In such a context, a relational database management system for microcomputers does not

require an optimizer to determine the most efficient access paths to answer a query; indeed, one can conclude that the query language then resembles more closely a network-type language with user-specified execution instructions than a so-called high-level language.

Notice that, although the query language of Condor 3 was used to demonstrate the above limitation, such a weakness is inherent in most microcomputer database management systems. For example, the query language of dBASE II merely follows suit.

Inexistent data controls

Several functions of a database management system incorporated in Figure 3-5 have been totally eliminated in the majority of database management systems for microcomputers, since either such functions represented elements not essential in a single-user environment or were too complex from an implementation point of view. Omitted functions principally pertain to the area of data control and include the following:

- INTEGRITY CONSTRAINTS,
- ACCESS CONTROL,
- CONCURRENCY CONTROL,
- DATA BACK-UP AND RECOVERY,
- Multiple user views.

In this context, the responsibility of controlling the accuracy and security of the database lies with the application programmer who, as

will be seen in our analysis of programming languages, does not always have the tools necessary to perform such a task. This limitation does not, however, apply to MDBS III; indeed, one will remember here that the latter system does provide extension packages to support at least three of the functions enumerated above, namely: ACCESS CONTROL, CONCURRENCY CONTROL and DATA BACK-UP AND RECOVERY.

Limited programming languages

With the noticeable exception of MDBS III and Revelation (Cosmos Co.), the vast majority of database management systems designed for the microworld do nct provide for an interface with one or more conventional programming languages such as BASIC, COBOL or Pascal. On the other hand, several microcomputer database management programs such as, for instance, dBASE II, R:base 4000 (MicroRim), and Knowledgman (Micro Data Base Systems) incorporate a structured built-in language, sometimes advertised as a fourth generation language, which is particularly suitable for nontechnical users wishing to use such a tool. One must realize, however, that these languages present less programming capabilities, and more inherent limitations, than do conventional programming languages. As a consequence, a valid rule of thumb would consist of limiting their use to relatively simple applications involving few entities. To illustrate, it seems to be generally accepted that such built-in languages were not intended for the development and implementation of integrated accounting application programs [KRU 83, SAU 84].

A survey dealing with business accounting programs for microcomputers reported by Heinz in [HEI 84] tends to confirm to a certain extent, the validity of this observation. Indeed, out of a sample of nearly forty accounting programs listed therein, only two companies, SBT Corporation and Champion Software, offered a multi-component software product built around a database management system, namely dBASE II in both cases. It should be noted, however, that such software does not qualify as integrated accounting packages. For example, the package offered by Champion Software actually corresponds to a stand-alone inventory system with automatic updates to the general ledger.

Nonetheless, a company located in Ohio, TLB Inc., markets two genuine integrated business systems named Solomon I - General Accounting, and Solomon II - General Accounting with Job Cost, both developed around MDBS III. To illustrate, Solomon II includes general ledger, accounts payable, accounts receivable, payroll, fixed assets, job costing and address maintenance subsystems, all of which belong to a single package and run from one master menu. The database serving all subsystems resides on one disk file [KRU 83].

Large set of field types

In order to define a field type, one may usually find a classical set of basic data types in large-scale database management systems, comprising principally INTEGER, CHARACTER and REAL. Certain microcomputer database

management systems, mostly among recent software products, provide a far more interesting range of data types. For example, DataEase (Software Solutions Inc.) supports no less than eight data types, namely: TEXT, NUMERIC, NUMERIC STRING, DATE, TIME, DOLLAR, LOGICAL (Y/N) and CHOICE [JAC 84]. The processing peculiarities of two of these data types, NUMERIC STRING and CHOICE, are quite unusual. With the NUMERIC STRING type, a user may design custom data masks such as "(999) 999-9999" for telephone numbers or "999-999-999" for social insurance numbers. Any separators of predefined field masks are automatically displayed to speed entry and reduce the incidence of errors. With the CHOICE type, as many as ninety-nine optional entries of up to sixty characters each may be created. These options appear at the top of the screen during data entry or modification. The entry message related to the number keyed in is automatically inserted in the database to speed entry and ensure standard field length and content.

Text processing capabilities

The microworld has recently witnessed the appearance of an extended type of database management system referred to as idea processors or text database management systems [SHA 84]. Contrary to other database management systems for microcomputers, these newcomers support variable length records and include advanced text editing facilities. In fact, such products integrate full text editing functions to conventional database management system functions, thereby offering a complete tool for processing a partly structured, partly free-format database for

applications such as a library reference system or a bibliography.

