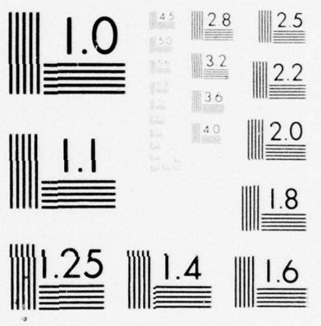
AD-A048 176 NAVAL INTELLIGENCE PROCESSING SYSTEM SUPPORT ACTIVITY--ETC F/G 5/2
EVALUATION AND SELECTION OF DATA BASE MANAGEMENT SYSTEMS,(U)
AUG 76 L E TOWNER

UNCLASSIFIED NIPSSA-51 NL

1 OF 1
ADA048176

END
DATE
FILMED
1-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

(6)

EVALUATION AND SELECTION OF DATA BASE MANAGEMENT SYSTEMS

(10) Larry E. Towner

(12) 53p.

(11) August, 1976

Naval Intelligence Processing System Support Act. ✓ New
(14) NIPSSA-51 ✓
2461 Eisenhower Ave.
Alexandria, Va. 22331

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

Data base management systems (DBMS) are being sold to data processing users in increasing numbers. Several reasons are being given for this popularity:

a. DBMS are easier to use than conventional data handling techniques,

b. DBMS solve file handling problems of complex data structures, and

c. The DBMS are utilized by the user's competitors--suggesting greater efficiency, lower cost, or other competitive factors.

Regardless of the reasons, and their validity, the fact remains that DBMS are enjoying great popularity in today's ADP marketplace. With that popularity comes an influx of packages which claim to provide vast capabilities to support data bases.

The increasing number of choices in DBMS packages presents the potential user with a problem of evaluating and selecting the system which fits the specific needs of the installation. (19:Preface) The complexity of the problem is increased by the relative inexperience of most users with DBMS, its properties, and its capabilities. (39,49)

This paper is aimed at the potential user of a DBMS. A collection of criteria which may be used to evaluate a DBMS is provided. Sufficient detail description is included to permit a "cookbook" approach, selecting those criteria which bear on the individual installation environment.

Chapter 2 provides a short background of the growth and direction of DBMS. It was included to help establish the picture of the DBMS environment and give the reader a feeling for the potential impact of this capability.

Chapter 3 asks questions intended to determine whether the installation of a DBMS is the practical solution to an installation's problems. There are just as many practical reasons for <u>not</u> installing a DBMS as there are reasons <u>for</u>. A DBMS is a very expensive tool, at least initially. The long-term payoffs must be significant to justify its use.(46)

Chapters 4 through 10 contain the detailed evaluation questions. With few exceptions each question is intended to stand by itself. When multiple related questions are presented, they are generally grouped together as a unit. When asking the questions of this type to vendors, it is recommended that they be separated so that they may be answered individually.

Chapter 2

BACKGROUND

Data management systems began appearing in the early 1960's. These early systems were known as File Management Systems (FMS). They were so-named because new approaches to processing of "files" of that time were developed. The Formatted File System, known as the FFS, was developed for the IBM 1401 computer and later updated to the IBM 1410. This system provided the user with the ability to process single files with a basic two-level tree structure.

These systems were subsequently converted to the IBM 360 and today are known as the Modular Data System (MODS) and the NIPS-FFS systems, respectively.(16:259,43)

The FMS techniques implemented by many vendors followed the same essential approach. They were, and still are, limited to concurrent access of only one or two files. The organizational structure is usually a two-level tree (hierarchical structure).(51)

Data management systems (DMS) expanded the capabilities of the FMS. By increasing the number of files which could be concurrently accessed, the DMS permitted more interaction between files within the installation. Some DMS packages increased the levels of tree structure to three and a few beyond that. IBM's Information Management System (IMS), as implemented on the 360 computer, is a good example of a DMS.(41)

Requirements for flexibility and consolidation of data files pushed system designers to develop the next, and current, level of implementation, the DBMS. The DBMS differs from the DMS in its scope of support.(44,45) The DMS is intended to permit concurrent access to multiple files structured in a similar way. Each file is independent of the others for some purposes and related for others. For example, a personnel file is related to a payroll file when viewing the complete information on an employee. Yet each may be processed independently of the other for personnel and payroll purposes.

The DBMS seeks to integrate all facets of the installation's data into one single data base.(17:5-6) The individual identification of separate files is lost as common data is used by all applications.(19:1-1) The payroll department may still have its own specialized data which is not available to other users of the data base but the common information (i.e., name, address, telephone number) is shared by the personnel and payroll groups.(1,2,3)

The DBMS therefore becomes the facility to implement an integrated data base covering all facets of the user's organization, eliminating separate files and the extra effort required to maintain them.(36:23,26)

The data processing industry has long been noted for its confusing terminology. Often different terms coined by individual vendors have the same meaning. The DBMS area is no different. New meaning for old terms have appeared along with a number of new terms. The glossary lists a few of these terms along with some common synonyms and a brief explanation.

The Conference on Data System Languages (CODASYL) was organized in the late 1950's to define a standard language for computer programming. COBOL was the result. CODASYL has continued to work on new standards, and the data base management area has received major emphasis.

In 1969 the Data Base Task Group (DBTG) of CODASYL released a DBMS standards recommendation to the data processing industry. This report has been upgraded several times, and work continues today on improving the specification. The 1971 revision of the 1969 report covers the Data Definition Language (DDL) and the COBOL-oriented Data Manipulation Language (DML).(17) A further revision of the DDL was published in 1973 by the Data Definition Language Committee (DDLC), successor to the DBTG.(18)

The most significant point to note about the CODASYL effort is its acceptance. Hardware and software vendors alike are developing new DBMS's using the CODASYL guidelines. Examples are Cullinane's IDMS, DEC DBMS10, Honeywell's IDS-2, and Univac's DMS1100. Industry leaders predict that, within a few years, enough major implementations of the CODASYL approach will be available to make cross-hardware transition of DBMS data bases relatively easy.(I1,I5,I6,I7)

The IBM user organizations SHARE and GUIDE developed a joint recommendation for data base development which was published in November 1970.(47) This document described the desired features to be found in the ideal DBMS. Unfortunately, and in contrast to the CODASYL specification, it did not provide the guidelines required to develop the DBMS. As a result, the SHARE/GUIDE document has received less attention than it deserves.

Chapter 3

DO YOU REALLY NEED A DBMS?

Numerous articles have been written expounding the merits of the DBMS.(9,32,37) Others cite reasons for procurement of a DBMS package.(41) Very few writers have stood "against the tide" to question the reasons for entering this new arena of ADP processing.(39,46,48,49)

There are many good reasons for adding a DBMS to the capabilities of an installation. Several will be discussed later in this chapter. However, the potential user must keep in mind that the reasons may not apply to every environment and the result of the evaluation study may conclude that installation of a DBMS is not the proper solution.(46) Each evaluation should be conducted with this possibility as one of the stated alternatives. The pre-conceived notion that a DBMS will be installed, with evaluation being used to determine which one, may result in a very costly error.(48)

As ADP use within organizations have grown, the complexity of the applications have grown too.(2) Many "independent" applications have evolved to satisfy the needs of individual users or departments. Where this has occurred, each user has provided their own data and received reports from their own exclusive files.

