32

Data bases

George Schussel*

A new concept in business data processing is achieving significant acceptance by both computer manufacturers and major users of computer systems. This new technique uses standard "data base management system" (DBMS) software packages to implement business data processing on a "data base" (DB). In mid-1975 the number of DBMS users in the United States was estimated to be approximately 2000. DBMS availability offers the possibility of reducing the long-term costs of data processing and of increasing the capabilities of the business programmer by automating many of the functions now performed manually by application programmers. It allows for more programmer productivity and provides for broader systems control. The DB concept has so developed over the last decade that it can be implemented by most medium- and large-scale users of data processing equipment. As the evolution toward data base becomes universal, companies will consider recasting their systems to achieve these advantages. To enjoy these benefits, however, the user will have to invest in more hardware and software overhead.

In the 1960s, the COBOL programming language emerged as the standard language for business systems; in the 1970s a combination of data bases and COBOL is becoming the standard procedure for implementing business data processing systems.

The DB concept involves software packages (usually called DBMS) that remove many of the functions of the business programmer from his direct control and vest those functions in the DBMS package. The package standardizes all information for the systems application while supplying it to the programmer in a more retrievable form.

Historically, business data-processing files have been oriented

* Vice President, American Mutual Liability Insurance Company, Wakefield, Mass.

toward storing information on tape. Each application had its own master file containing its own information. This file often contained data which was repeated on other master files.

The DB concept stands in stark contrast to this. It says, in effect; "Let us put all our eggs in one basket. Instead of storing identical information in different places, we'll put it all in one place. Everyone will know where it is and how to get it; it will be on an accessible piece of equipment and we'll control updating."

Standard file maintenance implementations have always had the application program directly maintaining and accessing the basic data. Now the DB approach structures business implementations like a sandwich with the application program sitting on top, the DB at the bottom and the DBMS in between providing the links between the two.

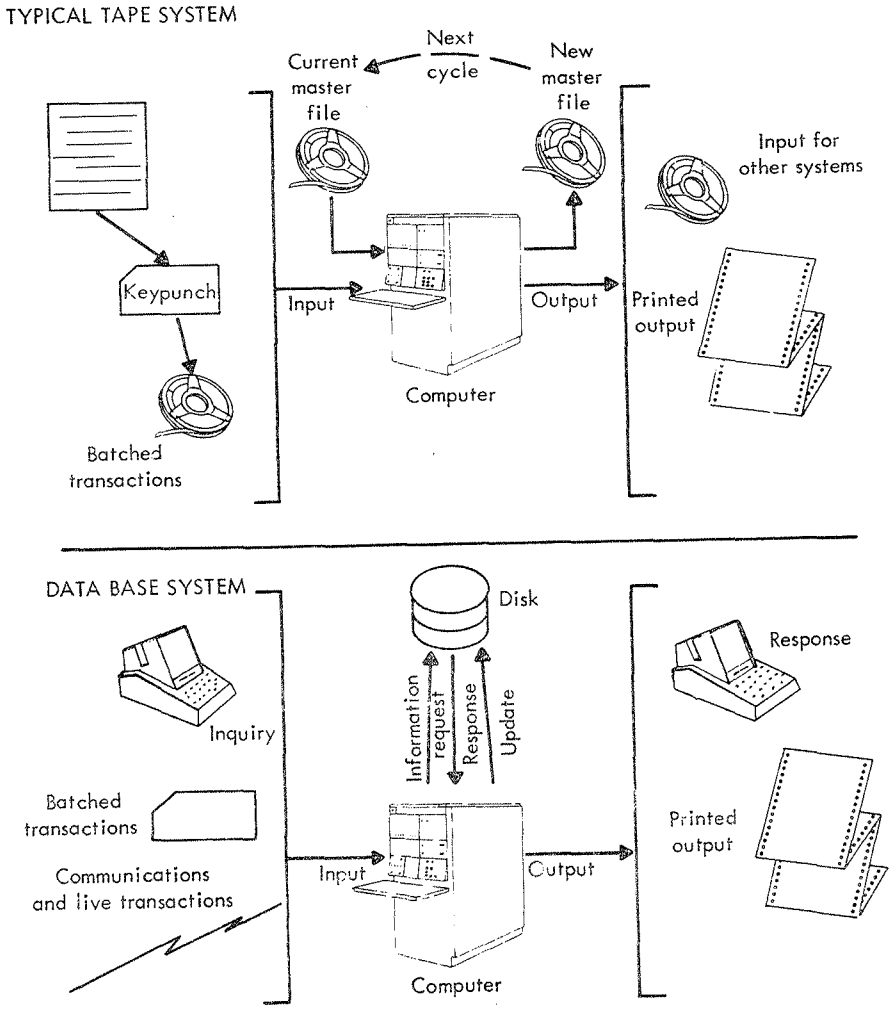
Since in the former technique, the structuring of the data has always been with the goal in mind of achieving optimum performance for retrieval of that data for the *one* particular application that used it, the reuse of the same data for other applications has been difficult or impossible. As a result, when new needs for this same information came along similar data was restructured in files which fit the next application, and so forth. The end product after a period of years with multiple developments has always been much data redundancy and the attendant expensive problems of much storage, difficult access, and poor control.

The DB approach is "data" rather than "program" oriented and builds the one DB to provide good accessibility to data for *all* of the applications.

Figure 1 contrasts the DB approach with tape systems. In the typical tape system, information is keypunched into computer format. Next, a number of similar transactions are batched. Periodically, these are put into the computer system along with the program which processes these data. At this time the current master file to be updated is loaded. The application program residing in the computer then matches the transaction against the master file, produces the output reports summarizing these transactions, produces input for other systems, and produces a new, updated master file. This updated master file becomes the "current master file"; it will be used in the next processing cycle for this application. The previous "current master file" becomes what is called a "father tape."

This processing technique contrasts with a DBMS processing on a DB system. Many of the transactions entering the DB will evolve similarly to the way they are currently developed in the tape system, from paper to keypunching and batching. However, other transactions and inquiries will come through more immediate on-line techniques such as local terminals, communications with other computer

FIGURE 1
Data base approach contrasted with a typical tape system



systems or remote users, and computer-assisted instantaneous monitoring of transactions (as in point-of-sale monitoring of the checkout register at a retail store).

The programs required to process the input and the master files against which input will be processed are located on disk storage and are available in thousandths of a second. Data entering the computer are analysed for what program is needed for processing. This program is called into the primary memory from the disk and is activated. As the program proceeds, it will require certain data off the master file.

Although the entire master file exists on disks, only a small portion of it is related to any single program. Therefore, only necessary pieces of DB information are entered into the primary memory, updated, and returned to the disk.

There are two ways to communicate with other systems. Either the input data are kept in the computer and other application programs are brought in to use the same information, or intermediate storage files—on the disk or sometimes on tape—are created. These subsequently serve as the input for updating the other systems.