Application development tools

A particularity of database management systems for microcomputers resides in the fact that these software products provide easy-to-use development tools to design and implement application programs. Such tools principally pertain to the areas of data input and data output, and include:

- On-line screen generators,
- A large set of preprogrammed mathematical and statistical functions,
- Sophisticated report generators.

One may also find similar tools in the minicomputer or mainframe environment; however, they are usually sold separately as stand-alone software packages. In the microworld, the input/output tools mentioned above form an integral part of the database management system and do not require any programming.

Although bearing in mind that there are some noticeable exceptions such as, for instance, MDBS III and Revelation, one may conclude from the preceding functional evaluation that most database management systems for microcomputers are directed to nontechnical users with relatively simple application needs. The first part of this affirmation derives from the particularities of these software products in the microworld,

which clearly emphasize easy-to-use tools for on-line applications. On the other hand, the second portion of the assertion made above results from the limitations inherent in database management systems for microcomputers, which put severe limits on the implementation of fairly complex applications resorting to a database schema with several entities as well as relationships between those entities.

6.2 An efficiency evaluation

The performance of a software program is usually assessed in terms of storage requirements and processing time. With regard to database mangement systems for microcomputers, the various factors impacting on their performance can be grouped together under two broad categories, namely: those elements which pertain to the configuration of the microcomputer system as described in Section 2.4, and those constituents which relate to the database management system itself.

To single out factors of primary importance, the first class includes the following elements:

(1) Memory space available,

(2) Microprocessor processing speed,

(3) Storage capacity of disks,

(4) Data transfer speed of disks,

whereas the second group rather comprises factors such as:

(5) File structures and access methods,

(6) Query language level.

The current section will solely examine the first four constituents enumerated above, i.e., factors associated with the configuration of microcomputer systems. In view of the direct relationship existing between the elements related to a database management system and the data model on which is based such a software program, our analysis of the last two factors mentioned above is deferred to the next chapter,

Chapter VII, where the two principal models (relational and network) will be evaluated as applied to databases on microcomputers.

Within such a context, the performance of a database mangement system running on a microcomputer system, for the purpose of the current section, depends to a significant degree on the capacities of two of the individual system components discussed in Chapter II namely, microprocessors and mass storage devices. One must recognize that low end microcomputer systems based on 8-bit or Intel 8086-like microprocessors with floppy disks automatically show poor performance in terms of factors pertaining to the microcomputer system configuration, i.e., factors (1) to (4) above. In this line of thought, one can summarize the major difficulties encountered in the development and use of a database management system for such a limited computer system in the following manner:

- The database management system code must be as compact as possible to fit into the small memory space of the microcomputer. One way to achieve this goal consists of coding the database management system in assembly language (such as MDBS) or to design a highly modular overlayable database management program (such as TITAN [FAL 82]). Another alternative resides in eliminating some functions and features of large-scale database systems for the microcomputer environment. As mentioned in the first section of the current chapter, this latter alternative has been selected by most designers and manufacturers of database management systems for microcomputers, which often do not support the following

functions:

- Multiple user views,
- INTEGRITY CONSTRAINTS,
- ACCESS CONTROL,
- CONCURRENCY CONTROL,
- DATA BACK-UP AND RECOVERY.

. The size of the database is limited by the low storage capacity of the floppy disks. As was demonstrated by Bell in [BEL 80], unless the organization is very small and the business involves an unusually small number of entity types with a moderate number of occurences of each entity, a so-called corporate database simply cannot be set up properly on a microcomputer database management system.

. Finally, in the microcomputer environment, the database system is clearly input/output bound, the main reason being the intrinsic slowness of floppy disks. In a relational environment, the system response time is in the order of seconds or minutes for simple queries. For complex queries with JOIN, the processing time may be so long that consideration should be given to treat them as off-line operations.

The three major difficulties discussed in the present section do not apply, as such, to high end microcomputer systems based on Motorola 68000-like 16-bit or 32-bit microprocessors with 216K to 512K of memory and high-capacity hard disk drives. To illustrate, Kruglinski reported

in [KRU 83] that the average time required by dBASE II to perform a sequential pass of a non-indexed file containing 1,635 records of 150 characters each with a 5-digit key is approximately one minute and thirty-eight seconds when the file resides on a floppy disk, but only twenty-five seconds when it resides on a hard disk.

6.3 Future trends

To conclude the analysis and evaluation of microcomputer database management systems being conducted in the current chapter, the principal tendencies that will prevail in tomorrow's microworld will be outlined. These tendencies may be extrapolated from the study of prior and current database management systems presented in this report. Needless to say, numerous areas of future development and expansion can certainly be outlined for microcomputer database management systems. Indeed, as demonstrated throughout our study of these software products, the technology involved here has evolved at an amazing pace over the past five years.

