Presented in three parts is a descriptive analysis of data-base information systems.

Part I reviews the evolution of data-base systems to reveal the direction of their growth and applications. Emphasized are the two primary functions of data-base systems: storage and maintenance of structured information; and presentation of structured output information.

Part II discusses the structuring of information, and introduces a new fundamental approach to this structuring. The approach provides a stable information oriented terminology for relating the conceptual frameworks of existing systems and future systems.

Part III presents a framework, the Data Independent Accessing Model (DIAM), *for describing information and its stored representations. The generality of this framework allows the model to describe most stored representations of existing systems in detail. Over the long term, it can provide a conceptual basis for systematic migration to systems with new improved capabilities.*

Data structures and accessing in data-base systems I Evolution of information systems

by M. E. Senko, E. B. Altman, M. M. Astrahan, and P. L. Fehder

The attention of the computer user community is increasingly focusing on data bases and computerized information systems because of two converging trends. Computerized information systems are coming to play an essential role in business operations, and the hardware and software technology for supporting information systems is in a period of rapid technical progress. In spite of this attention, there is still little common agreement as to what information systems are, the functions they perform, and—from a technical point of view—how they should be designed, implemented, installed, and used. These conditions arise naturally from the newness of the field and the ad hoc nature of existing computerized implementations. A compounding factor is the wide variety of perspectives of those who discuss information systems—for example, the executive, the management consultant, the systems analyst, the mathematician, and the systems programmer. In this paper, we emphasize aspects of information systems that are of importance to systems analysts, systems designers, and implementors.

To create a basis for improved common understanding, we try to step outside the confusion of specialized, overlapping terminology for specific systems, and to create fundamental descriptions of data structures and data accessing in data-base systems. Similar goals have been expressed for operating systems and procedural languages by Dijkstra.[1]

The first section of the paper is devoted to a general discussion of the technical evolution of information systems. The second section discusses information and its structure at the user interface to data-base systems. The third section presents a general model, the Data Independent Accessing Model (DIAM), which uses a few primitive concepts for describing data structures and data accessing properties of existing and proposed data-base systems down to the encoding on hardware devices. This model, even in abstract form, can be useful in comparing specific systems and gaining a clearer understanding of their characteristics. It is intended that DIAM provide the basis for a simpler, more powerful data-base system to which existing systems can evolve in a compatible fashion.

Functions of information systems

Information is defined as an organized collection of facts derived from sources such as reading, observation, and study. There are no widely accepted definitions for the terms *data-base systems* or *information systems*. Since this paper emphasizes the point of view of the systems analyst, a definition is used that covers only the primary functions of existing computer systems that have been given the name information systems. We consider the general category of *information systems* to include systems that perform the following two primary functions:

- Storage and maintenance of representations of structured information
- Presentation of structured output information from the stored representations on request

In this paper, computerized information systems are synonymous with data-base systems and either system is considered to be a subset of the general category of information systems. Typical information system applications are the storage and the presentation of information about such entities as people, airline seats, money, houses, and machine parts. A rapid and efficient storage and presentation mechanism for large amounts of information may be complex but the basic transactions are simple as shown in Table 1.

The complex processing of the substance of the information is excluded from the definition. That is, the processing of information by a linear programming system for decision making may be

Table 1 Storage and presentation of information

Parts file			*Maintenance transactions*	*Presentation requests*
Part name	*Color*	*Weight*		
Gear	Blue	17	M1 Add description of BOLT which is GREEN, weighs 1 lb.	P1 Information on GEAR
Saw	Red	3	M2 Change SAW COLOR to SILVER	P2 Information on all RED parts
Wheel	Red	6	M3 Change the COLOR RED to ORANGE	

supported by the information system, but this complex processing is not a part of the information system.

Advanced computerized information systems support their primary functions by secondary functions such as security, recovery, input checking, and scheduling of system resources. These functions must be considered and provided for in an operational system, but they can be separated from the central questions of representing and accessing organized information. We consider these secondary functions in our discussion, but not in great depth. References 2 and 3 provide access to literature on complex processing and secondary functions.