The output from the DB computer system is often similar to output reports from regular tape-oriented systems. In addition, however, the on-line characteristics provided by the DBMS enable some responses to be provided directly to teletypewriter or cathode ray tube terminals. Only seconds elapse between the inquiry and the answer.

Companies now utilizing time-sharing or interactive man-computer systems already appreciate some of the advantages of the DB approach. These interactive systems permit input and response from inquiry terminals (or through the use of communications) in the same way that DBMS's do. However, the interactive system is only one half the answer, since the data necessary for a particular application typically require special loading onto a disk. If it is already there, it is not interfaced with other applications for uniform use and updating by several systems. In other words, the DB approach will combine several master file updates into one operation.

Recent developments

The advent of the large-scale disk, with its capacity of 100 million characters on one removable disk pack, was the engineering breakthrough which made practical the DB concept. It is now possible to retain up to several billion characters at any one time on-line to a computer, and to summon any piece of that information in less than one tenth of a second.

The concept of a total DB on a direct-access device such as a disk drive is not new. Initial approaches were tried on second generation computers with smaller, slower disks ten years ago. These approaches were not widely adopted because they did not employ DBMS, standard programming packages, to do much of the tedious, yet necessary, programming housekeeping. These packages sell or lease for a small fraction of the cost of writing such a system. Several DBMS packages are available in today's market. Table 1 lists several popular DBMS's. In the Appendix there is more detail on five such systems. For EDP Departments about to undertake a DBMS procurement, Figure 2 presents 23 criteria which can be used as a checklist in an evaluation of systems. Surveying the total number of users of

TABLE 1
Data base management systems

DBMS	*Vendor*	*Address*	*Computer*	*Purchase price*	*Complexity level*
IMS	IBM	Armonk, N.Y.	IBM 360/370	Lease only. $600/$2,000 per month	High
System 2000	MRI	Austin, Tex.	IBM 360/370 CDC 6000 Univac 1100	$35,000/ $160,000	Medium
ADABAS	Software AG	Reston, Va.	IBM 360/370 Univac 70	$120,000	Medium to High
DMS	Xerox	El Segundo, Calif.	XDS Sigma	Bundled	Low
IDMS	Cullinane	Boston, Mass.	IBM 360/370	$37,500	Medium
GIM	TRW Systems	Redondo Beach, Calif.	IBM 360/370 Univac 1100	Lease. $2,000 per month	Low
TOTAL	Cincom	Cincinnati, Ohio	IBM 360/370 Univac 70 Honeywell 2000	$25,000/ $35,000	Medium
IDS	Honeywell	Wellesley, Mass.	H-6000, H60	Bundled	Medium
DMS/II	Burroughs	Detroit, Mich.	B6700/7700	Bundled	Medium

FIGURE 2
Checklist of characteristics for the ideal DBMS

1. *Automatically convert existing files into DBMS control.*
   Some systems have the capability of taking existing files such as tape sequential or disk index sequential and automatically converting them to control of the DBMS without a large amount of manual intervention or without re-keypunching the data.

2. *Minimum restrictions on structures.*
   DBMS capabilities vary, from some having only one data structure to others having multiple data structure capabilities including chains, hierarchies, lists, networks, rings, etc.

3. *Free form and highly adaptable inquiry system.*
   A query language (batch or on-line) is one of the more useful features of a DBMS. In many cases, it will obviate the need for producing long reports and allow access of files on a when-needed basis from a terminal. It should have the ability to access individual records or subsets on the basis of a number of different criteria or interrelationships among such criteria (Boolean Retrieval).

4. *On-line and batch modes simultaneously.*
   At the current time, most DBMS usage is in the batch mode; more on-line, terminal interaction will be used as time goes on, and the ability to have both modes interactive with the same data base simultaneously is useful.

5. *Efficient and automatic checkpoint restarts.*
   Any system will occasionally crash. The ease of recovery from such a crash is an important evaluation criteria.

6. *Multilevel security.*
   Security of the data base is very important since multiple users will be interacting with one data base. This represents a security problem of substantially greater scope than is present in most second generation tape-oriented systems.

7. *Data or program modification with minimal interdependency changes.*
   An important characteristic of the relationship of the application program to DBMS is data independence. To the greatest extent possible, changes should be allowed in the application program or in the data base without requiring housekeeping changes in other parts of the system.

8. *Random or direct access capability.*
   Many files of information can be processed most efficiently through random, direct access techniques. These are typically files that exhibit low processing hit ratios.

9. *Handle sequential processing requirements.*
   Conversely, many files are such that sequential provides the most effective technique for access. An example is processing which involves a very high hit ratio—such as might occur in the generation of reports.

10. *Allow inverted list processing.*
    The use of the inverted list which takes advantage of abstracting and indexing on the basis of individual keys has proven to be a very efficient technique for retrieving records with multiple keys. Capability should be available for many (1 — n) keys for any record or file.

11. *Automatically handle the administration of data storage devices.*
    A single data base may be stored on several different types of peripheral storage gear such as disks, drums, and tapes. System capability to handle all of these types should be available.

(*continued*)

FIGURE 2 (*continued*)

12. *Reentrant capability allowing for multiprogramming applications.*
    This capability will allow one copy of the DBMS to reside in main memory while several different application programs accessing one or more data bases are run under control of the same DBMS. If the code is not "reentrant" then a separate copy of the DBMS must be brought in for each program running at one time.

13. *Use of data compression or other techniques for efficient storage (minimum additional space overhead).*
    Much information in any data base consists of lengthy strings of nonessential or redundant information, such as leading zeros. Data compression techniques squeeze down the amount of space required to store such information into minimal amounts to save on storage requirements.

14. *Allow data base inquiry through procedural languages.*
    In addition to its own language, if any, interface to the data base should be respond to many types of inquiries.

15. *Allow data base inquiry through procedural languages.*
    In addition to its own language, if any, interface to the data base should be allowed for languages such as PL/1, COBOL, and Fortran.

16. *Multithreading allowing several accesses to the same file in parallel.*
    Multiple accesses into a single data base region should be allowed on an overlapped basis rather than forcing each request for information to be completely satisfied before the next is begun.

17. *Minimum DBMS overhead memory—few or no overlays.*
    To the extent possible, it is desirable to implement the DBMS using as little main memory as possible. Overlays for this type of software are not generally recommended since their use will substantially slow the execution speed of the system.

18. *Minimum restriction on the number of fields per record, records per file, files per data base, or hierarchical levels.*
    Obviously, minimal restrictions on any of these important numbers is advantageous.

19. *Allow variable length records and files.*
    More efficient use of main and secondary memory is often possible by the use of variable length files.

