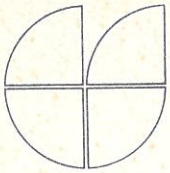


THE "WHY" AND CRITERIA OF DATA BASE MANAGEMENT SYSTEMS

To understand Data Base Management in today's technology, we must examine the evolution of information processing leading to the requirement for Data Base Management. This is best accomplished via an analogy.

Turn back the clock 12-15 years to a company with no data processing department. More specifically to the Order Services Department of this mythical company. Clerks processed and typed customer orders utilizing ledger cards for customer and inventory files. Copies of the typed orders were stored behind each inventory item. Ledger records were manually posted. Any analytical reports were extracted by hand which limited their frequency, computational complexity, and response time.

One day, a proponent of data processing proposed to the order services manager that he should automate his information processing. He was offered the following advantages:

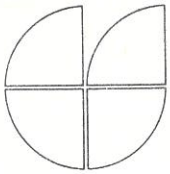
1. Clerical Savings.
2. Better Customer Services.
3. More accurate record keeping.
4. Better analytical information.

After lengthy persuasion, he agreed to automate. The data processing department sent its system experts to design the system. They did not fully understand order entry. The order entry people did not understand data processing. The design of the system was accomplished primarily by the systems analyst asking the order services people what they wanted in reports. The system was designed, programmed, manual files converted to mechanized files and implemented after much more expenditure of effort and passage of time than was estimated.

After initial implementation, the order service people discovered that they has lost access to data which they had in the past. They also discovered that they had neglected telling the system designer all of the information they needed.

"We must have a daily order received report by Customer." Data processing probably did not ask why; they simply programmed the report causing some reprogramming of original programs, the creation of another file and some increases in processing loads. The order services people did not prepare such a report under the old manual system. Why did they now require this report? Because they used to be able to go to a customer ledger card file folder and look at all of the copies of his order filed there when they needed that information in response to a customer question.

"We must have a daily order received report by product?" Again, why? Because under the manual system they could get to the ledger cards for an item and look at the copies of the unshipped order filed there.



This was the beginning of the report syndrome. More and more report requests being processed by the data processing department to serve the growing and changing information needs of the business and the ones overlooked in initial systems design.

As time passed, this same process was being followed as each department within an organization began automating its information processing. Each department's files were mechanized. For example, each department had their own Customer File: one for order entry, accounts receivable, sales department and so on. The load on the data processing equipment increased beyond original expectation. The number of maintenance programmers increased. The cost of data processing soared. Equipment acquisition did not come rapidly enough, thus service to the user departments degraded.

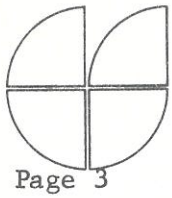
"We have to do something", shouted the Data Processing people. "We have to collect transactions faster and closer to their source. . . ." Thus the birth of a source data collection. "We have to solve this reporting problem. . ." Thus report writers. "We have to reduce reports. . ." Thus online inquiry.

When we went to the massive tape libraries to attack the problem of placing on discs data for online inquiry, we were astonished at the problem that batch serial processing hid from us - the problem of massive duplication of data. We could not afford the direct access space to store this volume of redundant information.

We forged ahead to try to consolidate files. This forced us to resolve inter-department differences over common data and much reprogramming of existing systems. We realized that information crossed departmental boundaries and so did sound system solutions to complex business problems. There is only one customer as a unique enity, not a different one for order entry, accounts receivable or sales.

Bringing data on-line caused the development of new and more complex methods of data organization. We could not respond to a demand inquiry by serially searching a file. We saw that our batch system handled the same transaction many times against many different files just to keep the duplicate data fields current and the separate application files "in phase". We could not afford this extra volume of transaction processing in real time. We saw that we had to develop interrelationships between files to provide demand-type and analysis-type report requirements and to allow us to develop more complex decision-making algorithms. . . .to make our data processing more planning and control-oriented and not simply historical reporting oriented as it was in the past.