In my opinion, three major trends for future database management systems for microcomputers become apparent from the study conducted in the present report. Thus, one may expect that the most probable and marked advances will be achieved in the following areas, each one being briefly discussed hereafter:

- Migrated minicomputer multiuser database management systems,
- Locally distributed database management systems,
- Integrated operating and database management systems.

Migrated minicomputer multiuser database management systems

As discussed in Section 2.3, the appearance of newer high end 16-bit and

32-bit microcomputers offering capacities similar to those of minicomputers triggered the migration of multiuser time-sharing operating systems from the minicomputer environment to the microworld. UNIX and Pick represent two major examples of operating systems migrated to the microcomputer environment. One is permitted to foresee that this same approach will be generally adopted with regard to database management systems for the newer high end microcomputers; thus, the industry may witness, in the relatively near future, a massive migration of database management systems designed for minicomputers to the microworld. In this context, a particularly plausible selection of prime candidate large-scale database systems for a scaled-down microcomputer version may include the following software programs, in view of the popularity enjoyed by such products as well as their relatively affordable scale:

- INGRES (Relational Technology),
- TOTAL (Cincom Systems Inc.),
- ORACLE (Relation Software Inc.).

Locally distributed database management systems

Given the increasing attractiveness of interconnected mini or microcomputers for medium and large organizations, we believe that the area of locally distributed microcomputer-based database management systems will stand as an important area of continued coordinated efforts in the future by both research centers and database management system manufacturers. Thus, one may foresee the introduction on the market of

a wide range of distributed database management systems over a local area communication network. Such systems could be described in the following manner, in increasing order of complexity:

. Basic file servers with minimal access controls,

. Database servers with extensive concurrency and sharing controls,

. Back-end database systems,

. Fully distributed database systems.

For more detailed information on the different possible configurations of a distributed database management system on microcomputers, one may wish to turn back to Sections 4.3 and 5.3 dealing with, respectively, the needs of large organizations in terms of multiple interconnected microcomputer and database systems, and multiuser database management systems for microcomputers.

Integrated operating and database management systems

The microworld represents an environment particularly suitable for the development and extension of integrated operating and database management systems. This type of software integrates the functions of a database management system to those of an operating system as was done, for instance, in the Pick operating system designed by Pick Systems Inc.. Indeed, such an approach can make life much simpler for the microcomputer system user. In this context, one may expect significant future developments for this type of software for microcomputers, which could be referred to as database operating systems.

VII. AN ANALYSIS OF DATA MODELS FOR DATABASES ON MICROCOMPUTERS

As mentioned in Section 3.3 introducing the three data models, there has been much debate in the literature with regard to the two principal data models, relational and network. Indeed, various authors, [MCG 76, MIC 76] among others, have engaged in most lively discussions in an attempt to assess which data model actually surpasses the other. The primary goal of the current chapter is not to carry on with this debate as such, but rather attempt to determine, within the context of our study of microcomputer database management systems, which model is indeed most appropriate as applied for databases on microcomputers. As indicated in Section 6.2 dealing with the overall performance evaluation of database management systems for microcomputers, such an endeavour inherently includes a performance evaluation of such software products in terms of those factors which pertain to database management systems themselves.

To achieve our stated goal, the analysis of data models for databases on microcomputers will be performed in two steps. Firstly, analysis criteria pertinent to data models will be established, and an overall evaluation of the two models based on these criteria will be presented in Section 7.1. In a second step, in Section 7.2, an attempt will be made to apply relative weights to the selected criteria within the specific environment of microcomputers in order to determine which data model a microcomputer database management system should support.

As will be seen in the following discussion, one may observe that current applications of the two models, relational and network, clearly tend over time to combine more and more particularities from both models. Indeed, database management system designers actually try to compensate for weaknesses found in their selected model by borrowing strengths identified in the other model. Thus, should this tendency persist, the debate itself with regard to these data models will become obsolete.

7.1 Analysis criteria

There are basically two types of criteria to evaluate data models for databases [ULL 80]: the use criteria and the performance criteria. The use criteria are determined strictly from the user point of view. They include considerations such as the ease of understanding the model, the ease of learning and using the database language, the ease of modeling the real-world environment into a restrictive schema, the ease of programming application software, etc. On the other hand, the performance criteria mainly include criteria related to space requirements (main memory and disk space) and processing (response) time.

In the literature, it appears to be generally accepted that, with respect to user criteria, the relational model is superior to the network model for several well-known reasons, namely:

- The conceptual simplicity (picturability) of the model,
- The availability of high-level languages (such as SEQUEL, QBE),
- The strong theoretical basis of the model,
- The novelty of the model.