20. *Be close in specifications to the CODASYL data base task group recommendations.*
    There is high likelihood that some time in the future U.S. Government standardization will result in one or a small number of approaches to Data Base implementation, in the same way that COBOL has been standardized. The closer a DBMS is to the CODASYL recommendations, the closer it will be to such standardization when it occurs.

21. *Allow multiple roots for a hierarchical data base.*
    In terms of defining an individual data base, multiple root segments for the data base are desirable as a capability; this is translated as "allowing many children for one parent and/or many parents for one child."

22. *Minimum reorganization with growth.*
    As the data base grows it should not require reorganization to maintain efficiency or such reorganization should be performed "on the fly" automatically by the system, or it should be as easy to do as possible.

(*continued*)

FIGURE 2 (*concluded*)

23. *Have RPG capability.*
Allow the use of a high-level report generator to format reports from the data base. This allows nonprogrammer users such as managers to have access to the DB for queries, analyses, modeling, or browsing. When combined with characteristic 3 (inquiry system) this facility becomes on-line. Also, ad hoc or one-time reports are greatly facilitated by RPG capability.

DBMS's leads to the estimate of 2000 U.S. companies with active DB systems in mid-1975.

The DB approach does raise real questions for management. The answers below reveal both the advantages and the disadvantages of the system. In addition, it has become readily apparent that a DB system necessitates a specifically qualified DB Administrator. "Is a DB system something that we should be considering right now?"—this question now must be answered by both EDP and top management.

Data base characteristics

The first important advantage in the DB approach is that the corporate DB contains all pertinent information that can be correlated and cross referenced. The overall plan of this total corporate approach can be organized in several different ways. For example, the operation of a typical insurance company could be divided into two parts. The larger area would be the actual insurance operation: premium collection, claim processing and payment, accounting and controls, and administration, including payment of expenses. The smaller segment concerns investing premiums to provide a steady income and capital appreciation. If the operation of these two areas is clearly segregated, there may be no need to correlate investment information with insurance results except in the annual statement where all results are gathered—usually manually. Given these definitions, investment information should dwell in a separate DB. Considering the limitations of today's computer hardware, this usually saves money.

A necessary criterion in establishing the DB—and one of its most essential characteristics—is that creators and users must agree on a common set of definitions for all information in the DB. This is not so easy as it might seem. Examination of data definitions in almost any medium- to large-scale user of data processing immediately exposes three classic problems: (1) synonyms, or identical items of data called by different names in different applications; (2) alternative definitions, or different systems using the same name to describe two different pieces of data; and (3) close definitions, or two different names used to describe different pieces of data with definitions so similar that there should be only one name and one definition.

The above characteristics fundamentally imply that a DB is one accumulation of data, used by multiple users. This characteristic, in fact, is often used as a basic definition of a DB and this gives the DB one of its primary advantages—the open-end nature of potential uses for the data. In the typical file orientation where data is tied to one application program, it is usually difficult and awkward to use the same data for other purposes. The DB approach, however, by leaving further uses potentially open encourages more applications using the same data. This use of the same data by multiple users does, however, cause some problems as are discussed below under "Problems with the Approach."

To benefit from the advantages of DB, the corporate user must be willing to surrender his esoteric definitions in order to live with a common set acceptable to all. First, the designers must develop a complete dictionary of definitions. After they obtain the support of the users for these definitions, they must then keep it current and enforce it for new applications. The use of a data dictionary will help most users keep definitions current and properly cross correlated. Computerized data dictionary systems consist of several reports providing information to programmers, analysts, and managers, while simultaneously serving the function of documentation within the EDP Department. In order to be most useful, a data dictionary should be integrated with the function that creates the data base (the data definition language). When independent data dictionary products are used, the result is often that the data dictionary gets out of synchronization with the actual data base, compromising the usefulness of both.

The next essential characteristic of the DB is that it resides in a secondary storage device, usually a disk drive. Primary memory is the very fast and very expensive magnetic core or integrated circuits. The essential benefit of the disk drive or other secondary device is that while much cheaper per character than main memory it still renders all data quickly accessible compared to tape or card storage. It can take a computer 150 times longer to locate a piece of data nearer the end of a tape reel than the beginning. The DBMS retrieves data off disk more swiftly, and in desired sequence.

Although disk drives are the most common direct access storage devices, either drums or magnetic cards or data cell devices are also used. Drums have faster access times than disks, but they are much more expensive on a per-character-stored evaluation. Magnetic cards or data cells are much slower but cheaper. Neither is in general use for DB. By 1980, completely new, massive random-access-memories (RAM) based on new technologies, such as molecular physics (bubble

memories) or optics (laser memories), should make electromechanical disks, drums, and data cells obsolete.

Another essential feature of the DB system is that it can be interrogated by on-line terminals, usually cathode ray tubes. One of the primary advantages of the DB approach is that by concentrating all of the information sources into one location, it often is possible to interrogate specifically from the DB in lieu of printing lengthy volumes of output which are stored and perused whenever a question arises. Converting a basic DBMS into an on-line system may be done for reasons such as:

1. A desire to interrogate the data base on a real-time basis through the use of a query program.
2. A desire to update information in the data base from a remote location through a terminal.
3. A desire to enter and run programs against the data base from a remote site.

TABLE 2
Some well-known teleprocessing monitors

TP monitor name	*Vendor company*	*Main memory required*	*DBMS used with*
ENVIRON/1	Cincom Systems Cincinnati, Ohio	50K Bytes	TOTAL
INTERCOMM	Programming Methods New York, N.Y.	50K Bytes	TOTAL, IDMS IMS, ADABAS
CICS	IBM Corp. Armonk, N.Y.	100K Bytes	TOTAL, IMS, DBOMP
IMS/DC	IBM	160K Bytes	IMS
TASK/MASTER	Turnkey Systems Norwalk, Conn.	40K Bytes	TOTAL, IMS

All of the TP monitors listed have been designed to run on any Model IBM 360 or 370 computer that has a large enough main memory for the TP monitor and application. Standard vendor-supplied interfaces are available for the DBMS's listed in the rightmost column.

For this type of capability, the on-line feature is needed. Although it is possible to program your own on-line interface to a data base, such interfaces are usually implemented through the use of teleprocessing (TP) packages, which interface to the DB through the DBMS.