We wish to call attention to the terms structured information and representation. *Structured information* is a framework of names for explicitly classified entities in the real world. Included is essentially all of the information stored in payroll systems, airline reservation systems, parts inventory systems, and personnel systems. Not included are unstructured textual documents. *Representations* are concrete materializations of structured information. There have been many useful representations for structured information—indentations in clay tablets, pencil marks in ledger books, magnetic spots on disks, etc. Computers deal directly with the representations and only indirectly, if at all, with the information represented by them. Thus the fact that a farmer owns seven horses is independent of the names that are used to stand for it and its possible representations in a ledger book or on a magnetic tape.

Even though information has an independent existence and properties, it has been common practice to force fit the information into a traditional representation or one that is convenient to a particular technology. Those aspects that do not fit must be

Table 2 Evolution of mechanized information systems

System technology	*Hardware technology*		
	Punched cards	*Magnetic tapes*	*Disk files*
Program	Wires	Machine language Symbolic assemblers Input/Output subroutines Procedural languages Report generators Formatted file programs	
Processing	Sequential	Sequential	Random-sequential
File organization	Sequential	Sequential	Random-sequential Indexes Chains
Time lag of representation of real world (minimum)	> 1 day	> 1 day	> 1 second

taken care of by some extra-system human intervention, such as the restructuring of old programs that process unchanging information after the representation of the information has changed. In this paper we work toward a definition of information organization that fits the information.

Evolution of integrated data-base information systems

In perspective, computerized information systems are only the most recent of a series of efforts to record valuable structured information dating back to certain paintings on cave walls, indentations in clay tablets, pencil marks in ledgers, and holes in punched cards. The evolution of data-base information systems discussed in this paper is summarized in Table 2.

punched card—wired program technology

The earliest mechanized information systems that we consider are based on the punched card and the wired program technology of sorters, collators, reproducers, interpreters, and tabulators. Such devices provide faster and more accurate maintenance and presentation of large amounts of information than hand-maintained ledgers. They also provide almost any kind of report available from computer systems. Wired-program technology, however, is highly constrained in representing and processing information, and often requires many more processing steps than a computer to produce the same report. One of the qualitative constraints is that the only method of efficiently processing files of records (boxes of cards) is one card after another in physical order. A quantitative constraint is the small amount of information that can be represented in a single card. The speed of processing cards is slow (a few hundred cards per minute), and all the wired-program processing required per card may not always be done in a single pass of the cards.

magnetic tape and stored programs

The next major advancement in information systems is the stored program machine using magnetic tapes for storing information representations. The primary advantages here are greater speed (one thousand to ten thousand representations read per minute), larger size of the representation (thousands of characters), and increased complexity of processing (the equivalent of thousands of program wires) that may be applied to each record in one pass of the tape. Although this is a dramatic change in technology, the change is more in processing speed and representation size than in the nature of the processing. The processing of information representations on tape continues to be in the sequential batch mode that is characteristic of punched card technology. The terminology of punched card systems—files, records, and fields—has been largely carried over to the tape systems.

With tape-processing equipment, presentations of information became somewhat more sophisticated. Instead of full printouts of the file, printouts are made of *specified subsets* of the file that meet particular preprogrammed conditions based on record content—exception reports. Such reports could be achieved with wired program technology, but they are more easily produced by computers.

stored program methods

In parallel with developments in tape processing technology were advancements in stored program methods for processing information representations. In retrospect, we now recognize the following four stages:

1. Program representations in terms of binary bits or decimal digits punched into cards for direct reading into the computer
2. Symbols for storage reference and machine operations to be translated by assemblers
3. Parameterized assembler language subroutines for processing files
4. Compilers for the translation of procedures described in English-like (COBOL) or mathematical (FORTRAN-ALGOL-PL/I) terms into machine code.

Each of these stages emphasizes sequential processing, and the programs necessarily include procedural descriptions for the accessing of information representations. That is to say, each record is requested in physical sequence from the file being processed and a processing procedure is applied to each record before the next record is requested.

Another parallel, but relatively independent development is the report generator, which is distinguished by the fact that its instructions apply to sets of information representations (files) rather than to single representations (records). Report generators do not normally require procedural descriptions for data accessing. A set of sequential or indexed sequential files is assumed, and the compiler of the report generator simply reads descriptions of the files to determine how records are to be obtained and decoded. Nor does the user need to give a procedural description of his output report construction. The desired representations are simply described in a nonprocedural fashion, and the report generator—by reading the descriptions—determines the required procedures for producing the representations.