We also discovered that the information of an organization is its most important asset. Information is separate and independent of departmental or application boundaries. It is also independent of the method of input



handling or ways to report information. The information of an organization should be managed by an independent function utilizing an independent application system. Before we go further, let us attempt to make a broad definition of what a "data base" is.

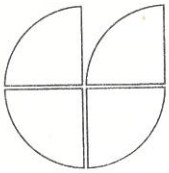
A "data base" is a pool or repository of all data required by an organization. Ideally, each piece of data is stored only once, eliminating the data duplication inherent in traditional methods. All data required by one application is immediately available to it, and several applications may share common information. As a result, data modified by one application is immediately available in its most current form to other applications. Transfer of data from one file to another is no longer necessary. Further, data association from a collection of inter-related or integrated data files is accomplished efficiently yet with file-by-file independency.

The concept of a data base is a simple one, but the problems and requirements surrounding the data base management system (DBMS) which is the essential implementation tool are extremely complex and not readily apparent. A properly designed DBMS should enable the user to rapidly and successfully begin to implement a data base approach which serves as the foundation for integrated management information systems.

However, eagerness to escape the problems of traditional processing and the obvious potential of a data base environment may lead many to commit to approaches which will further complicate rather than relieve their problems.

Therefore, industry experience indicates this resultant Data Base Management System (DBMS) should have the following design criteria in order to avoid in the future those multi-blind alleys of the past evolution of data processing which we have just discussed.

- Provisions for all data storage and processing requirements.
- Data independence at the data element or field level.
- Stringent control of data redundancy.
- Unlimited and flexible data association and relationship capability.
- Data base Integrity and Security protection.
- Environment Independence.
- Language and Equipment Independence.
- Optimum performance and efficiency in all environments.
- Ease of Conversion or Transition.



Each criterion is of critical significance and each are interrelated and interdependent. Each capability is necessary and the absence of only one may adversely and seriously effect the value or worth of others. For example, without comprehensive and virtually fail safe data base integrity, every other feature is of only marginal value.

THE DATA BASE MANAGEMENT SYSTEM MUST PROVIDE FOR ALL DATA STORAGE AND PROCESSING REQUIREMENTS

In the demand transaction oriented environment of terminal based systems there is no faster data retrieval technique than a direct or random approach. Since performance is so critical, this technique should be used whenever possible. However, in a data base environment, a file randomly organized may frequently be used to also satisfy serial batch oriented systems as well. Here a random file is a poor performer when compared to perhaps sequential processing. What the user gains in one area he loses perhaps many times over in another. This cannot be tolerated. Historically indexed approaches have been used as a compromises. Unfortunately, these compromises only insure the user that he will always get less than the best performance in both environments.

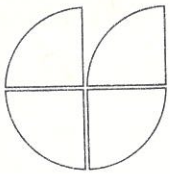
Ideally then, techniques should be inherently available within the DBMS to satisfy the high performance requests of demand or random processing in an optimal fashion but also be able to handle batch updating at speeds as fast or faster than sequential processing--without sorting the file.

Also, the systems designer should not be burdened with developing random algorithms, synonym handling, or disk space allocation or management. Periodic re-organization must absolutely be avoided. Therefore, the files should be self-optimizing over time after continual deletions and additions of records.

THE SYSTEM MUST BE DATA INDEPENDENT AT THE DATA ELEMENT LEVEL

The data storage and data presentation must be independent or transparent down to the data element (data field) level. Only with independence or transparency at this the data element level is it possible for users to be responsive to the dynamics of a changing business environment. An approach which provides for continuous progress in an evolutionary manner is essential. As new data elements are required, it must be possible to add them to both existing data records and data structures with no adverse impact on existing operational programs or systems. This should not require rework, redesign, or even recompiling of existing programs.

As we mentioned before, in conventional, traditional approaches where a data file is serving functional, non-integrated systems requirements, the problems of this deficiency in data handling are apparent, but where a change in a data file may have ramifications throughout every system in the company--this deficiency cannot be tolerated.