These interfacing TP packages are available from a number of vendors. A number of the most popular TP packages are listed in Table 2. Figure 3 lists some desirable characteristics that should be analyzed before purchase of any TP package. Besides interfacing the DB with a TP package, many users have acquired report program

FIGURE 3
Desirable teleprocessing monitor characteristics

1. *Support all types of terminals.*
   Different terminals such as CRT's, Teletypes, or high-speed line printers may be interfaced to the same TP system, each having its own advantages under certain circumstances. It is desirable to have the TP monitor support as many competitive terminals as possible.
2. *Run under any operating system.*
   Some computers come with several different operating systems or master control programs. As the user moves from one operating system to another, it is convenient and desirable to be able to use the same TP monitor.
3. *Interface with all high level languages.*
   Support and interfaces for languages such as Fortran, PL/1, and COBOL is obviously important.
4. *Minimum main memory utilization.*
   The less overhead is required for resident TP code in main memory, the more memory is available for applications.
5. *Handle message switching.*
   "Message switching" relates to a set of functions performed by a computer hardware/software configuration, where at a central point a computer accepts, stores, and forwards messages from multiple sites as a standard application.
6. *"Bundled" support and installation.*
   Installing the TP monitor may be a complex technical undertaking which requires quality on-site support from the vendor.
7. *Interface with the DBMS.*
   A standard support interface which will interface with the DBMS that is being used to control your data base is essential.
8. *Interface all files and access methods.*
   You may want the TP monitor to access separately set up and controlled files that are not under the DBMS.
9. *Dynamic run-time system control.*
   A capability should be available from the computer site, or remotely, to make decisions which impact the capability and configuration of the system as it is running to properly respond to transient conditions.
10. *Have a good assortment of debugging aids.*
    Debugging program errors in the real-time environment is one of the most difficult tasks in programming. Techniques such as simulation, core traces, system traces, selective dumps, and testing routines are useful in such debugging.
11. *Minimum restrictions on key statistics.*
    Message volume, the ability to multithread access the data base, no limit on applications size, ability to run simultaneous applications; all of these factors which should be analyzed with an eye toward minimum restrictions.
12. *Good restart/recovery.*
    Restart or recovery from system crashes in a real-time environment is more complex, requiring more technical knowledge than normally present in a batch environment. Since the system will crash occasionally, the ease of recovery is important.

generators (RPG) which can be used to select, format, and print reports off the DB. The advantage of the RPG package is that it typically requires only one third to one tenth of the programming effort in producing reports using COBOL. Nonprogrammers are often trained very quickly in RPG application. Table 3 lists some RPG products and commonly used DBMS combinations.

TABLE 3
Report program generators (for IBM 360/370's)

RPG	*Vendor*	*Purchase price*	*DBMS used with*
SYSTEM 2000 ..... (report writer)	MRI Systems Austin, Tex.	$15,000 extended $10,000 basic	SYSTEM 2000
SOCRATES.....	CINCOM Cincinnati, Ohio	n/a	TOTAL
CULPRIT.....	Cullinane Corp. Boston, Mass.	$17,500 plus 10%/year for maintenance	TOTAL, IDMS IMS
ASI-ST	Applications Software Torrance, Calif.	$33,000 perpetual license $—3,000 maintenance per year after first year	TOTAL IMS
MARK IV	Informatics Software Products Canoga Park, Calif.	Varies, based on features. $7,500 to $35,000	IMS, TOTAL
GIS/2..... (super rpg)	IBM Corporation White Plains, N.Y.	n/a $450 to $1,500 monthly license	IMS

The final key characteristic of a DB is the concept of multiple users. Users from different departments and for different application programs will be using one common DB. This has several important implications for the company electing this approach. Two of these—common data definitions and the data element dictionary—have been discussed. Others, such as the need for more emphasis on security and the need for creation of a totally new position, the data base administrator, are discussed below.

Data base administrator

The DB approach is oriented to data or information. This concept differs from most current business data processing which is geared to applications, and treats data as an adjunct to programs implementing those applications. Now, however, the definition and creation of the corporate DB is primary while the various application programs, which select needed data and operate against the shared total DB are secondary. This stresses the importance of the way the information is created, maintained, defined, and handled. Many DB benefits result from requirements placed on the systems department for the establishment and maintenance of the total DB.

Establishing the data base administrator function (held by one or more individuals) has been essential for successful DB implementation. The data base administrator's job falls into three broad categories. First of all, as custodians of the DB, they are responsible for the definition, control, integrity, and uniformity of the basic data itself. They are normally involved in the development of the data dictionary and the building of the DB on a computer file. As the DB evolves over time, it is the Data Base Administrator who controls its evolution by controlling the flow of new data elements into the DB and culling no-longer-needed ones.

The second primary area of concern for the data base administrator is the data structures, file organizations, and access methods that are used by the DBMS in storing and retrieving information from the DB. Many DBMSs offer several different indexing methods which can be used in storing data within one or more hierarchies on one or more different types of storage devices. By carefully analyzing the relationships of data and its use in different applications, the data base administrator will be able to pick the proper type of file organizations and indexing methods to minimize the amount of time that is spent "thrashing" around in the data base to retrieve information. By carefully monitoring DBMS statistics on data use, data relationships, frequency of access, and total volume, the administrator can suggest new file structures or access methods, can optimize the data locations on different peripheral devices, and can suggest new definitions or the elimination of unused information. For example, some DBMSs permit structuring of the total DB on several peripheral devices. A drum peripheral generally demonstrates high cost per piece of information stored but offers swift access when compared to a disk. Therefore, data which are needed frequently during the day may be more efficiently stored on a drum.

By determining the frequency of needed access, the response times required, and the mode of retrieval, the administrator is able to optimize the use of the hardware for achieving best results through the DB. By properly tuning the DB, therefore, the data base administrator will achieve great economies in program run-time. Needless to say, this important job requires competent technical skills in order to be handled effectively.

The third broad area of responsibility for the data base administrator is the DBMS itself. There are many analogies between a DBMS

and the computer operating system—the primary control program. The DBMS must be "generated," must interface to the operating system, and must have a set of standards for proper interface with application programs. The maintenance of the DBMS also falls to the data base administrator.

In reviewing the above responsibilities, it is obvious that some of the data base administrator's responsibilities are of the systems design nature and some are of a production mode. The person or group performing this job will have substantial responsibilities both before the system goes live and after it is running in a production mode. For example: Once the DB is established, the administrator is responsible for maintaining its security and for developing procedures for recovery from disaster. With all corporate information eggs in one basket, it is extremely important to prevent unauthorized or improper access to the DB.

The person (or group) occupying the data base administrator position should possess a good technical background in disk hardware and software, and if possible, have had some experience in writing operating system software. In the MIS or EDP environment, this administrator would normally report directly to the overall manager of the company's information systems or to the manager of systems and programming. Any point lower in the systems organization will not give sufficient visibility to the importance of this function.

Some "experts" suggest that the data base administrator not be part of the MIS or EDP department. If not located in EDP, then the alternatives are for the administrator to work for a user, such as a division, or to be in a centralized spot other than EDP, such as Corporate Headquarters. To exist in a user division seems directly in conflict with the basic goals of the data base administrator's job. Since the effort should be to globally optimize the construction of the data base for all users, working for one of those users would seem to be at odds with this primary objective. As far as being located outside of the EDP department but in a corporate office, it seems that such an approach might be possible, but in a normal information operation the functions of the data base administrator are so highly integrated and interdependent with those of other activities in the EDP department, that location in the EDP department seems to be the obvious solution.