This movement from specifying detailed procedures for accomplishing particular tasks to a less procedural or nonprocedural specification—a reduction of housekeeping—is a historical trend in the computing field. As soon as we discover a way of automatically translating a less procedural specification into a reasonably efficient complete specification of the required procedures, we use the less procedural specification in advanced languages. An earlier example is the handling of registers by compilers, and a more recent one is the nonprocedural query language.

The shift from processor orientation to data-base orientation of information processing began with a shift of attention away from files as part of the program. Under the older regime, a program may process many files, but a file is not normally processed by more than one or two programs. In this situation, file (data) descriptions are naturally embedded in the program. As files have become more integrated (in part to reduce update cycle times and to increase consistency), each file has become the target of more than one program.

Thus it has become more natural and clearly less redundant to associate data description with the file. The COPY instruction of COBOL reflects this trend. The trend is emphasized in the formatted file systems (designed originally for military applications) wherein the data description is a property of the data file. Query programs that access a file first access the data description associated with the file and adjust themselves to access the existing physical representation of the data. Another aspect of formatted file systems is that they are designed to handle nonpreplanned, one-time transactions. In earlier systems, transactions were preplanned and generally periodic, so the overhead to program the housekeeping could be allocated over many runs and, therefore, was not a very crucial factor. Overhead is a crucial factor in one-time transactions, both in time required to write a program and time to produce an error-free program.

Formatted file systems go beyond report generators in the report area, and allow informal reports that do not require column-by-column specification of the report form. The programs only require a list of the fields to be printed and then determine the procedures to satisfactorily present the information.

As we have mentioned, all of the nonprocedural programs of this period assume the serial or sequential file organization and this makes for a relatively easy translation of query programs into search procedures. Procedural assistance is, however, required for queries to multiple files. For many problems, complex hierarchic physical-record structures that contain all information about each item being processed are essential to provide efficient sequential processing. Security is under the physical control of the tape owner, who essentially combines the transactions to be run and the tape files in one physical package. The tapes are physically removed from the system when these transactions are completed.

DASD

The appearance of direct access storage device (DASD) technology in the form of large disk and drum storage capable of storing hundreds of millions of characters of representations produced a qualitative change in the processing methodology and power of information systems. In a disk system, any record location can be accessed in less than a second. In a tape system, although successive records may be only ten milliseconds apart, the average distance to a random record is half a file scan (minutes or hours) depending on the tape file size. DASD results in a speed-up of random access relative to sequential access by two to four orders of magnitude. Although direct accessing continues to be slower than sequential accessing to adjacent records by one to two orders of magnitude, the ability to go directly to any random record in the file very often more than compensates for the difference in speed by reducing the number of accesses to be made. Because of the large random access time between randomly selected records, users typically do not random process sequential tape files.

file organization

In sequential processing, the system has to scan the entire file to access records required for a particular transaction. Since the scan time increases little or not at all when more than one transaction is processed during a pass of the tape, the user is motivated to spend hours or days collecting a reasonable batch of transactions, and then process them all during one file scan. The delay in collecting the batches and scanning the tape file makes it physically impossible for sequential processing to keep an up-to-the-second picture of a large number of business activities. The cascading of updates from file to file, often followed by the transmission of updates by mail, can result in corporate-level information or resources being weeks or months out of date.

Online processing through terminals located anywhere in the world coupled to the computer by direct transmisssion lines makes it feasible to locate in DASD and process the records required by a transaction in a few seconds. An obstacle to be overcome in direct-accessing systems is that we normally wish to access and process records based on their content. Existing peripheral devices, however, access records only by record address. This means that in using DASD to avoid sequential scan we must obtain the addresses of the records having the desired content. The solution to this problem is to construct a file organization; i.e., to preprocess and prestructure the information so as to reduce the amount of scanning for the kinds of transactions that are anticipated.

To achieve a desirable structure, we typically divide the information elements into subsets that are intended to reduce the scope of search for the anticipated transaction load. These subsets are characterized on the basis of information content of their elements (for example, all representations containing the same department number) and the characterizations are stored with directions for getting to the location of the characterized subset. This characterization process can be cascaded in the sense that the characterizations can also be collected into characterized subsets (for example, all the department numbers in a corporate division).