It is interesting to note that nontechnical users apparently hold the same view concerning the superiority of the relational model. For instance, Hodges [HOD 82] concluded that for a user who is not a professional programmer, the relational type is almost mandatory, unless the user requirement is for a very simple, repetitive operation

with massive data.

On the other hand, one must consider that the performance of a database management system is directly related to the proximity of the conceptual (logical) and the internal (physical) levels of the system. Where the two levels are close to each other, as in the network approach, the implementation of the internal level (file organization, indexing techniques and access methods) is relatively simple and straightforward (for example, hashing and multilist in TOTAL). Furthermore, the processing requirements for the mapping of the two levels are relatively low. The database language is generally a low-level language consisting of procedure calls. The navigation decisions are taken by the user. Finally, the implementation requires the use of well-known techniques.

In the relational approach, there is a clear distinction between the two levels. The implementation of the internal level, and the mapping, are not so obvious. The database languages can be very high-level languages. As a consequence, the system must make several decisions. Many optimization algorithms are required (such as the single-variable query decomposition of INGRES [STO 76]).

One can conclude that, in a general sense, the implementation of a database management system of the relational type is more complex than a network-based system and, as a consequence, the overall system performance is lower.

In summary, the foregoing evaluation leads us to identify the following

major strengths and weaknesses in the two models:

- Relational model: (+) Simplicity of the model,
  (+) High-level languages,
  (-) Low performance.
- Network model: (+) High performance,
  (-) Complexity of the model,
  (-) Low-level languages.

As indicated above, the industry is currently witnessing a situation where the database management system designers actually compensate for weaknesses of a particular model by incorporating in it the strengths of the other model. To illustrate, relational-type System-R incorporates at the physical level several efficient storage techniques usually found in network-based systems such as inter-relation pointers and data clusters. Furthermore, as discussed in Section 5.2, MDBS III of the network type offers an interactive-like high-level query language with set-oriented operators. In addition, this system program supports an extended network model where many-to-many and recursive relationships may be specified as simply as in the relational model. In view of the above, one is led to believe that, should such a trend extend into the future, the debate as to which of the two models, relational or network, is a better model will simply be left behind. New database management systems will incorporate significant particularities of both types.

7.2 A comparative study

In the micro-environment, both types of evaluation criteria take on particular importance, especially where the database management sytem is developed to run on low end microcomputer systems as defined in Section 2.4.

Indeed, the typical user of database management systems for microcomputers is often a nontechnical individual who has limited resources, mainly in terms of skills and financial resources, to invest into the development of application software. Thus, the ease of use criteria become extremely important within such a context.

On the other hand, to cope with the intrinsic limitations of low end microcomputer systems, the implementation must be kept as simple as possible in order to achieve a relatively good system performance. With the significantly slower microcomputer hardware, the assumption, often set forth for large-scale database management systems, that the CPU processing time can be ignored no longer holds. For example, on an 8-bit microcomputer, the effectiveness of inverted lists over multilists is significantly reduced as the query complexity and the file size increase.

The decision as to which criteria are most important, in relative terms, for a microcomputer database is not a simple one to make. In view of the nontechnical nature of users of such systems, one may believe that

ease of use criteria outweight the performance criteria, in which case the relational model is most suitable for a database on microcomputer. Moreover, such a position may be reinforced by assuming that microcomputer users make, in general, simple queries on relatively small files.

However, if one considers the significant limitations of low end microcomputer systems described in Section 6.2, the network model appears much more appropriate for a microcomputer database. For instance, the execution of several relational operations such as JOIN, INTERSECT and DIVIDE is inherently inefficient, and particularly so on any 8-bit microcomputer due to the small amount of memory available and the slowness of floppy disk drives. It was reported in [FAL 82] that unless the files are very small, a JOIN on large files in not practical given that it may take up to an hour.

The survey of database management systems presented in Chapter V provides us with an answer to this question. Indeed, the microcomputer market, including both database management system designers and users, has definitely decided upon database management systems of the relational type for microcomputers. Given the choice made by the market, one may conclude that the ease-of-use criteria outweigh the performance criteria for microcomputer users, thereby calling for relational database designs. Microcomputer application software designers, however, seem to have a marked preference for a network-based database management system such as MDBS III, particularly where fairly complex application programs are involved.

VIII. CONCLUSIONS AND RECOMMENDATIONS

The present study of database management systems for microcomputers enabled us to highlight the fundamental concepts related to the two technologies amalgamated in these software products, microcomputers and database management systems; and, more importantly, to proceed to the review, analysis and evaluation of the characteristics and objectives of this new type of software.