This unregulated and independent growth of applications has resulted in many related but disconnected files. As corporate need for data grows, the need to relate previously separate files becomes more urgent. Unfortunately, many files are structured in such a way that ready association is impractical. When attempts to modify the files to improve relationships are made, the complexity may overwhelm the programming staff. (12,13)

The DBMS is often advertised as the solution to this type of problem. Properly implemented, it may well solve the need to relate complex data. The capabilities of the various DBMS packages varies widely in this area. (16,19:2-13) The extent and complexity of the files to be related, present and future, will bear extensively on which DBMS, if any, is selected. (39,42,49)

The independent development of files, as described above, results in redundant data stored in many files. Costs associated with the entry, maintenance, and reference to redundant data may be significant. (19:1-1)

The integrity of the data base is compromised when redundant data, stored in separate files, is not consistently maintained. (4,5,15) When two departments each enter the same item into two files, the probability of error increases. The timeliness of entry is seldom consistent. One department enters the data before another, and the overall data base is no longer "in sync." If the item being updated is used as a key to relate files together, the delay will result in wrong answers if a query is entered before the second user updates his portion of the data base. (39,41)

Many of today's complaints about bad computer outputs can be traced to this problem. Only by limiting the entry of data to a single path into a single logical location in the data base can the integrity of the data be assured. (4,5) The DBMS is capable of supporting this requirement. The packages currently available vary widely in their ability to reduce or eliminate data redundancy. (27,28)

The volume of data to be stored in the data base affects the performance of most DBMS's. Depending on the method employed to relate data, system efficiency may degrade seriously when the data base approaches one million bytes in size.(38)

Inverted file systems typically update slowly and retrieve rapidly. The installation which processes a high volume of updates (10 per cent or more of the data base size per month) will find most inverted files systems excessively costly to operate.(15,22,I10,I11) This problem may force the user of a high-volume, large-size data base to either eliminate use of inverted systems or devise an independent updating method which will reduce update overhead.(19:1-14)

Chain pointer systems vary widely in their processing efficiency. As a general rule, the longer the data chain the slower the system operates. Very large data bases, therefore, process much slower than small ones.(46) Careful design of the data base structure can overcome some of this inefficiency. However, to adequately "tune" the structure it is necessary to know ahead of time the anticipated loading volume of each record type.(34,I1,I6)

The complexity of relating data within an organization was the downfall of the "total system" concept of the mid-1960's. The programming skills and resources to implement such a system were lacking in all but the largest installations. The inability of the industry to support "total systems" created much distrust and reluctance to embark on another such venture.(I7)

Today's Management Information System (MIS) developments are a subset of the "total system." The original plans have been scaled down to a level which can be developed without the need of specialized skills and hardware. The DBMS, with its improved data handling capabilities, may again open the door to the "total system." A few systems, principally those which adhere to the CODASYL specifications, have the ability to relate data in the manner it occurs in the "real world." (17,45,15)

The DBMS's relational ability is one of its most important assets. The structure of most businesses is hierarchical in nature. The data handled by the departments or segments of a business crosses over the boundaries of the organizational hierarchy. Efficient and cost-effective use of data demands that the DBMS be able to effect such a cross-over with a minimum of step-retracing and redundant data. Therefore a true networking capability is required for a DBMS to effectively support an application area as broad as the complete business. (44,13) If such an implementation is not available on the hardware used by an organization, it may be better to defer procurement until one is available.

On-line is a term which is commonly used in ADP today. Many organizations are developing direct input to their data bases and bypassing traditional keypunching of data. (20) The timeliness and accuracy of data is generally enhanced, and costs of data preparation are reduced. On-line data entry implies that on-line retrieval is also desired. The complexities of developing on-line data entry and its improved timeliness is seldom justified on the basis of costs and accuracy alone. Management desires faster response to its inquiries to meet the increased demands of the marketplace. (23)

On-line retrieval requires that those areas of the data base which may be requested from a terminal be available immediately. When the data is stored in several data bases, this requirement becomes difficult to meet. An integrated DBMS, using fully networked data structures, can permit access to any portion of the data base.(25) The entire organizational data structure is processable from any terminal through one set of software.(44) This feature dramatically reduces development costs and operating time.

Again, much care must be taken to determine if the candidate DBMS implementations can support the scope of the organization's data needs. If not, procurement of a "next best" substitute will be a waste of time and money.(46)

No organization should purchase a DBMS just because their competitors have bought one. The status symbol of owning a DBMS may actually improve the competitors position if their operation is suited to DBMS usage and yours is not. Competitive pressure applied by proper application of a DBMS justifies purchase only when the organization has determined that a DBMS will reduce the pressure by improved efficiency of operation.(13,14)

Chapter 4

TYPES OF DBMS AVAILABLE

One of the most confusing aspects of DBMS evaluation for the unfamiliar is the different types of DBMS implementations available. Each vendor claims his is the best, most cost effective, and flexible. For the purpose for which it was designed, most DBMS packages perform very well.

Data is structured by the packages using either data networking or a subset. Networking of data should not be confused with the networking of communications equipment supporting an organization. The communications network describes the relationship between points where users of an ADP system enter and receive data. The data network describes the relationship assumed between elements of data, records, and files within a data base.(40)

Data networks provide the most flexible and useful form of data relationships. Only through the network approach can data be freely related so that this data represents the actual relationships it occupies in "real world" conditions. DBMS implementations which permit unrestricted networks of data require more effort to design the data base structure but are easier to program and use.(42,44,15)

The hierarchical or "tree structure" method of data relationship is the most often used subset of networks. A majority of early implementations utilized this technique.(16) The hierarchical technique has the advantage of being easy to implement. It has the disadvantage of restricted relationship to records other than those directly above or below in the tree. This restriction makes access to related records in other trees or branches of the tree slow and difficult. Some packages have developed coupling techniques which use auxiliary records or tables to locate related records. While this accomplishes the end result, it is inefficient in data storage space and processing time. (19)

Each DBMS package utilizes some form of networking or hierarchical data structure. The method used to link the individual data records together within the network or hierarchy also differs considerably.

One of the most often heard but least understood methods of connecting related data together is the "inverted list." This technique stores data records in the data base (again by several different methods depending on vendor) and associates the data through lists of pointers stored elsewhere. It is possible to establish many lists to point to certain data values in records containing those values. In some cases the data itself is removed from the record and referenced to the list. This feature reduces the volume of data stored in the data base but increases the overhead of processing data. (15,22,II4)

The most important feature of the inverted list technique is its random, or ad hoc, retrieval speed. The use of many lists pointing through the data base makes the ad hoc query very fast. The same is not true of sequential retrieval. Sequentially produced reports are often slow. Update efficiency degrades on an accelerating curve as the volume of data stored in the data base increases.

Chaining is the other most common method of data linking in use today. This method requires the placing of a physical pointer in the data base record to identify the next record in the string of data. Often reverse pointers are used also. The pointers use data storage space and represent overhead. Where complex data networks are present, it is possible for pointers to occupy more space than the data.