Without this data independence or transparency at the element or field level, the systems designer has no realistic choice except to attempt to anticipate every data requirement for every system that may be effected before any development can be begun. This may mean months or years of systems research which will ultimately be only marginally correct. The obvious delays and post-ponements of systems progress are of course expensive not only in increased implementation costs, but also in the very real costs of not achieving the benefits of the systems objectives as quickly as might otherwise be possible. A system which will save $50,000 a year, if it is a year late, really costs your company that $50,000.

THE SYSTEM MUST ELIMINATE DATA REDUNDANCY

Exclusive of control information there should be no toleration of duplicate or redundant data. The problems of duplicate maintenance and inconsistencies are already well known from traditional systems approaches. If the DBMS is truly data independent or transparent at the data element or field level and also provides for flexible data structuring techniques, there will be no need for the system designer to build redundantly.

However, limitations or characteristics of certain data associations or relation techniques virtually force systems designers to intentionally build duplicity since they have no efficient alternative. The solutions to data redundancy and data associations can best be illustrated by a brief discussion of these two closely related objectives.

Among traditional computer files required by a manufacturing company, the order entry system may have an order file containing one record for each outstanding order. Each record may be identified by a unique order number and contains such information as the customer's name and the quantity, number, and description of each item ordered. The accounts receivable system may use an account file in which each record represents one customer's account. Each record is identified perhaps by a customer number, and includes such information as the customer's name and the amount of his account. In addition, the company will maintain an inventory file. Each record there may be identified by the item number of one of the company's products and the record contains all information pertinent to that particular item. Obviously, a great quantity of data is duplicated on two or more of the company's files.

In order to overcome this problem, the company decides to develop a data base. Each piece of information should be stored only once. However, since all departments of the company will use the data base, the data must be organized in such a manner that each department can access what it requires. One of the problems is the question of how to "key" into the data base. If the order number is chosen as the only entry point to the

data base, the accounts receivable department cannot easily obtain the amount outstanding for a particular customer's account. If customer number is chosen as the key to data base records, warehouse employees cannot determine the quantity on hand for a particular item. Thus the data base must be designed so that variable entry points, corresponding to different applications and requirements, are available. The data must be organized such that each department can retrieve pertinent information efficiently, yet avoid redundant data. The methods of representing logical relationships of information by physical storage organizations will be examined here.

Certain data base systems currently in use employ some form of tree data structure to represent logical relations of information. "A tree structure is a hierarchical structure in which each element may be related to 'n' elements at any level below it, but to only one element above it in the hierarchy. However, the highest element is known as the root of the tree and has only dependents."[1] Figure 1-1 represents a tree structure.

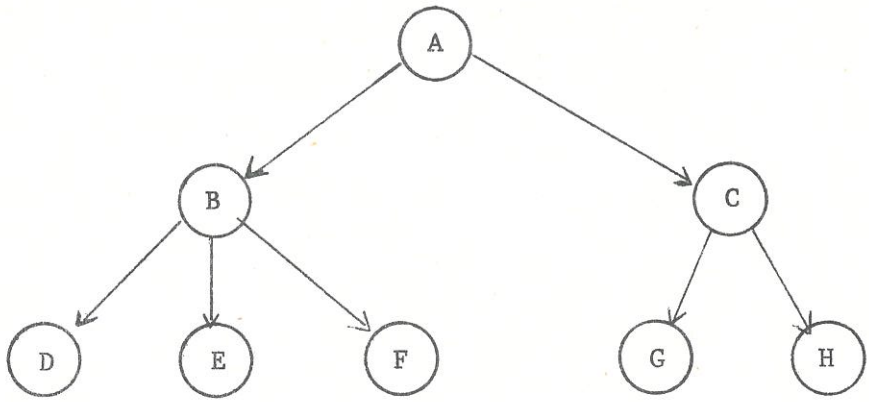


Figure 1-1 Tree Structure - One Way Hierarchy

Element 'A' is the root of the tree. Each element has zero or more branches leaving it. For example, three branches leave element 'B' to enter elements 'D', 'E', and 'F'. However, each element (except 'A' which is the root) has exactly one branch entering it.