Advantages of the DB approach

The "total corporate management information system" concept, which was discussed in numerous journals during the second half of the 1960s, proved to be a disaster in the real world. Out of this era, however, the concept of the DB approach to the development of management systems emerged as a clear and viable concept. Because the software evolution has delegated more and more programming tasks to the computer in an attempt to make programming easier, it is clear that a DB tool-arsenal for the programmer will emerge as the next stage.

The most important reason for adopting a DB is to achieve "data independence"—the ability to make changes in application programs without necessary housekeeping changes in the data base and vice versa. It is data independent which allows the second, third, or fourth applications to be implemented just as easily as the first one was on the data base. The ability of severing the umbilical cord that previously tied together data and application programs means that data have their own rationale and meaning and the result is data that are much more widely useful than previously was the case.

Another primary reason for adopting a DB is to realize significant savings in programming and debugging time when developing new business applications. The few studies that have appeared on this subject show that once the DB has been established—by no means a trivial task—implementing comparable programs with competently trained programmers can be two to four times faster than with the COBOL programmer who must develop his own master files. These studies have usually also given the DB the advantage of allowing report programs to be written in a high level RPG type language. Since the cost of hardware in most large data processing departments has dropped to below 50 percent of the budget, and in many cases to 25 percent of the EDP budget, any plan which saves on labor costs provides a persuasive argument.

The third important reason is the efficiency in data storage achieved through the nonredundancy of information that derives from clustering all data in one file. When the information is in one file, controls are more readily applied to it, it can more easily be updated or protected, is more readily accessible, and is, therefore, more reliable.

A seemingly small, but actually very important, advantage of the DB approach derives from the fact that DB information is maintained on line in disk files. In most older environments, tape files are continually being swapped on and off the computer system while being manually controlled. By contrast, in the DB environment, most files are already mounted on the computer, and access to the required information is controlled by the computer itself, resulting in fewer computer operators and, most importantly, fewer associated human errors.

One of the DB benefits most helpful to top management is the ability to process one-shot programs for decision-making or operations research against data already resident in the corporate DB. Because they currently require centralization of numerous tape files, such

programs are either impossible to implement or prohibitively costly and time-consuming in today's typical environment. It is, of course, this type of information which is one of the most valuable products of a data processing department.

Figure 4 is a flow-diagram of a representative DBMS job operating on a DB. The question can be asked: "How does this help solve systems problems?" The problems and the improvements offered by a DB approach are discussed in Figure 5.

Because of these multiple advantages, companies presently using

FIGURE 4
Data base management systems control many of the logical steps in the total EDP run

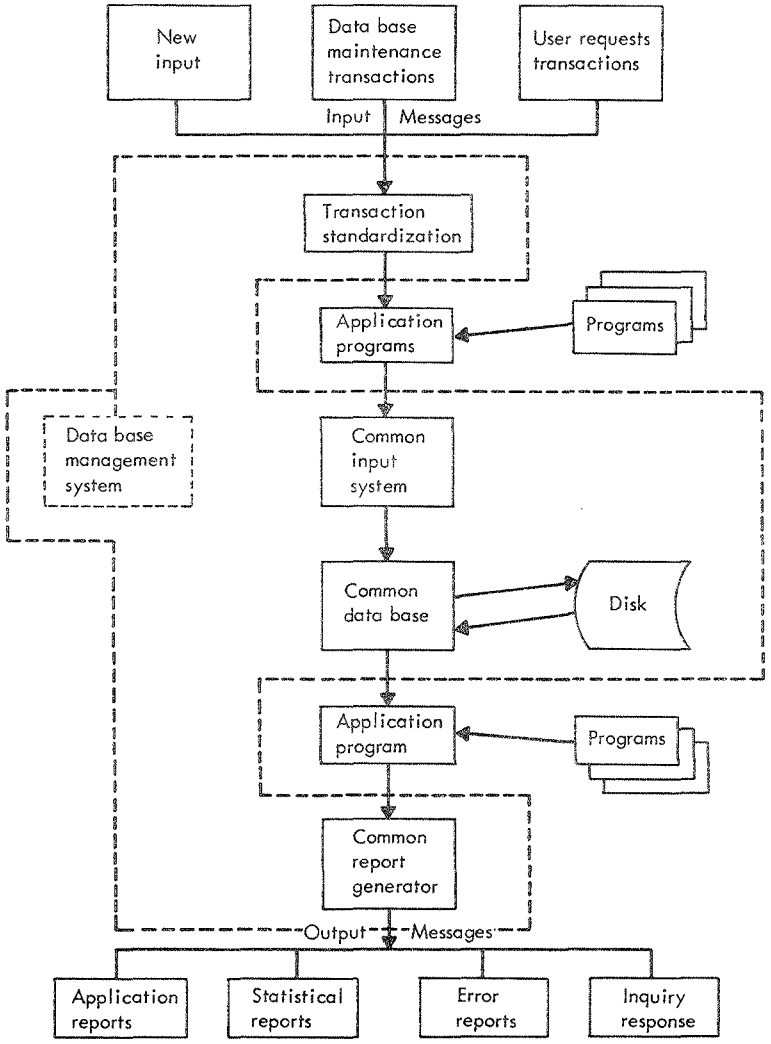


FIGURE 5
Advantages of the data base approach

Systems department problems	Solution with data base
Firm commitments for systems During the design stage of a new system, it is necessary to firm up the specifications so that effective programming can commence. The specifications that have been generated by analysts may be incomprehensible to management personnel. Therefore, even though approval is given for a set of specifications, numerous changes may be required once management sees the final product.	Since DBMS provide an easier mechanism for changing processing and output formats, implementation under this type of system is easier to alter to meet changing or uncertain requirements of management.
Report format As different managers come into new jobs, the requirements asked of data processing often change. Reports change to cover different scopes of information with different formats.	Changes in the information contained in reports and even the scope of the data processing are made considerably simpler through the use of report generator modules available in DBMS.
Acceptable data base In the typical business environment there is difficulty in identifying, defining, and agreeing upon definitions or data items that are repetitively used in different systems.	Instead of having numerous different data bases which were set up at different times with different definitions, the DB is standardized at one time and subjected to continuing review. This procedure minimizes the problems due to this factor.
Data duplication Data that repetitively exist in multiple files cost money because of multiple costs of storage and the resulting difficult control problem.	The whole concept of the DB approach eliminates this problem by storing one value per item in one location only and, most importantly, thereby facilitating improved control.
Master file manipulation The development and maintenance of master files represent the most difficult job facing the business data programmer.	Much of the work required in establishing the master files has already been done once the common DB is established. The application job is, therefore, much easier.
Programming The cost of programming is high and rising all the time. It is difficult to obtain productivity increases in this area.	Because the problem of master file creation has been drastically simplified, the resulting application programming is much simpler and on a higher level with the DB approach. RPG availability is also very helpful in increasing programmer productivity.