Chain pointers are implemented in a variety of ways. Some systems rely completely on the pointers to relate data records. Others require that subsidiary records carry the physical key value which relates them to the owner. This approach is extremely wasteful of data storage space and creates an update bottleneck. The key field of the owner cannot be modified without modifying all subsidiary member records also. As a result, those packages which utilize this method of data relationship normally do not permit updating key fields. (19)

Logical pointer arrays are related to both chain pointers and to inverted lists. The member records are related to the owner occurrence through a secondary list. This list contains the data base addresses of all of the member records. Logical pointers eliminate the need for chain pointers imbedded in the physical records, saving data storage space. The pointer list is available in the same manner as an inverted list, improving the retrieval qualities of the data base. (33:216, 36:136-8)

It must be emphasized that all DBMS implementations have some overhead associated with relating occurrences of data together. The overhead may occur within the data base storage area itself, or it may occur in auxiliary storage areas. The "bottom line" of determining DBMS overhead is the total storage space required for all data sets necessary to operate the DBMS. Don't be fooled by vendor claims that one package is more efficient because no pointers are present in the main data base area. The overhead is still there. It is simply distributed differently. (33:136)

Chapter 5

EVALUATION OF DBMS VENDORS

Selection of a DBMS requires a careful investigation into the qualifications, background, and capabilities of the vendors of DBMS packages. Unless the user has the capacity to assume support of the software package at some future point, the stability of the vendor should be of prime importance.

Structural Background

When was the vendor's company organized?
Is the vendor incorporated?
Is the stock publicly held?
Is the stock listed on any stock exchange?
What is the vendor's Dun & Bradstreet rating?

The first group of questions are intended to identify the structural background of the vendor. It is important to know how long the vendor has been in business as an indicator of its stability and ability to survive over the long term.

An incorporated vendor is less apt to dissolve as a result of the death or departure of a principal than is a proprietorship or partnership. When the vendor's stock is publicly held and listed on a stock exchange, a further indication of possible stability is provided.

It may be wise to investigate the dividend policy in these cases to be sure that the company maintains a conservative financial profile. A high dividend rate may indicate that inadequate research and development of the DBMS line is being done. Finally, the Dun & Bradstreet rating of the vendor is a good indicator of the financial stability of the company.(39,I4)

Location

Where is the vendor's main office located?
Does the vendor maintain other offices within the United States?
What is the location of the closest office?
Does the vendor provide technical support personnel at all offices?
Is 24-hour technical support available at all offices? If not, is 24-hour technical support available?

The location of support facilities for the DBMS is very important. The ability of the vendor to train user personnel and to provide technical assistance is affected by his location. The distribution of sales and technical support bears heavily on the speed and cost of support. The make-up and qualifications of local office personnel are important when determining what level of support may be expected. The installation which operates around the clock must have 24-hour technical support available.(48,I8,I14)

Personnel

How many personnel worked for the vendor on January 1, 1975? On January 1, 1976?
How many of the vendor's personnel were dedicated to DBMS support on those dates?
What percentage of the DBMS support personnel is devoted to technical support?

People are the greatest asset of the software vendor. The ability of the vendor to produce, market, and support a quality DBMS product is dependent upon the make-up of its personnel. It is useful to know the number of people working for the vendor on the same date over a two- or three-year period. This reflects the intent of the vendor to support a growing number of customers. The number of personnel actually supporting the DBMS on the comparison dates provides an indication of the importance of the DBMS product within the vendor's company. (L2,I1,I2)

Diversification

Does the vendor market any other software product?
What percentage of the vendor's revenue comes from the DBMS?
How many installations of the DBMS were installed on January 1, 1975? On January 1, 1976?
Were any installations discontinued during the period?
How many different industries are using the DBMS?

Diversification increases the stability of a company. The software business is no exception. The vendor who relies entirely on a DBMS and its related packages for its economic survival is forced to adopt policies less stable than one who has other products to absorb some of the costs of operation. If other products represent less than 50 per cent of the vendor's income, a marginally successful DBMS product could affect corporate stability to the point where the DBMS product would be dropped.

The growth of the DBMS over a year's period provides some indication of the market penetration and popularity of the package. It may also be an indicator of the vendor's marketing capabilities. This marketing factor will often show up in a large percentage of non-technical personnel. The total number of installations is meaningful only if the number of installations which dropped the DBMS is also known. A high turnover in users often indicates a lack of continued customer satisfaction.

Finally, a users list provides an indication of the ability of the DBMS to support a broad application area. A users list oriented only to a single or to a few industries warns of a package which has limited application. (I5)

Chapter 6

EVALUATION OF DBMS ENVIRONMENT AND UTILITIES

The physical environment into which the DBMS will enter must be considered as part of the evaluation process. This environment includes the capabilities of the user to support the DBMS and the amount of supportive programming required to keep DBMS operations functioning smoothly.

Hardware Requirements

Is the DBMS implemented on more than one computer manufacturer's equipment? If so, which ones?
Is implementation of the DBMS consistent across all hardware lines?
Does installation of the DBMS require procurement of special devices, materials, or equipment?
What amount of main memory is required to effectively utilize all advertised features of the DBMS?
Can the DBMS be operated in less main memory space?
If so: What is the minimum main memory required to operate the DBMS? What degradations in operating effectiveness and feature support occurs with the reduced memory usage?
How many disk units (spindles, modules) are required to assure efficient operation of the DBMS?
Must all disk modules be on the line during operation of the DBMS?

Each DBMS package has different hardware requirements. Some are very thrifty in their use of core storage and disk space. Others require large amounts of both to operate effectively. The options open to the user increase when the DBMS is implemented on more than one line of computers. The user may select different hardware for varied uses and still operate the same DBMS. If this approach will be seriously considered, it is important to determine if all versions of the DBMS on different equipment are at the same level of implementation.

Several DBMS packages require that the user purchase supplementary equipment and/or software to assure effective operation. This condition places added burdens on the user in terms of cost and maintenance of the supplementary items.

DBMS vendors regularly understate the amount of main memory necessary to operate their package effectively. In each instance the package will operate with reduced main memory but usually with a coincident reduction in feature capability or operating efficiency. The disparity occurs when vendors advertise the minimum memory required to operate the DBMS without explaining the reduced performance caused by extensive use of overlays. Performance increases as the use of main memory increases, reducing the number of overlays employed.

The number of physical disk units required to operate the DBMS at its advertised speed is often larger than what is apparent in vendor's literature. Using fewer than the optimum number of units may seriously degrade performance. (An excellent example is found in the case of MRI's System 2000: Six data sets are required for each data base. If placed on one disk unit instead of six, a 40- to 50-percent throughput degradation is encountered. (I14)

Flexibility of operation improves if the user has the ability to switch disk units or commit only a portion of the disk space to the DBMS for a given execution.

<u>Auxiliary Support Programs (Utilities)</u>

Does the DBMS package include utilities to save and restore the data base?

Do save and restore utilities include loading density statistics on the areas being processed?
Can areas of the data base be selectively saved and restored?
Are backup and recovery utilities included as part of the vendor's package?
Can data base recovery utilities be operated from the operator's console?
Does the vendor supply utilities necessary to repair the data base in event of partial destruction?
Can utilities be used by operators and non-programming personnel with a high probability of accuracy?
Can DBMS utilities be executed while the system is running?
How much in-house programmer effort is required to write utility programs to support the DBMS?

All DBMS packages require some auxiliary support. None of the packages on the market today does everything internally. The number, versatility, and usability of supporting utility packages can either dramatically enhance the usefulness of the DBMS or detract seriously from its ability to support the user. Small installations must be particularly aware of the depth of utility support. Those packages which do not supply adequate utilities force their users to dedicate expensive senior-level programming support to the DBMS. The presence of extensive utilities does not in itself indicate a low level of support costs. The ease with which operators and non-programmers grasp and use the utility programs is important.

Data base save, restore, backup, and recovery programs must be flexible and easily used. The time when these programs are needed is seldom the most advantageous. Speed, especially in an on-line environment, may be essential to keep vital operations running.