The hierarchical tree is suitable for representing some data structures. For example, consider university students enrolled in courses. Each student may be represented as the root of a tree. Each course in which he is enrolled can then be considered a dependent element of the student root as shown in Figure 1-2. A collection of student trees may form the data base and the entry-point to the data base would be the student, the root of each tree.

[1]Codasyl Data Base Task Group, October 69 Report (1969), pp. 2-20.

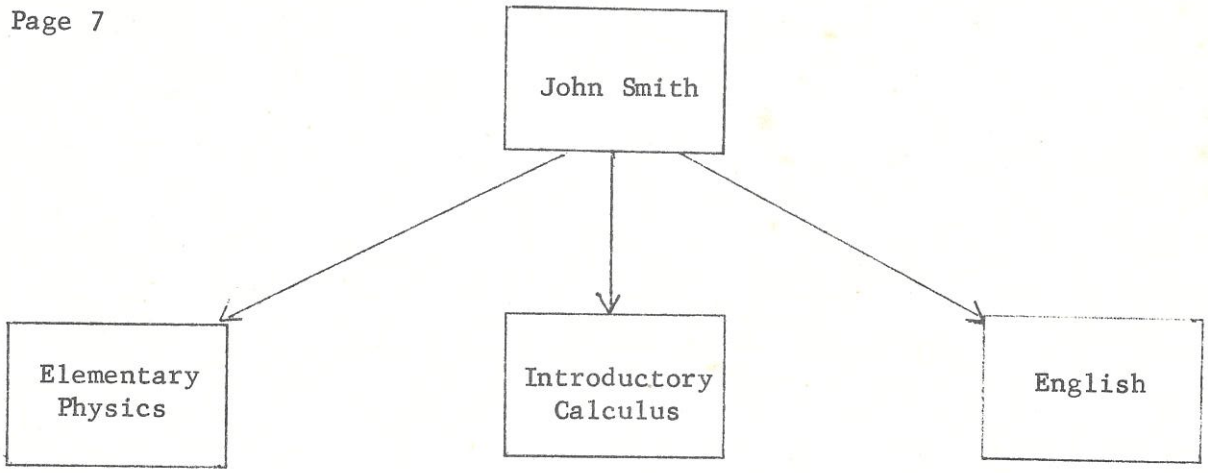


Figure 1-2 Tree Representation of Student Program

Similarly, it may be desirable to know all students enrolled in each course. In this case, the course is considered to be the root element and the enrolled students the dependent elements. This structure is shown in Figure 1-3.

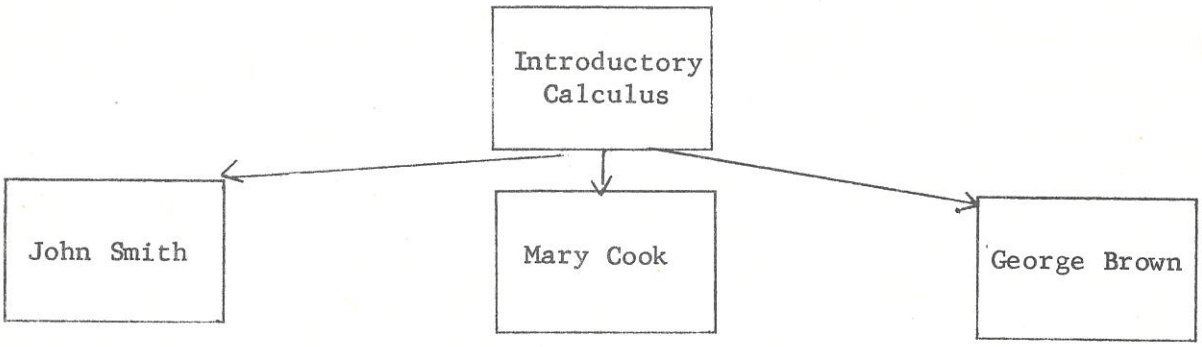


Figure 1-3 Tree Representation of Course Enrollment

The collection of course trees would form the data base, and the root element, the course, would be the entry to the data base.