(continued on next page)

FIGURE 5 (*continued*)

Systems department problems	Solution with data base
Data interchange among systems The various systems that constitute the total management information system for the company were designed by different people in different locations at different times. This has resulted in significant problems in effectively interchanging data among these systems.	The common data of the DB approach alleviates this problem by permitting a much simpler systems interface.
Interaction with management Top management has difficulty in interfacing with the data processing department because of the specialized jargon used by EDP personnel.	The DB approach is one that is understood by almost all nontechnically oriented top management people.
Modern systems design Availing your systems of the latest technology for on-line interface and readily available flexibility remains a continuing problem.	The DB approach is a modern, flexible approach towards business data processing. Most DBMS that do not have on-line system characteristics can be interfaced with TP monitors for access by terminals.
System evaluation Every time a new system is designed and implemented, oaths are taken to continue to review the usefulness of the system and modify it so that it stays current and useful. In practice, this never happens.	The ability to monitor a system and its data is far simpler under the DB approach since the DBMS can develop continuing statistics on both the data and the application.
Documentation Poor system documentation is prevalent since it is considered one of the least desirable tasks of programming. As a result, later maintenance on the system becomes more difficult because of inadequate documentation.	Documentation under the DB approach is simpler since documentation of the master file structure has been made simpler.
Large maintenance load The Systems Department always seems to be behind in implementing the many maintenance level changes that are requested by user departments.	Because program patches are easier to apply under the DB approach, maintenance of programs becomes easier and the backlog can be cleared away with fewer programmers.
Large program development costs Because implementing new applications often requires numerous programs and complex manipulations among many different files, the cost—especially labor costs—of implementing new systems are too high.	The DB approach with central maintenance of the master files simplifies and reduces the costs of implementing new applications after the DB has been initially created. This can be especially important in programs that are written for "one-shot" decision-making purposes.

second generation equipment or emulating with third generation equipment should study assiduously the concept of redesigning major applications for a DB environment. Merely rewriting for COBOL or another language compatible with third generation computers is obviously not enough.

Problems with the approach

As many companies have already discovered, implementing a DB is not a simple job. Numerous problems should be fully understood before commitments are made.

The first and probably the thorniest problem is that while DB is conceptually advanced, there is a concomitant lack of well-trained people to implement it. Time will overcome this lag. Nothing in the DB approach precludes its universal acceptance in the 1970s, just as COBOL was welcomed during the 1960s. Training is readily available from computer manufacturers, software vendors, and management firms specializing in executive and technical seminars. The passage of time should remedy this shortage of people; already, several hundred applications of DB systems have been implemented and some knowledgeable people are available.

A second DB problem is that it withdraws functions the programmer once performed and delegates them to the integrated hardware/DBMS system. By so doing, a need for computers with larger memories and greater processing power is created. To date, most DB users have been those who can afford powerful computer systems. This is likely to be the situation for a few years, until new computer capabilities can depress the overhead cost to a level feasible for smaller users.

Another problem related to DB is also an associated opportunity. Rarely are a company's existing systems merely converted onto a DB. The resourceful company will use the conversion as a complete redesign springboard for systems and will develop a more versatile product. For example, it will generate exception reports instead of grinding out massive stacks of paper for inquiry use. One requirement for a redesign is a total commitment of money and time; after all, we're talking about systems that may have had a ten-year genesis. They will not be replaced quickly or easily, even with tools such as DB. The fact, therefore, that DB implementation merges with redesign of major systems infers that it may involve more time and more money than other systems. For example, redesign and implementation of the complete back-office computer system for one of the New York Stock Exchange's largest member firms was accomplished during the period 1968 to 1972 at a cost of $3 million. This project used straight COBOL. DB was not considered because of the state of the art at that time, but the same project with DB would have been of the same magni-

tude. (For more information see Schussel and May, "Wall Street Automation: A Primer," *Datamation*, April 1970.)

Along with the possibility that the new system may not be just a simple conversion of the old but, in fact, be new and different, comes the concomitant problem of inability to use parallel runs. These computer runs allow old runs to be semiautomatically compared with new runs to reveal any errors in the new system. This means that the output from the new DB will have to be visually examined, a much more laborious process than is involved in many standard system conversions.

While DB technology permits more junior programmers to become productive sooner, it also necessitates greater expertise on the part of the systems programmers. These people, who maintain the computer's operating system, will also have to control the DBMS. They are typically among the most highly skilled and highly paid data processing professionals.

Malfunctions in computer hardware or software during the running of today's typical business applications call for reruns. Normally the program is rewound to a check point or to the beginning, the original master file and transaction file are loaded, and the run is regenerated. Even if a particular master file is destroyed, one can return to the "father" tape—the original master from which the current master was created—and rerun a couple of times to make the system current. Under the DB system, several programs may be updating and accessing the DB at the same time; therefore, the results of a malfunction in the hardware or software are not so easily cured. The primary technique for recovery in this situation involves continual journaling of transactions against the DB and the use of these lists to restore. Because of the large number of potential interactions, this procedure is more complex and requires more time than would be necessary with current procedures.

Also, the current technology of DBMS implementation has not yet evolved to a point where standardized approaches are used by all vendors. Some individual DBMSs are really unique in their approach and as a result will lock their users into one type of software/hardware, making it difficult—if not impossible—to ever convert the system to another vendor's computer. This represents a disadvantage in the sense of flexibility sacrificed, of course.

For many applications, the use of a DBMS may result in computer-run times that are substantially longer than those when more conventional file-management techniques are used. The systems manager will be faced with the decision of trading off the advantages of standardized control that the DBMS gives him for the disadvantage of having to pay more in the way of computer time for some individual runs. Although there are many DBMS applications where the run

times will be just as quick, if not quicker than in a standard approach, in some applications the run time will be so much slower that the DB approach is not a good decision. Therefore, the DBMS decision should only be made after a study of access techniques to the data is performed by qualified technical individuals.

In parallel with the above problem is the frequent situation that implementation of the first DB system results in more costs and time than a conventionally implemented comparable system. This happens because of two reasons. First of all, by moving into data base technology, most companies are moving into a new technical area with which the data processing personnel have not been familiarized. Any time a new technology is encountered, time is spent in education and debugging of approaches. Secondly, setting up the DB itself is analogous to a one-time capital investment. Often, the return on this investment is not realized with the first system alone, but is realized with the implementation of later systems which take advantage of the multiple uses of data.

One final problem is the necessity for management commitment, initially and throughout the project. In a complete evaluation of the company's systems, previously undiscovered errors will almost certainly come to light. Data processing and user management must be prepared to accept these as normal and must take necessary steps to eliminate such inefficiencies in the new system design. Management commitment is also necessary because unfamiliar definitions whose use is mandatory to interface with the DB may be generated.