Only a few DBMS utilities can operate on the data base while other DBMS functions are processing. This feature, again in an on-line operation, may be very important to assure proper security of the data bases without interruption of normal processing.

Finally, the amount of in-house effort required to write utility programs can have serious cost impacts on use of a DBMS. This factor should be included in cost-effectiveness portions of any DBMS evaluation.

Chapter 7

EVALUATION OF DBMS DATA BASE STRUCTURES

While the number of access methods supported by hardware vendors to handle data storage is limited, the methods which can be used to store data using these access methods is almost limitless. A majority of DBMS's implemented on the IBM 360/370 utilizes the Basic Direct Access Method (BDAM) as the storage vehicle, but the similarity ends there. Different intentions for the use of data dictate different storage methods. Evaluation of any DBMS must include the comparison of these storage methods to determine which one, if any, processes data in the manner desired by the prospective user.

Program Independence

Can the structure of the data base be modified without requiring the recompilation of programs using the structured area?
Can the structure of the data base be upgraded, adding new applications, without affecting the operation of existing programs?
Can data elements be added to or deleted from a record used by an application program without forcing recompilation of the program?
Can the length of a data element be modified without forcing the recompilation of a program using the modified data element?

Program independence is one of the most important advantages of a properly-designed DBMS. Program independence permits the structure of the data base to be modified and enhanced with minimum impact on all application programs using the data base. (19:3-5,36:35-36) No impact at all should be evident on programs whose records are not modified. Recompiling programs to adjust for data base structure changes becomes more and more impractical as the scope of the data base increases and the number of programs grows.

A few data base systems permit adding new data elements to an existing record definition without recompiling the programs using the record. Of course if the program must use the new data element, recompilation is often necessary.

Modification of data element size can have significant impact on using programs. Few DBMS implementations can keep such a change transparent to the using program's logic.(45,I1)

Hardware Independence

Is the logical structure of the data base independent of the storage device used?
Does the DBMS support storage of the data base on both disk and tape devices?
Is the physical device type transparent to the application program using the data base?
Can the size of data base storage areas be modified without altering the structure of the data base or the programs which use it?
Can the size or device location of the data base be altered without forcing the unloading and reloading of the data base?
Can the programs using the data base be restricted to certain physical units of the data base without affecting the logic of the application program?

The ability to define data base structures and populate them without regard for the type of storage media is a definite advantage. All DBMS implementations support disk processing. The direct access approach is consistently the mainstay of DBMS storage philosophy. Disk, however, is not the only storage device which supports direct access. Magnetic drums and staged devices, such as the new IBM 3850 mass storage unit, are all serviced by direct access techniques.

Few data base systems support magnetic tape processing. This unfortunate circumstance forces new DBMS users to alter their processing medium at the same time they alter their file philosophy. There are still numerous applications where disk-residence of low-usage data is not cost-effective. (44,18)

Change being the by-word of ADP, the ability to alter the size of data base storage areas or the devices upon which they reside is a very important feature of the generalized DBMS. (I1)

Security and flexibility of data base structuring is further enhanced by the ability to restrict certain data base areas to physical storage units. This assignment restriction must be transparent to the programs using the data. This feature is particularly useful where data is sensitive and needs to be physically removed and placed in a secure area without affecting the operation of the system as a whole. (36:184-85)

Data Sets

What is the minimum number of data sets which must be established to operate the DBMS?
What is the maximum number of data sets which are supported by the DBMS?
How many data sets are required for effective use of each logical data base?
Must data sets be assigned in any order, in any sequence, or to specific devices to achieve optimum performance?
Can multiple data sets be defined for one logical data base?
Can multiple logical data bases be defined within one data set?
What is the principal access method employed to access data sets?
Does the DBMS support multiple access methods?

The number of data sets required by a DBMS is not a critical selection criteria in a large installation. In a small installation with a limited number of disk units, system performance may be degraded seriously if sufficient disk units are not available. (I3)

The DBMS must permit a variable number of data sets to be assigned. This number can be expected to grow continuously over a period of time as more and more applications are integrated into the data base structure.

Flexibility in the assignment of data sets enhances the usefulness of the DBMS. If an application using sensitive data requires several disk packs because of optimum data set distribution, the willingness of the user to implement the DBMS is often reduced. (14)

Most DBMS implementations utilize the BDAM as their means of communicating with disk storage. A few have developed their own methods. Both approaches have merit and hazards. Using BDAM recognizes its stability and ease of utilization. It also subjects the DBMS to modifications which may be implemented by the vendor to the BDAM software. (12) Independent access methods may be more efficient but increase the complexity and maintenance costs of the DBMS. Many different access methods are available today. (31:150-54,33:195-96)

A very useful feature, supported by few DBMS implementations, is multiple access-method use. If this feature is available, the user may define the method which processes his data best.

Data Structures

Is the data base structure adequate to project data in the way it exists in the real world, or must data structures be adjusted to meet the restrictions of the DBMS?
Does the DBMS limit the size, quantity, mode, and organization of data elements in any way? If so, to what extent?
Does the DBMS support variable-length records?
What is the minimum and maximum lengths permitted for data elements and records within the data base?
Are structural relationships within the data base dependent on actual data values repeated between related records?
Does the system permit modifications of data elements which are used as key fields in relating and ordering data elements?
Does the DBMS structure approach permit effective reduction of redundant data elements between record types?
What form of structure is used to relate associated data records together (i.e., chain, inverted list, pointer arrays)?
Can indexing be provided both forward (next record) and backward (prior record)?
Does the DBMS permit compacting of data record occurrences to reduce storage usage?
If compacting is permitted, what is the processing overhead associated with the compacting process?

The relationship between data elements has grown more complex as the structure of business expands. A major problem in keeping ADP applications current with the needs of organizations has been the intricacies of data relationships. It is very important that a DBMS be able to interrelate data elements as they actually exist in the real world. When it is necessary to adjust or "bend" data structures to meet the restrictions of the DBMS, a great deal of flexibility is lost. Further, the increase in program complexity is measured in orders of magnitude. (15,17)

Most computers provide the ability to represent data in several forms or modes. The DBMS must also be able to process data in all modes available to the host computer. Restrictions on the size, quantity, or organization of the data frequently make conversion of existing data bases difficult, if not impossible.

Variable-length records are used extensively to improve storage efficiency and flexibility. This advantage should be offered by the DBMS vendor as a basic part of the package features. (19:3-7) Some vendors have provided an alternative approach to variable-length records which permits the variable portion to be defined as subsidiary records. While this approach will solve the data storage requirements in most cases, it is not adequate to meet the original needs of the user.

When the DBMS vendor places restrictions on the size and quantity of data elements and records within the data base, he places an added burden on the user. The user must be constantly conscious of this restriction when designing a data base. The user must also be careful to allow room for future growth of the records. (11) If his crystal ball is inaccurate, he may find himself having to completely restructure the original design of the data base after millions of records have been stored.

The accepted method of relating similar data records in sequential data files is through the data itself. Key fields containing live data determine the relationship and position of a record within the file. This arrangement has one major disadvantage. The user is unable to modify the contents of the key field without major structural modification of the data base contents. Unfortunately some DBMS implementations also rely on the contents of data fields to provide the relational linkage between records. Repeating the key field between members and owners is very wasteful of storage space and prevents modifying the key field. The impact of this technique becomes clearer when multiple set structures are considered. The records, which are owners or members of more than one set, can reach the point where modification of many data elements is prohibited because of their use as key fields.