The search process is to scan the characterizations of the highest level subsets and to access and scan only those subsets that are necessary to obtain the desired information. This process significantly reduces the number of elements to be scanned.

There is also a desire to keep the elements to be scanned close together to minimize search time. The structure that results from this combination of subset definition and placement of subset representations close together is usually called *file organization.* A well-designed file organization provides a way of immediately processing transactions one by one, thereby, allowing us to keep an up-to-the-second stored picture of the real-world situation.

Historically, the file-organization area started with special cases of techniques implemented for specific applications and devices. Each implementer developed his own way of defining, placing and interconnecting subsets, and invented his own terminology for his unique combination of primitive processes. The result stands today as a proliferation of inconsistent and incompatible, special-case terminologies.

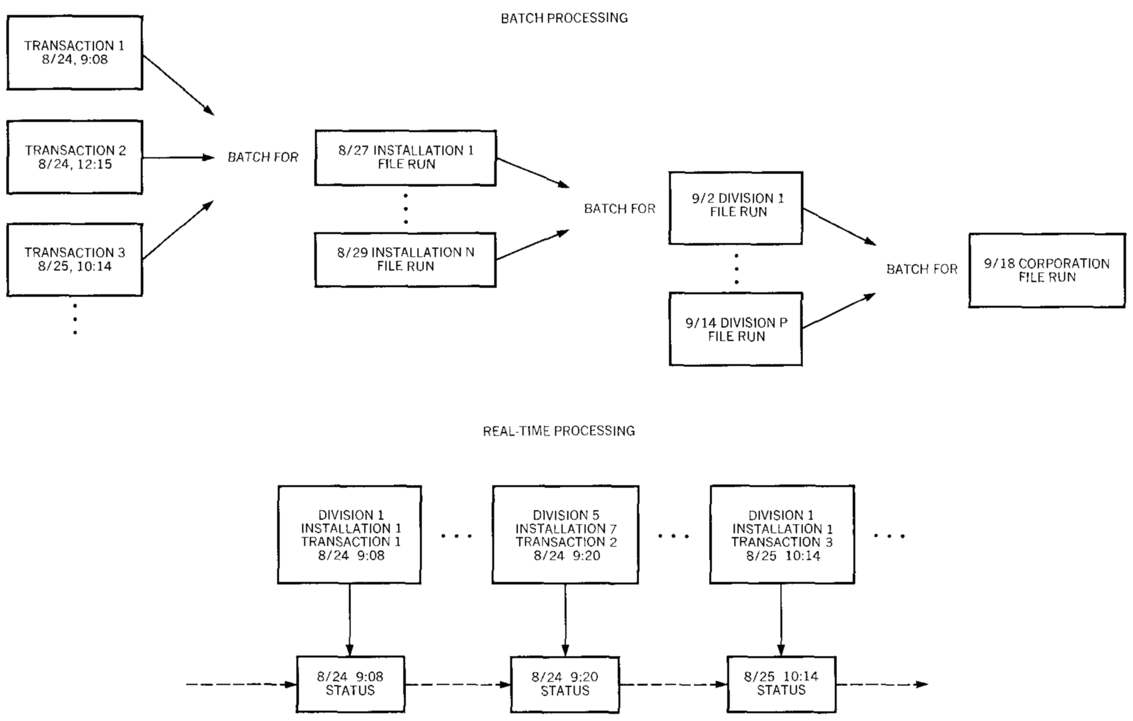
Later, some partial generalization has taken place in which specific implementations have been given new functions (and additional terminology) to process a wider range of applications. The present situation is that we have a somewhat smaller set of rather more general file organization implementations, each with its own terminology for the specific combinations of the basic underlying primitive processes that it provides. Typical of such terminology are the following terms: non-repeating groups, physical hierarchy, logical records, files, data sets, and indexed sequential organization. When the same term is used in two different systems, it often has a slightly different meaning in each system (as, for example, the term "record"). Each time the systems analyst encounters a new system he must learn new meanings for literally scores of complex and poorly related concepts by studying the system in its overwhelming complexity at the bit level. One of the objectives of the Data Independent Accessing Model (DIAM) is to contribute a common set of concepts and terminology toward which integrated data-base information systems can systematically evolve.

old systems—new systems

A segment of the management information systems literature tends to ascribe almost mystical powers and functions to the new real-time integrated data-base systems. The main difference that we see in the new systems is that they can maintain and present an up-to-the-second, consistent picture of an enterprise's resources. Earlier systems could only maintain multiple pictures that were often inconsistent because they were varying amounts of hours, days, weeks out of date with the actual situation. The up-to-date picture allows the management of a corporation to use its valuable resources much more effectively. Figure 1 illustrates a possible timing difference between a batch processing and a real-time processing system.