By way of conclusion to this report, and considering the broad nature of the subject being addressed, let us review and summarize the major findings of the foregoing study, grouping them by themes, as follows:

- The categories of database management systems for microcomputers,
- Their computing resource requirements,
- Their target microcomputer user groups,
- The set of functions they support and the extent to which they satisfy target user application needs,
- Their price tags.

Thus the conclusion will include a summary of each important group of findings indicated above, within distinct paragraphs. On the basis of this study, some important recommendations with regard to the future of microcomputer database management systems will be formulated.

In less than five years of existence, three successive generations of so-called database management systems for microcomputers have already

appeared on the market, namely: file management systems, single-user database systems and multiuser database management systems. Each category presents distinct fundamental characteristics. File management systems can access only one data file at a time and that file alone constitutes the database. In a single-user database management system, only one user can access the database at a time whereas multiuser database management systems support multiple concurrent users on a time-sharing basis or over a local area network of single-user systems.

The level of complexity of each type of database management system increases with each new generation and, along with it, the system requirements in terms of computing resources. For instance, the average memory space required for a file management system is approximately 32K (from 24K to 48K), increasing to 128K for single-user database management systems (48K to 320K), and to approximately 192K for multiuser database management systems (64K to 256K). The appearance of each type of software products on the market basically follows, with a few years time lag, the pattern of microcomputer systems: file management systems for 8-bit microcomputer systems with floppy disks, single-user database management programs for Intel 8086-like 16-bit microcomputer systems with mainly floppy disks, and multiuser systems based on Motorola 68000-like 16-bit or 32-bit microprocessors with hard disks running on a UNIX-like operating system.

Each group of software programs is directed at a distinct market segment among various microcomputer users. File management systems are mainly intended for individuals, single-user database management systems are

aimed at small businesses with relatively simple business application needs, whereas the target market of multiuser database systems consists of medium and large businesses with fairly complex application needs and multiple concurrent users. In general, each category of database management systems includes a set of functions sufficient to satisfy the application needs of the target microcomputer user group. File management systems usually include an adequate set of features and functions to satisfy individual needs for computerized card index systems. Single-user database management systems fulfill the relatively simple on-line business application needs of small businesses, whereas multiuser database management systems can support the development of batch applications involving a complex database schema within a multiuser environment.

The selling price of a database management system for microcomputers generally stands as a good indicator of the system category and capabilities: in the vicinity of $250.00 for file management systems, at approximately $500.00 for single-user database management systems, and over $1,000.00 for multiuser database management systems.

Pursuant to the extensive study of database management systems for microcomputers performed within the context of the current report, were are now in a position to formulate some important recommendations with regard to the future of such software products, namely:

. An urgent improvement to incorporate in currently existing database management systems for microcomputers pertains to the

inclusion of data control functions which are nearly inexistent at this point in time, namely:

- ACCESS CONTROL,
- INTEGRITY CONSTRAINTS,
- CONCURRENCY CONTROL,
- DATA BACK-UP AND RECOVERY.

. In view of the evolution of microcomputers into more and more minicomputer-comparable systems, database system vendors should give serious consideration to the migration of large-scale database management systems into the microcomputer environment, rather than allocating their efforts to the development and upgrading of database systems designed for microcomputers.

. The emphasis should be placed on pursuing the combination of important particularities of the two data models (relational and network) for databases on microcomputers so that a perfect mix may reside in a conceptual (logical) relational-type database management system over an internal (physical) network-type organization.

. Extensive efforts should continue to be invested in the following research areas related to microcomputer database systems since, in increasing order of complexity, they appear as very logical steps forward for database systems on microcomputers:

- Microcomputer-based database servers,
- Multimicrocomputer database systems,

- Distributed database systems without data replication,
- Distributed database systems with data replication.

. Finally, from a user point of view, there is a need to clarify some basic notions concerning database management systems for microcomputers due to frequent inaccurate classifications; to name some important ones:

- File management systems labelled as database management systems;
- Single-user database management systems advertised as database system programs without any reference to their inherent limitation: a single user at a time;
- Database management systems marketed as multiuser systems while corresponding in fact to multiple independent single-user systems on a single computer.