It is important that the DBMS allow key fields to be modified as necessary. When a key field is modified, the record in which is resides must be relocated logically in the sets in which it participates. The redundant key field technique described above is very wasteful.

One of the major features of a good DBMS is effective and efficient utilization of data storage space. Data redundancy reduction or outright elimination is a mandatory feature when selecting a DBMS. (36:32)

Several structural approaches are used to associate data within the data base. The most popular is chaining; the second, inverted lists. Different update and retrieval capabilities are present for each method. (See Chapter 3.) (31:155-162,33:189)

Indexing of data normally utilizes lists or chains with forward pointers looking to the next record in the index. It is very useful to be able to optionally point to the previous record in the index (a trade off of improved performance versus reduced storage space). This permits scanning sorted lists in reverse order or backing up to the previous record without having to walk the chain or list all the way around to the previous record.

As part of the efficient management of storage space the DBMS must be able to compact data, removing unnecessary blanks and repetitive characters. The reduction in data storage, depending on the format of data, can often exceed 50 per cent. The cost of using data compaction techniques, however, must be considered. The number of records compacted may depend on how much processing time is required to perform the compaction (and its opposite, expansion) process. (36:433-34)

CODASYL Relationship

Does the DBMS conform to the data structures recommended by the CODASYL DBTG?
Does the syntax of the DBMS's DDL conform to CODASYL specifications?
Are any limitations placed on the extent of adherence to CODASYL specifications? If so, define in detail.
Are all eight CODASYL data relationships supported? If not, which ones are unsupported?
Does the DBMS support the CODASYL subschema technique?

The CODASYL Data Base Task Group (DBTG) specifications have become increasingly accepted by the data processing industry as the standard to be followed for DBMS development. The potential user needs to consider the impact of such standardization prior to purchasing a particular DBMS package. The packages which do not conform to the CODASYL specification will ultimately have to do so to survive. At the time the vendor changes to conform, users will face a painful change to syntax and procedures or be forced to continue using an increasingly obsolete version of the older package.

The Data Definition Language (DDL) defined by CODASYL is very similar to COBOL in syntax. It is very flexible and easy to use. The options defined in the DDL syntax cover such a range of data definition that most DBMS vendors have implemented only a small portion of those available. (17:65-148)

Few DBMS vendors have implemented more than half of the CODASYL specifications for DDL and Data Manipulation Languages (DML). It is important to evaluate the extent that DDL and DML features have been implemented to determine if the system can provide the tools necessary for the installation's environment. (44) Partial use of CODASYL combined with non-CODASYL approaches must be carefully reviewed to determine if future compatibility will be affected. (15)

CODASYL defines eight data structure relationships. Seven structures were defined in the original definitions. (33:192-206) The DBTG added the eighth in December 1975. The newest structure allows a record to be related to itself.

The subschema technique is a major feature of the CODASYL specification. The subschema operates as a filter between the application program and the data base, a "window" into the data base. As such, the subschema allows the application program to view a portion of the data base which it must access to perform its functions. The program, and its user, is not allowed access to the rest of the data base. Indeed, the user does not know any other data base exists.

Chapter 8

EVALUATION OF DBMS SECURITY, INTEGRITY, AND RECOVERY

The DBMS provides one of the best opportunities to integrate data and reduce processing costs that has been available to the ADP community since its inception. It also presents an unprecedented opportunity for compromise of that data. Separate files can be penetrated one by one. But without access to a complete set of files, the whole picture of the data cannot be obtained and compromise is seldom complete. The integrated data base removes these obstacles.

Internal Security

Does the DBMS implement passwords to control access to the data base?
At what level (i.e., data set, set, record, data element) can passwords be employed?
Can selected data within records be encrypted? If so, what is the processing overhead associated with encryption?
Does the DBMS implement passwords to control access to the data dictionary?
Can the data base be structured in such a way that certain data base record types are assigned to a physical device?

Passwords are the most common and easily implemented means of securing data. Unfortunately passwords by themselves are easily compromised. A scheme of multiple passwords and additional checks is necessary to achieve an acceptable protection level. The DBMS must possess some means of password protection of its data. The optimum arrangement is to permit password definition at all levels (i.e., data set, set, record, data element) at the user's option. (17:203) Password protection to the record level is fairly common among DBMS implementations. Data element protection involves a dramatic increase in processing overhead and data storage space. (19:3-2)

Another security method is encrypting the data within the data base. Encrypting generates significant processing overhead. Unless selected data elements can be encrypted instead of the whole record or file, inefficient processing will result.

Several DBMS implementations include a data dictionary as part of the package. This dictionary is the most sensitive file in the DBMS data base. Access to the data dictionary provides a penetrator with complete descriptions and structures of the entire data base. With such a road map, compromise of the data base becomes relatively easy. Extensive password protection is needed to secure the data dictionary. (17:78-79)

Each DML verb (store, modify, delete, etc.) should be individually protected to restrict application programs to their functions. (17:203) This feature is an integral part of the CODASYL subschema and can be readily implemented. A physical security constraint can be imposed if data can be organized so that all record occurrences of a certain type or sensitivity level are located on one device, such as a disk pack. This facilitates removal of the pack when access to the sensitive data is not required. (36:34) This approach is the least expensive but reduces flexibility and response time to data requests.

Data Set Security

Can data sets be restricted to certain users?
Can data sets be physically separated from the data base and normal operation of the DBMS continue?
Can the data sets be secured to prevent "dumps" of their contents?

Data sets are the operating system's reference to the data base, identifying where and how much space is set aside for data. It is essential that the DBMS provide maximum flexibility and security to data set selection and assignment. The DBMS must be able to selectively define certain data sets as belonging to specific users and containing only that user's data. (19:1-11,I1,I13)

It is a desirable feature of the DBMS that physical data sets (disk packs, etc.) be removable from the active data base without affecting the normal operation of the DBMS. (27) This facilitates physical separation and security of very sensitive data. (I9)

It is generally acknowledged that data base security halts at the machine room door. Operators and programmers can readily print the contents of data sets using standard utility programs. It is useful for the data base administrator (DBA) to have the ability to prevent such printing or to make the output meaningless (encrypting).

Functional Security

How does the system prevent one user from accessing data belonging to another user?
Can a user gain control over the DBMS system buffers through manipulation of his own data area addresses?
Is the subschema approach defined by CODASYL utilized fully?

An integrated data base contains data which "belongs" to many users. Typically each user is concerned that access to "his" data by other users is restricted. This restriction requirement may vary from minor to total. The DBMS must be capable of enforcing such variable restrictions and varying the level of restriction from user to user. (19:1-13) The manner in which the restriction capability is implemented must be carefully reviewed. Improperly handled, this feature can have serious impacts on program logic. (I5)

Most DBMS implementations utilize core buffers in which blocks (pages) of data are stored while being used by application programs. Each page commonly contains data from more than one user, and the combination of all buffers contains significant data. A skilled programmer can gain access to these buffers and read data beyond that authorized. DBMS implementations must guard against this invasion when permitting multiple users' data to coexist in the data base. (17,19)

The subschema approach defined by CODASYL provides a selective window into the data base which may be tailored to each individual application requirements. The subschema may be very sophisticated and provide a large measure of protection while enhancing the independence of the application program from the data base. (17:153-196)

Integrity

Does the system provide the means to validate raw data element occurrences prior to the storage of the data into the data base?
Can entry of data into the data base be automatically restricted to prevent conflicting data from being stored?
Does the DBMS permit concurrent update of individual data base records by multiple users?
How is "deadly embrace" avoided?
Can access to areas of the data base be restricted to control the level of multiple accesses to any one portion of the data base?
How is file lockout between contending users prevented by the system?