From the point of view of the systems analyst, the major change has been the gradual integration of individually maintained files from scattered locations and applications into a single, centrally maintained set of files. This integration is necessary to achieve the up-to-the-second, consistent picture of the business. Carried to a logical conclusion, file integration results in a situation where there is only one place in the information structure to store a fact, with multiple representations of the fact often used to aid the search process. Thus there are no inconsistent or out-of-date storage locations. A second factor is the appearance of a qualitatively new software technology for preserving the integrity of the information base. This technology replaces older techniques using physically received multiple tape copies.

Figure 1 Comparison of batch and real-time processing


BATCH PROCESSING
TRANSACTION 1 8/24, 9:08
TRANSACTION 2 8/24, 12:15
TRANSACTION 3 8/25, 10:14
BATCH FOR
8/27 INSTALLATION 1 FILE RUN
8/29 INSTALLATION N FILE RUN
BATCH FOR
9/2 DIVISION 1 FILE RUN
9/14 DIVISION P FILE RUN
BATCH FOR
9/18 CORPORATION FILE RUN
REAL-TIME PROCESSING
DIVISION 1 INSTALLATION 1 TRANSACTION 1 8/24 9:08
DIVISION 5 INSTALLATION 7 TRANSACTION 2 8/24 9:20
DIVISION 1 INSTALLATION 1 TRANSACTION 3 8/25 10:14
8/24 9:08 STATUS
8/24 9:20 STATUS
8/25 10:14 STATUS

The systems analyst is concerned with both the information organization and its representation. At present, he must discuss both in terms of representation based terminology. For the future, it is our objective to provide him with an information-based terminology to stabilize at least part of his terminology environment.

implementation

In discussing implementations, it is useful to distinguish two types of information system transactions that, until recently, have been served by two relatively different kinds of direct-access information systems.

1. Retrieval of a representation that describes a single entity is performed on the basis of exact match to a single unique identifier for that entity such as man number, flight number, or part number. The representation provides the full answer to the request as illustrated by Table 1 transactions M1, M2, and P1. We shall call systems where this type of retrieval predominates *operational systems*. Most of the computing systems in the world are involved in this type of retrieval. At the present time, these systems are mainly operating in a sequential batch mode. In the following, an example of an advanced system that serves a thousand or more geographically distributed terminals is presented.

2. Retrieval of one or more representations for analytical purposes based on an exact match to a Boolean qualification specification such as WHERE Weight > 2 AND Color = Red as shown by transactions M3 and P2 in Table 1. The set of representations provides the full answer to the request. We shall call systems where this type of retrieval predominates *executive systems*. Probably hundreds of computers are involved in this type of retrieval.

operational systems

Although operational systems are found in almost all environments, some of the largest systems may be used for controlling seat inventories for the airlines and money transactions for banks. Smaller systems are used for parts inventories of manufacturers and for utility business offices.

A typical retrieval is based on an information representation identifier such as, "Give me all your information on Part Name = GEAR." Part Name is a unique identifier, and the stored information representation might include Weight, Quantity on Hand, Color, and Name, all stored in one physical location. Having retrieved the information, the user may then wish to modify some field by entering, for example, "Change Color to Red." The system then replaces the modified representation in its original position. The file organization can be quite simple, using, for example, a hash coding or an index on the identifier (Part Name). In fast-response systems that handle slightly more complex questions, indexes might also be maintained on one or at most two other fields in the file, such as Color, for example.