BIBLIOGRAPHY

[ABB 82] Abbott J.L.
DATABASE MANAGEMENT WITH ASTON-TATE'S DBASE II
BYTE Magazine, Vol. 7, No. 7
pp. 412-416 (July 82)

[AHL 78] Ahlgren A. and Garland H.
EXPANDING APPLICATIONS FOR MICROCOMPUTERS
In: Micro-computer Systems
Vol. 2: Invited papers
Infotech International
pp. 1-10 (1978)

[AND 82] Andrews M.
PROGRAMMING MICROPROCESSOR INTERFACES FOR CONTROL AND INSTRUMENTATION
Prentice-Hall (1982)

[AST 76] Astrahan M.M., Blasgen M.W. et al
SYSTEM R: RELATIONAL APPROACH TO DATABASE MANAGEMENT
ACM Transactions on Database Systems, Vol. 1, No. 1
pp. 96-133 (March 1976)

[BAR 81] Barley K.S. and Driscoll J.R.
A SURVEY OF DATA-BASE MANAGEMENT SYSTEMS FOR MICROCOMPUTERS
BYTE Magazine, Vol. 6, No. 11
pp. 208-234 (November 1981)

[BEL 80] Bell D.
DATABASE SOFTWARE FOR MICROCOMPUTER SYSTEMS
Database Journal, Vol. 10, No. 4
pp. 16-20 (1980)

[BEN 77] Benhaset I. and Goldstein R.C.
DATA BASE SYSTEMS FOR SMALL BUSINESS: MIRACLE OR MIRAGE?
Database Journal, Vol. 9, No. 1
pp. 5-8 (Summer 1977)

[BON 84] Bond G.
A DATABASE CATALOG
BYTE Magazine, Vol. 9, No. 11
pp. 227-238 (October 1984)

[BOW 80] Bowles K.L.
BEGINNER'S GUIDE FOR THE UCSD PASCAL SYSTEM
BYTE Books (1980)

[BRY 84] Bryan S.
COVERING ALL THE DATABASES
Microcomputing, Vol. VIII, Nos 6, 7 and 8
pp. 46-54 (June 1984)
pp. 48-52 (July 1984)
pp. 52-57 (August 1984)

[COD 70] Codd E.F.
A RELATIONAL MODEL FOR LARGE SHARED DATA BANKS
Communications of the ACM, Vol. 13, No. 6
(June 1970)

[COD 79] Codd E.F.
EXTENDING THE DATABASE RELATIONAL MODEL TO CAPTURE MORE MEANING
ACM Transactions on Database Systems, Vol. 4, No. 4
(December 1979)

[COD 81] Codd E.F.
THE SIGNIFICANCE OF THE SQL/DS ANNOUNCEMENT
IBM (January 30, 1981)

[COD 82] Codd E.F.
RELATIONAL DATABASE: A PRACTICAL FOUNDATION FOR PRODUCTIVITY
Communications of the ACM, Vol. 25, No. 2
(February 1982)

[COO 84] Cook R. and Brandon J.
THE PICK OPERATING SYSTEM - PART I: INFORMATION MANAGEMENT
BYTE Magazine, Vol. 9, Nol. 11
pp. 177-198 (October 1984)

[DAH 82] Dahmke M.
MICROCOMPUTER OPERATING SYSTEMS
BYTE Publications Inc. (1982)

[DIE 81] Dieckman E.M.
THREE RELATIONAL DBMS
Datamation, Vol. 27, No. 9
pp. 137-148 (September 1981)

[DOR 82] Dorricott K.O.
MICRO DATABASES
C.A. Magazine
pp. 106-107 (August 1982)

[ELB 82] Elbra T.
DATABASE FOR THE SMALL COMPUTER USER
NCC Publications (1982)

[FAL 82] Falquet G. et al
A PORTABLE RELATIONAL DATABASE MANAGEMENT SYSTEM FOR MICROCOMPUTERS
Microprocessing and Microprogramming, Vol. 9, No. 1
pp. 17-25 (January 1982)

[GAG 81] Gaglé M. et al
DATA-BASE MANAGEMENT SYSTEMS: , POWERFUL NEWCOMERS TO MICROCOMPUTERS
BYTE Magazine, Vol. 6, No. 11
pp. 97-122 (November 1981)

[GER 79] Gerritsen R.
MICRO-SEED AND ITS APPLICATION
COMPCON fall 79
pp. 188-190 (September 4-7, 1979)

[GOL 82] Goldberg L.A. and Honickman H.W.
INFORMATION MANAGERS; THE KEY TO COMPUTER SUCCESS
C.A. Magazine
pp. 44-54 (August 1982)

[GUP 83] Gupta A. and Toong H.D.
MICROPROCESSORS - THE FIRST TWELVE YEARS
Proceedings of the IEEE, Vol. 77, No. 11
pp. 1236-1256 (November 1983)

[HAM 78] Hamacher V.C. et al
COMPUTER ORGANIZATION
McGraw-Hill Inc. (1978)

[HAR*80] Harmon S.
MICROPROCESSOR TRENDS BEING PIONEERED BY MOTOROLA'S MC68000
Proc. Symposium on Microprocessor Applications in the 80's
Arizona, U.S.A.
pp. 6-12 (March 12-14, 1980)