The CODASYL specification provides the facility to validate raw data before storing the data in the data base. (17:98-99) This feature has not been widely implemented but has the potential to increase the DBA's control over the data base, preventing insertion of erroneous data.

It is very desirable to be able to control the insertion of new data into the data base so that it is consistent with existing data already stored. This feature can be programmed by the using run-unit. This approach, however, is beyond the immediate control of the DBA. A better control can be achieved if the DBA has direct control over the insertion function.(17:94-97)

Concurrent update occurs when two independent users of the data base simultaneously attempt to update a specific record occurrence. This condition, while not common in actual practice, can destroy the integrity of a data base in a short time. A DBMS system must have the ability to overtly prevent such contention conditions from occurring. Deadly embrace, the fear of all systems people who deal in multiple users of a single data file, occurs when two users attach resources serially as they require them. Overlapping requests can render both users inactive without recovery. The method employed by DBMS's to avoid or recover from this condition is important to the ability of a DBMS to operate without significant operating problems.(20,27)

Contention and deadly embrace can be avoided by distributing the data over several areas of the data base. User access can be controlled more effectively, and the number of users accessing any one area is reduced.

File lockout occurs during periods of contention by multiple users. When one is updating, others must be prevented from also updating. It may, however, be desirable to permit some users to retrieve while others are updating, particularly in on-line environments. The DBMS must be able to selectively control lockout to assure maximum effective use of the data base while maintaining maximum integrity.

Recovery

Does the DBMS provide a method for taking regular checkpoints of its operation?
Are checkpoints automatic, initiated by operators, initiated by programs, or a combination of the above?
Do checkpoints include core images of the programs using the DBMS?
How rapidly can the DBMS be restarted from a checkpoint?

Does the DBMS provide facilities for physical backup of data bases on tape or disk?
Does the DBMS automatically recover from the failure of an application program using the data base when operating in batch and on-line modes? Must operator intervention be used to assure recovery?
Can the DBMS automatically recover from hardware failure associated with the devices and/or channels on which the data base resides?
When total hardware failure (CPU halt or power failure) occurs, can the DBMS be restarted "warm" or must a "cold" restart be performed?
When restarting "cold," how much time is required before normal operation may resume?
When software or hardware failure causes the DBMS to fail, what damage to the data base is likely?
Can the data base damage be recovered through use of standard vendor-supplied utilities?

Data base recovery is one of the critical aspects of any DBMS. Without the ability to recover following abnormal situations, a data base is useless.(I2)

Checkpoints are the classical method of recovering from malfunctions during long processing runs. When a DBMS operates in a multi-tasking environment, regular recovery points are necessary. These checkpoints should be frequent enough that recovery can be accomplished quickly with minimum disruption of service to users.(I1)

Checkpoints should be automatically taken to assure continuity. It is useful, but not essential, that operations personnel be able to initiate additional checkpoints.

Classical checkpoint techniques have included core images of the program executing. While useful to recovery of a batch program, this feature becomes an overhead factor in on-line multi-programming environments. If included, the core image feature should be optional.

Restart of the DBMS from a checkpoint must be accomplished rapidly. In an on-line environment, speed is essential and may be very critical to the users supported by the DBMS.

Associated with checkpoints is the use of a physical backup of the data base on tape or disk. These "saves" of the data base are typically used as the points where reconstruction of a damaged data base begins. It must be possible to save a data base at any point where its integrity is known to be intact, typically at a quiescent or idle point in processing.(I7)

When an application program updating the data base abnormally terminates, the data base is left partially changed. Left alone, this condition damages the integrity of the data base. It is important to remove the effects of the program by returning the data base to the condition prior to the program's start. Such a feature must be automatic when operating in an on-line or multi-tasking mode. Operator intervention should not be required except in a single-task batch mode.(I1)

Hardware failure is the weak spot of most DBMS implementations. Few systems can recover automatically from failures associated with the devices and/or channels which serve the data base. It is essential that a smooth recovery be allowed by the system without reloading the data base if it has not been damaged.

When total hardware failure occurs, execution of the DBMS halts at an abnormal location if it was executing at the time of the failure. This halt may prevent the DBMS from being restarted upon recovery of the hardware. It is important to on-line users that the system begin operation as soon as possible. A "warm" restart permits the system to continue where it halted with little or no impact on its operation. A "cold" restart often requires a complete reloading of the data base and restarting of operations from a previous checkpoint. The time difference between warm and cold starts can be significant, and a warm restart capability is very desirable.

Data base damage is common when the DBMS terminates abruptly due to hardware or software failure. The extent of the damage ranges from minor to extensive destruction in the systems currently marketed. It is very important to the user, particularly in an environment which experiences frequent disruptions, to determine what external impacts have upon DBMS performance.

Part of the price a user pays for a DBMS package goes toward the utilities needed to provide auxiliary support. Data base recovery utilities are essential to operation of the DBMS. Without them the system cannot be maintained, and the user is forced to write his own recovery capability. Few users have the extra resources available to write such programs. Depending on the complexity of the DBMS file structure, the task may be beyond the users abilities. Standard user-supplied recovery utilities, therefore, are a mandatory feature of the DBMS package.

Chapter 9

EVALUATION OF DBMS PROGRAMMER INTERFACES

A DBMS is a powerful tool to increase the scope of ADP processing. It is useless, however, unless it can be properly used. The ease with which programmers, analysts, and end users can utilize the feature of the DBMS will bear heavily in its evaluation.

Languages Supported

What programming languages may be used in conjunction with the DBMS?
How are the interfaces effected?
When a language preprocessor is used, does the preprocessor perform syntax and logical validation of the Data Manipulation (DML) statements used?
Does the COBOL language interface comply with CODASYL specifications?

While most data base system publicity is aimed at the business user, scientific and statistical data is often part of an integrated data base. It is important that all users and potential users of the DBMS have reasonable access to the data base through the language which best supports their application.(6)

Interface with the DBMS may be accomplished through several approaches. Nearly all of the systems implemented currently provide the ability to access the DBMS software through the CALL statement in those languages which implement calls. Several implementations also provide direct DML statement usage which accesses the DBMS either directly or by converting the DML verbs to call statements.(31:308-12,169-70)

The use of preprocessor to convert DML commands to call statements is becoming more common. The services provided by these preprocessors ranges from a straight one-for-one conversion of DML commands to calls to extensive syntactical and logical validation of the source program code. The greater the level of program verification performed by the preprocessor, the more effective and accurate the program code generated.(7,8)

The CODASYL DML structure has been proposed as an addition to the American National Standards Institute (ANSI) COBOL language upgrade. It is likely that the ANSI standards committee will approve the proposed DML verbs or a very similar structure.(17:201-69) As discussed earlier, the adoption of the CODASYL standard can have positive effect on those users of CODASYL-compatible systems and a negative effect on those whose DBMS does not conform.

Programmer Interface

What level of programmer experience is required to effectively use the DBMS?
Does the DBMS perform support functions which relieve the programmer from actions which are repetitive and/or error-prone?
Does the system permit definition of data elements using symbolic names? If so, what limits are placed on the structure of the names?