Because of the reasons listed above, unless top management understands the objectives and is committed to the DB approach some of these perils may lead to the premature death of the project. This means that the EDP Department must mount an effective management orientation program before committing itself to this approach. approach.

Data base plan of action

After having studied a large number of successful DB implementations it has been possible to identify a number of general truths that if used will greatly increase the success probability in a DB implementation. These are summarized in the ten conclusions, or "Management Guidelines," below.

Plans. Since many data base implementation projects will be long and expensive, it is necessary to do good long-range project planning, which involves at a minimum the identification of multiple milestones and keeping Gantt chart progress against these identified milestones. Project control systems can be helpful on this point. As in any project, it is helpful to take a modular approach to implementation. In other

words, bust the problem up into as many pieces as possible and proceed toward implementation on a basis which involves control of individual modules.

Standards. One of the key characteristics of successful DB implementation has been the development and use of a comprehensive set of standards (names of data elements, programming language interfaces, implementation guidelines, etc.). With a community file used by a community of users, well-thought-out standards actively used by all parties are an essential element of successful implementation.

Technical evaluation. Early in the project, it is wise to do a thorough technical study of the size and structure of the DB and the proposed access methods the DBMS must use to effectively process against the DB. Different DBMSs have widely varying capabilities for accessing a DB and it is important to have a feeling for the proposed DB applications before making a commitment to any particular DBMS. An incorrect DBMS choice will result in heavy needless expenditures for additional computer hardware.

Corporate committee. A team of middle managers representing all major business functions should be formed sometime during or after the preliminary feasibility study has been conducted by the data processing department. This team has the final authority for defining data and should hear appeals regarding conflicts in approaches. All DB-related functions must be represented on this team.

Logical organization. The DB structure itself and the list of applications to be programmed against it should be organized functionally: marketing, production, engineering, and so forth. Peculiarities of an organization, such as a specific individual holding a certain job at only one point in time need not be stressed in the scheme.

New approach to systems. The corporate committee, in concert with the data processing department, should identify the decision-making processes in the business and make sure the system is designed accordingly. In other words, current systems and their requirements should not be accepted as the prototype for the new system. A more detailed investigation into information needs must form the basis for the DB system.

Dictionary development. Early in the project development the DB elements should be defined and put into a dictionary format. At this stage the relationships among the data elements can often be established. The analysis of this data should be structured to eliminate duplication and to develop standardization.

Data administrator control. A system for data control, managed by the data administrator, must be established and implemented before system activation.

Phased implementation. Finally, a detailed schedule should be developed for phased implementation of reports and information out of the DB system. While the overall implementation may cover a lengthy period, products should be identified which can be successfully delivered at early points—and, if for no more than public relations, they should be made available as soon as possible throughout the development cycle.

Small, batch, and noncritical. Don't forget the three commandments of all new systems' projects. Keep your first application relatively small, implement it in a batch mode before you make it on line, and make sure that your first application is noncritical, at least to the extent that you allow yourself time for slippage in the project development.

Appendix

Five popular DBMS products

IMS is IBM's top line DBMS product, operational on the larger 360 and 370 models. As befits this role, it is the most versatile of all currently available DBMS products. Entry into the data base can be through any of the commonly used access methods: sequential, indexed sequential, direct, and indexed direct. From the root segment entry point, access to the rest of the DB is through a hierarchy which may be up to 15 levels deep from the root. Each segment block at any level may have up to 256 different dependent segment types. IMS is available in both batch and teleprocessing versions and with various query languages. The system has a reputation of benchmarking very slow (execute speed) compared to almost all other DBMS. The cost in terms of required main memory overhead to run IMS is also higher than almost any other system. This main memory requirement prohibits effective use of IMS together with IBM's operating systems on a computer with less than 512K bytes of main memory for batch and 768K for on-line. The system also has a reputation for being difficult to implement and, accordingly, requiring major investments in programmer training and software support. It is available on a lease-only basis from IBM for approximately $600 per month for the batch version and $1700 per month for the TP version.

IDMS—Of all the major DBMS available for IBM 360 or 370 equipment, the only one which meets CODASYL standards is IDMS. IDMS was developed from 1969 through 1971 by the B. F. Goodrich Chemical Corporation. Subsequently, Goodrich contracted with Cullinane Corporation of Boston, Massachusetts to take over further development, generally enhance and market the system. IDMS is a full-fledged, general purpose DBMS with a reputation for broad

capability and cleanly implemented code. It represents the only choice for those IBM users who wish to acquire a network type CODASYL DBMS. Typical IDMS installations require main-memory overhead of 50–65K, and IDMS has been marketed successfully on small installations (IDMS, TOTAL and IBM's DOS/VS-DL1 are the primary viable choices for DOS users desiring a DBMS) as well as large. A particularly interesting feature of IDMS is the associated report generator, CULPRIT, that is marketed by Cullinane Corporation. With several hundred installations, CULPRIT is a well established and highly regarded product. For those users who are looking at a DBMS largely in light of its report generating capabilities, Cullinane's IDMS/CULPRIT combination offers a good choice of obtaining a high quality DBMS and RPG from the same vendor. The IDMS integrated data dictionary system is one of the best. License fee for IDMS is $37,500 and for the associated CULPRIT report generator $17,500.

MARK IV—Although a file management system and not a DBMS, as a DBMS is defined in this chapter, Mark IV is described here because it possesses some characteristics of a DBMS and with over 800 installations ($35,000 purchase price) it has been one of the most successful software products sold to date. The Mark IV system primarily accomplishes four functions: (1) edit and validation, (2) file maintenance, (3) data retrieval and selection, and (4) editing and report formatting. To accomplish these functions a user or programmer uses the Mark IV language, which is much higher level than COBOL. On systems with relatively low computational requirements, a typical expectation might be for the Mark IV implementation to require one fourth the programming, test, and debug time of COBOL. Every time a Mark IV program runs, it "compiles." The object code generation is normally very fast but total execution speed for a Mark IV application is typically slower than for a well-written COBOL equivalent. The system is available in several different software configurations for IBM 360/370 (also runs on Univac series 70) computers and has been frequently implemented on systems as small as 360/30's. A number of special features for Mark IV have evolved over time and caused it to be competitive to DBMS approaches in some applications.