[HEI 84] Heintz C.
AUTOMATING YOUR BUSINESS ACCOUNTING FUNCTIONS
IBM PC Update, Vol. 1, No. 5
pp. 14-31 (December 1984)

[HOD 82] Hodges C.L.
SELECTING A CP/M DATA BASE MANAGEMENT SYSTEM

[HOL 82] Holsapple C.W.
MDBS III: A FULL-SCALE DBMS FOR MICROCOMPUTER SYSTEMS
In: State of the Art Report
Database - The 2nd Generation
Pergamon Infotech Ltd
pp. 499-519 (1982)

[HUG 82] Hughes P.
THE OPERATING SYSTEM OF THE FUTURE
Microcomputing, Vol. VI, No. 6
pp. 28-31 (June 82)

[INF 78] Infotech State of the Art Report
MICRO-COMPUTER SYSTEMS
Vol. 1: Analysis and Bibliography
Infotech International (1978)

[INF 82] Infotech State of the Art Report
DATABASE - THE 2ND GENERATION
Pergamon Infotech Ltd. (1982)

[INF 83] Infotech State of the Art Report
DBMSs - A TECHNICAL COMPARASON
Pergamon Infotech Ltd. (1983)

[JAC 84] Jacobson B.
DataEase VERSUS Condor AND dBASE II
BYTE Magazine, Vol. 9, No. 11
pp. 289-302 (October 1984)

[JOH 83] Johnson J.
MDBS - BIG PUSH IN MICRO SOFTWARE
Datamation, Vol. 29, No. 1
pp. 50-55 (January 1983)

[KIM 79] Kim W.
RELATIONAL DATABASE SYSTEMS
ACM Computing Surveys, Vol. 11, No. 3
pp. 186-211 (September 1979)

[KOR 83] Kornstein H.
CONCURRENT CP/M-86 AND RECENT ADVANCES IN OPERATING SYSTEMS
Microprocessors and Microsystems, Vol. 7, No. 8
pp. 391-393 (October 1983)

[KRA 81] Krass P. and Wiener H.
THE DBMS MARKET IS BOOMING
Datamation, Vol. 27, Nol. 9
pp. 153-170 (September 1981)

[KRA 84] Krajewski R.
DATABASE TYPES
BYTE Magazine, Vol. 9, No. 11
pp. 137-142 (October 1984)

[KRU 83] Kruglinski D.
DATA BASE MANAGEMENT SYSTEMS: A GUIDE TO MICROCOMPUTER SOFTWARE
Osborne/McGraw-Hill (1983)

[MAC 82] Machrone B.
MDBS: A DATABASE MANAGEMENT SYSTEM
Microsystems, Vol. 3, No. 3
pp. 28-33 (May/June 1982)

[MAR 80] Marshall M.
MICROCOMPUTERS ADDING DATA-BASE MANAGEMENT
Electronics, Vol. 53, No. 16
pp. 98-99 (July 17, 1980)

[MAR 77] Martin J.
COMPUTER DATA-BASE ORGANIZATION
Prentice-Hall, 2nd Ed. (1977)

[MAR 81] Martin J.
AN END USER'S GUIDE TO DATABASE
Prentice-Hall Inc. (1981)

[MAR 79] Maryanski F. and Slonim J.
MICROCOMPUTER DATABASE SYSTEMS.
In: Micro-computer Solftware, Vol. 2: Invited Papers
Infotech International
pp. 157-171 (1979)

[MAR 80] Maryanski F.J.
BACK-END DATABASE SYSTEMS
ACM Computing Surveys, Vol. 12, No. 1
pp. 3-25 (March 1980)

[MCG 81] McGauley E.J.
CHALLENGING THE MINIS
Mini-micro Systems
pp. 135-140 (September 1981)

[MCG 76] McGee W.C.
ON USER CRITERIA FOR DATA MODEL EVALUATION
ACM Transactions on Database Systems, Vol. 1, No. 4
pp. 370-387 (December 1976)

[MCG 79] Mc Glynn D.R.
PERSONAL COMPUTING: HOME, PROFESSIONAL AND SMALL BUSINESS APPLICATIONS
John Wiley and Sons Inc. (1979)

[MIC 76] Michaels A.S. et al
A COMPARAISON OF THE RELATIONAL AND CODASYL APPROACHES TO DATA-BASE MANAGEMENT
Computing Surveys, Vol. 8, No. 1
pp. 125-151 (March 1976)

[MIT 82] Mitchell J.G. and Dion J.
A COMPARAISON OF TWO NETWORK-BASED FILE SERVERS
Communications of the ACM, Vol. 25, No. 4
pp. 233-245 (April 1982)