Few ADP installations have the luxury of a totally-experienced staff. There are always a few junior-level personnel on the staff. It is important that all programmers on the staff be able to effectively use the DBMS.(27) Without that ability, the use of DBMS will be limited to the few senior programmers whose skills can master the DBMS.(19:3-6)

Several DBMS systems provide support features which relieve the programmer of many tedious and error-prone tasks, such as data record formatting, call-statement generation, and communications area coding.

The greater the support provided by the DBMS, the less effort necessary by the programmer. This effort reduction increases productivity through less program coding and fewer coding errors. Significant coding convention and data name standardization features further enhance the programmer interface on some systems.

Meaningful symbolic representation of the data elements within the data base improves the readability and understanding of the data base structure. The number of characters permitted in symbolic names varies widely among available implementations, ranging from a low of 2 characters to a high of 16. If installation conventions require descriptive data names, less than 12 characters will present severe constraints upon naming within the data base structure.

Training

Does the vendor provide adequate training for all levels of programming staffs?
Is training a reinforcement of manuals and guide, or is formal training necessary to effectively use the system?
Is training available on video tape?
Does the vendor limit the number of participants in training classes?
How much training is included in the contract price?
Is training available locally?

Training of programming staffs is a continuous expense to all ADP installations.(37) The addition of a DBMS package to the software library requires that most, if not all, of the programmers and analysts on the staff become familiar with the capabilities, technical features, and use techniques of the DBMS. The type and depth of training differs from level to level within the staff. Application programmers, junior through senior, must know how to write interacting programs which will use the DBMS file features. Systems programmers must understand the interaction of the package with the operating system and other software packages. The systems analyst needs to know how to apply the capabilities of the DBMS to the application needs of the installation's users.(43)

Formal classroom training is often used to expand the material provided in the manuals supplied by the vendor. While this technique assures a rounded education on the system, it lacks the ability to refer to the manual for information once the instructor has gone. It is important that vendors documentation be as complete as possible and permit each level of user to effectively utilize the system without consulting continuously with the vendor.

Repeat training is a costly and disruptive necessity. As new staff members are hired, it is necessary to train them in the various software packages being used in the installation. The DBMS, when integrated into the installation's operation, must be understood by all staff members. Training courses available on video tape reduce the cost and improve the timeliness of instruction. While not as personal as live instruction, video courses may be run at will and can be repeated as often as desired. Programmers previously trained may replay certain tapes to refresh or reinforce their knowledge.

Vendors often limit the number of persons who may attend a specific training course. This is logical when the course is offered to the general public or when close individual support is required. However when a course is offered in-house, the vendor should be flexible to the needs of the user. The costs of training can be reduced significantly if a larger number of students are permitted in classes which are primarily lecture.

Most DBMS packages include some training in the contract price of the system. The amount of training varies widely. The quantity of training provided is seldom adequate to fully prepare the installation for DBMS use. Supplemental costs, with a few systems, may approach the license price of the DBMS itself.

The availability of local training is a very important feature. The cost of out-of-town training extends beyond dollars alone. The time required for travel and the loss of the trainee services during the course period can have severe impacts on installation effectiveness. Few installations can afford the disruption caused by extensive use of remote training.

Chapter 10

SUMMARY

Data base management systems is one of the most exciting developments on the computer scene today. It holds the potential, properly guided and controlled, of expanding the scope of ADP applications far beyond its current boundaries.

The pitfalls facing the potential user of DBMS are enormous and costly. It is essential that careful evaluation and comparison of available packages be made. As part of this evaluation, a careful meshing of DBMS features and installation requirements is necessary to assure effective use of a DBMS.

It is possible that the results of an evaluation will indicate that the use of a DBMS is not practical. When this occurs, procurement of a DBMS will be on shakey ground and potential users must proceed at some risk.

This paper has provided many questions for potential DBMS users to ask of vendors. Often the answers will generate more questions. When fully explored, these answers will provide the basis for a decision to buy or not to buy.

A final comment: "Let the buyer beware" is particularly applicable to buying a DBMS. The wrong decision will have lasting long-term effects on an organization and its ADP program.

GLOSSARY

This glossary includes terms common to the DBMS technology. Only a few of the most dominant or confusing have been included from the many used when discussing data base systems. A majority of the definitions has been developed using references 17, 19, 33, and 36.

Area (realm): A named sub-division of the addressable storage space within the data base to which records may be assigned independent of set membership. A logical piece of the data base.

CALC: One of the three record storage modes, based on the computation of a data base storage location using values supplied by data within the record. Used for direct-access records.

Chain pointers: A chain of pointers which can be followed from record to record and provide for sequential access to all records in the set occurrence. A storage mode for sets where data is stored with linked organization for serial access.

Data-aggregate: A named collection of data-items within a record and referred to as a whole.

Data base: All physical record occurrences, set occurrences, and areas defined by a specific schema.

Data-item: The smallest unit of named data within a data base. An occurrence of a data-item is a representation of a value and may consist of any number of bits or bytes.

Data set: A named collection of physical records, including the data used for locating the records (indices).

DDL: Data Definition Language. The language used for describing a data base or that part of a data base known to a program. Defined in terms of names and characteristics of data-items, data-aggregates, records. areas, and sets, and the relationships existing between occurrences of those elements in the data base. A language for the logical description of the data base.

Direct: One of the three record storage modes where a unique identifier, supplied by the using program, identifies the location in the data base of a record occurrence.

DMCL: Device-Media Control Language. The language used to describe the relationship between the logical schema and the physical storage space used to store data base records.

DML: Data Manipulation Language. The language used by the programmer to communicate with the data base system.

First: One of five set orders in which the new member record is inserted as the immediate successor to the owner record occurrence.

Integrity: Safeguards against occasional failures and accidents which can occur during processing of a data base. The safeguarding of data from undesired interaction of programs against the data base. The checking of the value of data to be stored in the data base to assure its consistence with data already present in the data base.

Last: A set order where the new member record is inserted as the immediate predecessor to the owner record occurrence.

Mandatory: A set membership condition which indicates that once the membership of a record occurrence in a set is established, the membership is permanent.

Member (child): A record within a set subsidiary to and dependent on an owner record.

Network, data base: The most general form of data structure. In a network any given element may be related to any other. A data structure in which an n-to-n relationship is permitted between elements.

Next: A set order where the new member record is inserted after another record occurrence which was the latest record within the set to be accessed.

Owner (parent): A record whose existence establishes the existence of a set occurrence. The elementary record of a set.

Pointer array: Sets organized through a list of member record occurrences stored with the owner record.

Prior: A set order where the new member record is inserted before another record occurrence which was the latest record within the set to be accessed.

Privacy: Protection against unauthorized access of the data. Refers to the rights of individuals and organizations to determine for themselves when, how, and to what extent information is to be transmitted to others.

Schema: Consists of data definition (DDL) entries and is a complete description of a data base. It includes the names and descriptions of all of the areas, set occurrences, record occurrences, and associated data items as they exist in the data base.

Security: Protection of data against accidental or intentional disclosure to unauthorized persons, or unauthorized modification or destruction of a data base.

Segment: Data-aggregate and/or record which contains one or more data-items and is the basic unit of data which passes to and from the application programs under control of DBMS software.

Set: A named collection of record types. As such, a set establishes the characteristics of an unlimited number of occurrences of the set. The basic structure of the CODASYL language specification.

Subschema: Consists of data definition (DDL) entries. It need not describe the entire data base but only those areas, sets, and records which are to be known to a specific program or programs. An application programmer's view of the data base.

Via: One of three storage modes based on the location of the owner occurrence of the set.