Whereas the retrievals are simple, the numbers of retrievals and the sizes of the fields may be very large. An example operational system is the IBM Advanced Administrative System (AAS) which is used to control computer orders, inventory, and accounting. In the configuration described in Reference 4 there are approximately fifteen hundred CRT terminals located in branch offices, plants, and headquarters across the United States. The central complex at White Plains, New York includes four System/360 Model 65 computers for message processing and two Model 85s for managing the data base stored on approximately forty 2314 disk units with eight drives each. The total storage capacity of the system exceeds 2.5 billion characters allocated to over 20 million data representations and 27 million index representations. There are approximately twenty applications, including order entry, delivery scheduling, and payroll, which require about 450 types of transactions. Terminal interactons with these programs generate up to fifty inputs per second, or one-and-one-half million inputs per twelve-hour day.

Possible types of transactions are rather limited in operational systems, and this situation allows the systems designer to preprogram and precompile transaction programs, thereby trading flexibility for transaction processing speed. Terminal users in these systems follow strict protocols in requesting retrievals. The user is led through a series of preprogrammed multiple choice displays by which he supplies information that is transmitted back to the computer through the telephone network. An appropriate program processes his information and, if necessary, accesses the data base with one or more identifying keys to retrieve or store data-base information representations. These protocols can be implemented in both operational systems and executive systems. In both cases, the protocols tend to restrict the damage that can be caused by the entry of incorrect transactions.

In an operational system, information maintenance and presentation consist of retrieving or storing a very few representations per transaction. The major challenges lie in developing fast access paths tailored to the anticipated transaction pattern, protecting the integrity of the data, recovering from errors, and providing adequately balanced computing, transmisssion, and data accessing capacity to handle the enormous loads.

executive systems

Executive systems generally perform more complex analyses of information for long-range planning and, therefore, need not be as up-to-the-second as operational systems. In fact, information representations stored in such systems may be extracted on a daily or weekly basis from data bases maintained by operational systems. A typical executive system retrieval is based on the content of one or more information elements in the stored representation, such as, "Give me the Average Age of all cars with COLOR = Blue and MANUFACTURER = Dodge." To obtain the desired information, the executive system accesses each representation where COLOR = Blue and MANUFACTURER = Dodge. Even with the most efficient access structure, the system may have to access hundreds or thousands of representations. Because each question is composed anew, preprogramming generally cannot be used. Also each retrieval may involve large numbers of representations. These two facts distinguish executive systems from operational systems.

Executive systems do not often reach the large sizes typical of operational systems. An example executive system[5] that we have studied has the following characteristics:

78 files

150×10^6 characters

50 terminals

2000 transactions per day

1 System/360 Model 65

3 IBM 2314 Disks with 8 drives each

Even though the rate of transactions is low, each transaction tends to be more complex than that of this operational system. In fact, the executive system must read approximately one million representations per day. This gives an estimated transaction complexity factor for the executive system of ten to one hundred times greater than that of the operational system. Since transactions in an executive system take many forms and normally cannot be preprogrammed, the questioner uses a simplified query language to specify his information requirements. Thus language interpretation or compilation contribute further processing unit overhead per question not required by operational systems. Executive systems that have been proposed or built include the Time Shared Data Management System (TDMS)[6] by the System Development Corporation, the World Wide Military Command and Control System (WWMCS) proposed by a number of manufacturers, GIS/360[7] and MIS/360[5] by IBM. A batch-oriented system, MARK IV,[8] by Informatics, Inc. has also been popular.

directions for investigation

In practice, operational and executive systems are but extremes of a continuous spectrum of possible computerized information system implementations. Most installations have some characteristics of each type of system and, therefore, fall somewhere between the extremes. At the present time, however, one or the other of the two types of system implementation are used in the implementation of intermediate systems.

The primary area for research and development in the computerized information systems is improving their ease of implementation and use. It is well-known that implementations of large real-time information systems can require three to five calendar years and hundreds of man-years of effort. Much of the system development cost is not in the hardware, but rather in the designing, creating, maintaining, and using software. If the use of computerized information systems is to continue to grow and spread, then the nonproductive personnel costs involved in program deciphering, program maintenance, and housekeeping activities must be reduced.