[MOR 82] Morse S.P., Ravenel B.W. et al
INTEL MICROPROCESSORS: 8008 TO 8086
In: Computer Structures: Principles and Examples
Mc-Graw-Hill Inc.
pp. 615-646 (1982)

[MOT 79] Motorola Semiconductor Products, Inc.
MC68000, ADVANCE INFORMATION
(1979)

[NEL 82] Nelson H.
SWEET SIXTEEN - MICROS COME OF AGE
Microcomputing, Vol. VI, No. 5
pp. 36-38 (May 1982)

[PHI 83] Phillips C.
16-BIT OPERATING SYSTEMS STANDARDS AND MS-DOS
Microprocessors and Microsystems, Vol. 7, No. 8
pp. 371-373. (October 1983)

[RIN 79] Ringland R.
MICROPROCESSOR ARCHITECTURE AND SOFTWARE; CONNECTIONS AND IMPLICATIONS
In: Micro-computer Software
Infotech International
pp. 211-231 (1979)

[ROT 78] Rothnie J.B. and Goodman N.
A SURVEY OF RESEARCH AND DEVELOPMENT IN DISTRIBUTED DATABASE MANAGEMENT
In: Tutorial: Distributed data base management
IEEE Computer Society
pp. 30-44 (1978)

[SAU 84] Sauve P.
UNE BASE DE DONNEES PUISSANTE A LA PORTEE DE L'USAGER:
R:base 4000
Informatique et Bureautique, Vol. 5, Nos 9 and 10
pp. 18-21 (November 1984)
pp. 8-10 (December 1984)

[SHA 84] Shapiro E.
TEXT DATABASES
BYTE Magazine, VOL. 9, No. 11
pp. 147-150 (October 1984)

[SHU 77] Shugart Associates
SA 850/851 DOUBLE-SIDED DISKETTE STORAGE DEVICE
OEM Manuel (1977)

[SHU 79] Shugart Associates
SA 1000 FIXED DISK DRIVE
OEM Manuel (1979)

[STI 81] Stiefel M.L.
SINGLE-BOARD COMPUTERS OFFER GREATER CHOICE AND POWER
Mini-micro systems
pp. 121-132 (September 1981)

[STO 76] Stonebraker M. et al
THE DESIGN AND IMPLEMENTATION OF INGRES
ACM Transactions on Database Systems, Vol. 1, No. 3
pp. 189-222 (September 1976)

[STO 79] Stonebraker M.
CONCURRENCY CONTROL AND CONSISTENCY OF MULTIPLE COPIES OF DATA IN DISTRIBUTED INGRES
IEEE Transactions on Software Engineering, Vol. SE-5, No. 2
pp. 188-195 (May 1979)

[STO 80] Stonebraker M.
RETROSPECTION ON A DATABASE SYSTEM
ACM Transactions on Database Systems, Vol. 5, No. 2
pp. 225-240 (June 1980)

[STO 81] Stonebraker M.
OPERATING SYSTEM SUPPORT FOR DATABASE MANAGEMENT
C.A.C.M., Vol. 24, No. 7
pp. 412-418 (July 1981)

[TAY 82] Taylor R. and Lemmons P.
UPWARD MIGRATION
PART 2: A COMPARISON OF CP/M-86 AND MS-DOS
BYTE Magazine, Vol. 7, No. 7
pp. 330-356 (July 1982)

[TIN 78] Ting P.D. and Tsichritzis D.C.
A MICRO-DBMS FOR A DISTRIBUTED DATA BASE
Proceedings 4th Int'l Conference on Very Large Data Bases
pp. 200-206 (September 13-15, 1978)

[TIT 81] Titus C.A. et al
16-BIT MICROPROCESSORS
Titus, Titus and Larsen (1981)

[TSI 82] Tsichritzis D.C. and Lochovsky F.H.
DATA MODELS
Prentice-Hall (1982)

[ULL 80] Ullman J.D.
PRINCIPLES OF DATABASE SYSTEMS
Computer Science Press (1980)

[WEI 80] Weiss H.M.
DOWN-SCALING DBMS TO THE MICROWORLD
Mini-micro systems, Vol. 14, No. 2
pp. 187-193 (April 1980)

[WHI 81] Whinston A.B. and Holsapple C.W.
DBMS FOR MICROS
Datamation, Vol. 27, No. 4
pp. 165-167 (April 1981)

[WIC 78] Wickham R.
THE IMPACT OF MICROCOMPUTERS ON SMALL BUSINESS DATA PROCESSING
In: Micro-computer Systems
Infotech International
pp. 253-263 (1978)