SYSTEM 2000—This is one of the few DBMS's that offer a fully inverted file access method. It will construct tables on as many fields (keys) in a record as are desired, and retrieval is then achieved by Boolean search of criteria through the Tables (indexes) rather than in the much larger data base itself. As a result it is highly efficient for query-type searches. A hierarchical structure is generally applied to the data base. As opposed to most DBMSs which are written in computer assembly language, SYSTEM 2000 is partly a Fortran implementation and, as a result, its implementation on different computer systems has been simplified. It is operational on large CDC computers (64 bit word structure), large Univac 1100 series computers (36 bit word structure), and IBM 360 or 370 computers (8 bit byte structure). Requiring main memory residency in the range of 180K bytes (IBM) SYSTEM 2000 would typically be implemented on a computer with processing power equal to or greater than an IBM 370/145. SYSTEM 2000 has been modularized to a greater extent than most DBMSs and the purchaser may buy only those features of interest. The following are examples of modules and purchase prices. Basic SYSTEM 2000, $30,000; language interface, $10,000; RPG, $15,000; sequential processing, $15,000; TP, $25,000; multiple threading, $20,000; etc. As of mid-1975, SYSTEM 2000 was the third most popular (number of installations) DBMS in the U.S. on IBM equipment (TOTAL and IMS holding the first two places).

TOTAL—As of the beginning of 1975, TOTAL was the most popular (in terms of number installed) DBMS in use on IBM equipment. In addition to being operational on IBM 360 and 370 computers, various versions of TOTAL are operational on IBM System 3, Honeywell 2000, and Univac series 70 computers. In contrast with most other DBMSs, TOTAL requires only a small amount of main memory residency (30K to 50K including buffers) and as a result has been implemented on hardware as small as a 32K IBM 360/30. The structure of the system is very simple, essentially consisting of a basic direct access method into a "single entry file" followed by chaining links into the rest of the data base "variable entry data sets." The system has a good reputation among its users as being very simple to implement, requiring minimal software support. The prospective user, however, would be well-advised to carefully study TOTAL's access method in order to determine if this approach and its associated restrictions (especially for retrieval on secondary keys) represent an effective approach for the update and retrieval needs. Several TP packages and report program generators are available both from Cincom and others to interface directly with TOTAL data bases. Available under lease or purchase plans, the system sells in the general range of $30,000 for the batch version.

Bibliography

LYON, JOHN K. *An Introduction to Data Base Design.* New York: Wiley-Interscience. 81 pages.

General introduction to data base management for programmer types. Covers relevant concepts from mechanical storage devices to logical organization of data bases and their implementation. Trade-offs and efficiency of various systems are described, and examples are presented

which have used IBM's IMS/2, Honeywell's IDS, and Burroughs' Forte.

ACM. *DATABASE*. 1133 Avenue of the Americas, New York, N.Y., 10026.

This quarterly newsletter of the special interest group on data processing is primarily concerned with data base and data management systems. The newsletter is published in magazine format. One moderately good article within this newsletter follows.

TREANOR, RICHARD. "Data Management Fact and Fiction." *DATABASE*. Spring/Summer 1971.

PAN, GEORGE S. "The Characterization of Data Management Systems." *Data Management*. June 1971.

This article is a very short general-purpose treatise on DMS. Definitions, structures, file organizations, and overhead structures are covered in a six-page article. The material presented is too highly condensed to be of use to anyone other than the person who already knows the subject under discussion.

MCCARTHY, JOHN, JR. "Data Base of the 70s." *Data Management*. September 1970.

This is an excellent, short, theoretical view of data base, its definition and its approach to solving some of the significant problems in the data processing sphere.

GUIDE INTERNATIONAL. *Comparison of Data Base Management Systems Reports*. October 1971.

A position paper representing Share/GUIDE opinions on a proposed data base management system and IBM's response to those opinions, objecting to the structure of the proposed Data Manipulation Language.

"Creating the Corporate Data Base." *EDP Analyzer*, February 1970.

"Organizing the Corporate Data Base." *EDP Analyzer*, March 1970.

"Processing the Corporate Data Base." *EDP Analyzer*, April 1970.

"Data Security in the Corporate Data Base." *EDP Analyzer*, May 1970.

The four articles above are a series of publications in the *EDP Analyzer*, a monthly newsletter published by Richard Canning in Vista, California. The first one presents an overview and the introductory report on the concept of a common data base. Problems and promises are discussed. The March issue addresses the problem of file organization and data relationships. The April issue discusses data management systems, giving specific examples of several brand items. The May issue is addressed to the less exciting but very important problem of data security in the situation when sensitive data resides in on-line files. Many good references are listed.

"The Debate on Data Base Management." *EDP Analyzer*, March 1972.

Must reading for anyone who wants to understand the various positions of GUIDE, CODASYL, and the various hardware manufacturers on standards for data base management systems.

"The Data Administrator Function." *EDP Analyzer*, November 1972.

Another worthwhile article in this newsletter on a subject of interest to data base implementers.

BYRNES, CAROLYN, and STEIG, DONALD. "File Management Systems: A current Summary." *Datamation*. November 1969.

Although several years old and, therefore, somewhat outdated, this article gives a good summary overview of the file management systems generally available in 1969. General characteristics and specific capabilities of AEGIS, ASI-ST, CDMS, GIM, GIS-BASIC, INFORMS, and MARK IV are covered.

GALLEY, THOMAS. "An Approach to Data Base Design." *Journal of Systems Management*. February 1969.

A short, well-reasoned article describing data base and data management. Objectives, requirements, and approaches are discussed and effectively summarized.

Data Base Management Systems: A Critical and Comparative Analysis. May 1973. QED Information Sciences, 170 Worcester Street, Wellesley, Mass. 02181.

Starting with an introductory overview, this book gets into a detailed comprehensive discussion of features such as data manipulation, query languages, file structures, communications and DBMS installations. From these general discussions, the text proceeds into a comparative analysis of four IBM-oriented DBMSs; IMS, TOTAL, SYSTEM 2000, and ADABAS.

SCHUSSEL, GEORGE. "Data Base: A New Standard for Insurance EDP." *Best's Review*, October 1972.

SCHUSSEL, GEORGE. "Business EDP Moves to Data Bases." *Business Horizons*, December 1972.

The above two somewhat similar articles provide a good introductory overview of the data base concept. Especially good is the discussion of the advantages and disadvantages of the data base approach contrasted with more normal data processing.

GUIDE INTERNATIONAL. *The Data Base Administrator*. November 1972. (GUIDE publications are not available for sale. Contact a GUIDE member (most large IBM users) or contact an IBM sales office.)

This 70-page report is easily the most outstanding and comprehensive analysis of the duties of a data base administrator.

GUIDE INTERNATIONAL. *Basic Requirements for a Data Base Management System*. February 1973.

Eminently readable technical primer describing the minimum functions of a DBMS.

ACM. *Data Base Task Group Report to the CODASYL programming Language Committee*." April 1971. 1133 Avenue of the Americas, New York, N.Y., 10026.

Must reading for the serious data base implementer—this 269-page report should be skipped by those who are just interested in general familiarity with the subject or management overview. An approach is presented for general data base capabilities to interface with a number of host languages. The general proposal contains provision for a data description language and a data manipulation language.