One way to attack the problem is to add more power to existing systems. Essentially, this is a process of evolution by adding more function and modifying existing conceptual structures. Existing or proposed product programming systems are generally oriented to this type of progress. The Data Base Task Group (DBTG) Report[9] from CODASYL is essentially an extension of COBOL to give it more power in handling data structures. IBM's Information Management System (IMS),[10] Generalized Information System (GIS),[7] and Customer Information Control System (CICS)[11] are evolutionary programs designed to reduce implementation costs by providing generalized facilities that can substitute directly for the most difficult fifty or more percent of the application implementation code. The other software producers previously mentioned have essentially the same goals. These systems, although already essential to the economic implementation of on-line information systems, address only one aspect of the system programming problems, that is, displacing the need to write major sections of the information system program. We would, of course, like to provide a compatible path for them to evolve to systems with new ease-of-use properties. This involves a second method of attack.

The Data Independent Accessing Model, which is the main subject of this paper, is characteristic of this second method of attack on the ease-of-use problem—that of simplifying the conceptual structure of information systems and making them more rational and powerful. Information systems at times do not make good common sense, and they do not appear to the user to operate in a common sense way. For example, changes in the organization and storage of information representations require major changes in application programs even when logical relationships of the information remain unchanged. (Fifty percent or more of the programming effort in many installations is devoted to the nonproductive work of adapting old programs.) The emerging concept in this area is to have programs address and use the more permanent logical relationships between information elements and to find ways of structuring these information relationships so that they remain even more permanent. A vital corollary of this concept is to have a complete separation between the logical relationship structure and the means for representing it. Thus representations can be changed without affecting programs that deal with the logical structure. This concept, often called "data independence," is more accurately termed "data structure independence," and is an important element in the task for making information systems easier to implement and use.

There is also in this second area of attack the question of a fundamental step forward in our conceptual structure for information systems. Existing systems have grown and become somewhat generalized through evolution from application programs. Their terminologies have the appearance and limitations of botanical or biological taxonomy in that they describe external features, when what is needed is a theoretical description of system's work in terms of primitive building blocks and the properties of block interaction. None of the existing terminologies are a basic in the sense that one system's concepts are able to describe, with useful accuracy the concepts and processes of any significant group of other systems. Because we have in the past lacked such a general standard of description and comparison, we could not progress by incorporating desirable new properties and the best aspects of many existing systems. In a very real sense, a general description covering the existing systems is a requirement to any new conceptual framework. Over the long term, the conceptual framework must provide a basis for easy migration to new improved systems. It can only provide the basis if it can encompass the capabilities of the existing systems. In the second part of this paper, we address the question of information organization to provide insight for defining primitive building blocks for data structures and data accessing.

CITED REFERENCES

1. E. W. Dijkstra, "The humble programmer," *Communications of the ACM* **15**, No. 10, 859–866 (October 1972).
2. J. T. Tou, Editor, *Advances in Information Systems Science* **2**, Plenum Press, New York, New York (1969). See contribution by M. E. Senko.
3. M. E. Senko, "File organization and management information systems," *Annual Review of Information Science and Technology* **4**, 111–143, C. A. Cuadra, Editor, Encyclopaedia Britannica Company, Chicago, Illinois (1969).
4. J. H. Wimbrow, "A large-scale interactive administrative system," *IBM Systems Journal* **10**, No. 4, 260–282 (1971).
5. G. F. Duffy and F. P. Gartner, "An on-line information system for management," *AFIPS Conference Proceedings, Spring Joint Computer Conference 1969* **34**, 339–350, AFIPS Press, Montvale, New Jersey 07645 (1969).
6. R. E. Bleier, "Treating hierarchical data structures in the SDC Time Shared Data Management System TDMS," *Proceedings of the ACM 22nd National Conference*, 41–49, MDI Publications, Wayne, Pennsylvania (1967).
7. *Generalized Information System GIS/360, Application Description Manual (Version 2)*, Form GH20-0892-0, International Business Machines Corporation, Data Processing Division, White Plains, New York 10604 (1970).
8. J. A. Postley, "The Mark IV system," *Datamation* **14**, No. 1, 28–30 (January 1968).
9. CODASYL Data Base Task Group, *Report to the CODASYL Programming Language Committee*, Report CR 11, 5(70)19,180, ACM, New York, New York (October 1969).
10. *Information Management System IMS/360, Application Description Manual (Version 2)*, Form GH20-0765-1, International Business Machines Corporation, Data Processing Division, White Plains, New York 10604 (1971).
11. *Customer Information Control System (CICS), General Information Manual*, Form GH20-1028-2, International Business Machines Corporation, Data Processing Division, White Plains, New York 10604 (May 1972).