REFERENCES CITED

A. PRIMARY SOURCES

1. Auerbach Publishers, Inc., Data Base Management - Concepts for Management (Philadelphia: Auerbach, 1974), #3-06-01.

2. _______. Planning for Data Base Environment, #3-06-02, 1974.

3. _______. Implementation of a Data Base Environment, #3-06-03, 1974.

4. _______. Data Base Administration, Part 1, #3-06-04, 1975.

5. _______. Data Base Administration, Part 2, #3-06-05, 1975.

6. _______. COBOL Data Base Facilities, Part 1: Introduction, #3-07-01, 1974.

7. _______. COBOL Data Base Facilities, Part 2: Application Programming, #3-07-02, 1974.

8. _______. COBOL Data Base Facilities, Part 3, #3-07-03, 1974.

9. Blanchard, J. Stevens, "We Bet Our Company on Data Base Management," Datamation, September, 1974, pp. 61-65.

10. Braun, Roy, "Ingredients of a Data Base," Infosystems, December, 1973, pp. 32-34.

11. Cagan, Carl, Data Management Systems (Los Angeles: Melville, 1973).

12. Canning, R. G., "The Cautious Path to Data Base," EDP Analyzer, June, 1973.

13. _______. "Problems in Data Management," EDP Analyzer, March, 1974.

14. _______. "The Current Trends in Data Management," EDP Analyzer, February, 1974.

15. Cardenas, R. F., "Analysis and Performance of Inverted Data Base Structures," Comm ACM, May, 1975, pp. 253-263.

16. CODASYL DBTG, Technical Report of the CODASYL Systems Committee on a Survey of Generalized Data Base Management Systems, ACM, May, 1969.

17. _______. CODASYL Data Base Task Group Report, ACM, April, 1971.

18. CODASYL DDLC, CODASYL Data Description Language, Journal of Development, NBS Handbook 113, June, 1973.

19. Cohen, Leo J., Data Base Management Systems: A Critical and Comparative Analysis, PDC, 1974.

20. Cuozzo, D. E. and Kurtz, J. F., "Building a Base for Data Base: A Management Perspective," Datamation, October, 1973, pp. 71-75.

21. David, H. M., "Computers, Privacy, and Security," Computer Decisions, May, 1974, pp. 46-48.

22. Earley, J., "Toward an Understanding of Data Structures," Comm ACM, October, 1971, pp. 617-627.

23. Evans, R. W., "Data Base Organization for On-Line Systems," EDP In-Depth Reports, September, 1971.

24. ________. "DBMS," EDP In-Depth Reports, October, 1971.

25. ________. "Data Communications for Data Base," EDP In-Depth Reports, January, 1972.

26. ________. "Current Developments in Data Base Management," EDP In-Depth Reports, January, 1973.

27. ________. "New Software for Data Base Management, Part 1," EDP In-Depth Reports, October, 1973.

28. ________. "New Software for Data Base Management, Part 2," EDP In-Depth Reports, October, 1973.

29. Fife, D. W., and others, A Technical Index of Interactive Information Systems, NBS Technical Note 819, March, 1974.

30. Guide International, The Data Base Administrator, DBA Project, (New York: Guide, November 3, 1972).

31. House, William C., ed. Data Base Management (New York: Petrocelli, 1974).

32. Huhn, Gerald E., "The Data Base in a Critical On-Line Business Environment," Datamation, September, 1974, pp. 52-56.

33. Katzan, Harry, Jr. Computer Data Management & Data Base Technology (New York: Van Nostrand/Reinhold, 1975).

34. Leavitt, D., "Generalization Heightens Problems of Tuning DBMS," Computerworld, March 20, 1974, Vol. 8, #47.

35. Lyon, John K., An Introduction to Data Base Design (New York: Wiley, 1971).

36. Martin, James, Computer Data Base Organization (Englewood Cliffs, N. J.: Prentice-Hall, 1975).

37. Nolan, Richard L., "Computer Data Bases: The Future is Now," Harvard Business Review, Sept.-Oct., 1973, pp. 98-114.

38. Patterson, Albert C., "Data Base Hazards," Datamation, July, 1972, pp. 48-50.

39. Prendergast, S. Lawrence, "Selecting A Data Management System," Computer Decisions, August, 1972, pp. 12-15.

40. Reside, Kenneth D. and Seiter, Theodore, "The Evolution of an Integrated Data Base," Datamation, September, 1974, pp. 57-60.

41. Romberg, Bernhard W., "Data Bases: There Really is a Better Way to Manage Your Files," Infosystems, May, 1973, pp. 56-58.

42. Ross, Ronald G., "Evaluating Data Base Management Systems," Journal of Systems Management, January, 1976, pp. 30-35.

43. Rothnie, James and Fersch, Robert T., Data Management Systems, unpublished paper for the Navy Department, 1972.

44. Schubert, Richard F., "Basic Concepts in Data Base Management Systems," Datamation, July, 1972, pp. 42-47.

45. ________. "Directions in Data Base Management Technology," Datamation, September, 1974, pp. 49-51.

46. Schussel, George, "When Not to Use a Data Base," Datamation, November, 1975, p. 82.

47. SHARE-GUIDE Data Base Requirements Group, Data Base Management System Requirements, Share, Inc., November 11, 1970.

48. Taylor, Alan, "Ignoring Problems Common in Data Base Selection," Computerworld, November 19, 1975, p. 13.

49. Testa, Charles J. and Laube, Sheldon J., "How Do You Choose a Data Base Management System? Carefully!" Infosystems, January, 1975, pp. 36-39.

50. Urbach, Harold, The Role of Data Base Administrator (New York: Guide, International, November 1, 1972).

51. Welke, Larry, "A Review of File Management Systems," Datamation, October, 1972, pp. 52-54.

REFERENCES CITED

B. INTERVIEWS

I1. B. F. Goodrich Corporation
Corporate Headquarters
Mr. Jim Gilliam
Corporate Systems

I2. B. F. Goodrich Corporation
Chemical Division
Mr. John Menyes
Programming Manager

I3. Gould, Incorporated
Trenton, New Jersey
Mr. Clifford Johnson
Director of Data Processing

I4. Mutual Benefit Life Insurance Company
Neward, New Jersey
Mr. Chuck McCaig
Project Manager

I5. RCA, Incorporated
Princeton, New Jersey
Mr. Tom Stiller
Director of Corporate Systems

I6. The Royal Bank of Canada
Montreal Quebec, Canada
Mr. Martin Frobisher
Corporate Data Base Administrator

I7. Southern Railway System
Atlanta, Georgia
Dr. Bill Linn
System Director

I8. U. S. Government
Department of Labor
Bureau of Labor Statistics
Washington, D. C.
Dr. Lester Sachs

I9. U. S. Government
Department of Agriculture
Washington, D. C.
Ms. Roxanne Williams
Head, Data Base Systems Division

I10. U. S. Government
Department of Agriculture
Washington, D. C.
Mr. Paul Holm
Data base system user

I11. U. S. Government
Department of Agriculture
Washington, D. C.
Mr. Elden Hildebrandt
Data base system user

I12. U. S. Government
Department of Agriculture
Washington, D. C.
Mr. Ken King
Office of Operations

I13. U. S. Government
Department of Agriculture
Washington, D. C.
Mr. Mike Rose
Data Base Systems Division

I14. U. S. Government
Department of Commerce
Washington, D. C.
Mr. Tom Collins
Systems Manager