Suppose now that we wish to implement a data base for this university. Class instructors will wish to enter the data base through a course to learn all students enrolled in their courses. People concerned with

students will wish to enter the data base via an individual student to access all data concerning the particular student. If either one of the examples above is chosen, one group of users will be unable to enter the data base to access required data efficiently. If both 'data bases' are used, a great deal of information is duplicated. If the problems of redundant data are to be resolved, a more desirable structuring of the data base such as shown in Figure 1-4 is essential. This technique of data structuring or association is known as a "network structured data base".

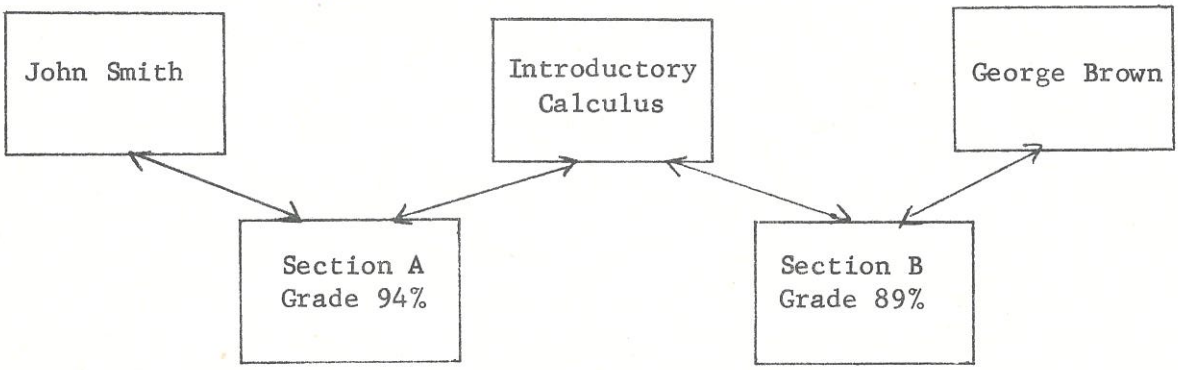
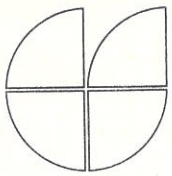


Figure 1-4 Data Base Structure Communication in any Direction

The data base can be entered on either course or student and no duplication of data exists. As in the previous examples, the dependent elements represent information which exists because a particular student is enrolled in a particular course. The student's course section and grade are such data. However, because dependent elements are entered by more than one branch, the structure shown is not a tree structure. The information desired cannot be efficiently represented by a tree.

The problem encountered above is a common one. It arises because the natural relationships of the information retained by most organizations are too complex to conform to a tree structure solely. Some other structure is required to ensure a flexible and efficient data base.

A network is the most flexible and powerful of data structures. In a network, any element may relate to any other number of elements. In particular, this implies that any number of branches or pointers may both enter and leave any element. Thus, conflicting problems to achieve multiple entry points, flexible and powerful data structures, data associations and data non-redundancy within the data base are resolved.



DATA INTEGRITY AND SECURITY ARE ESSENTIAL

Data integrity and security are frequently used interchangeably. Although both are considerations, the importance of integrity (protection) is paramount. Every safeguard against hardware malfunction, bad data, programmer or operator error must be anticipated. The DBMS must protect the data base, but in the unlikely possibility of error speedy and efficient restart and recovery techniques must be available. The techniques must function in all environments, multiprogramming, multiprocessing, etc., and must be comprehensive. The critical nature of data integrity cannot be stressed too strongly. It is a must.

Data security (privacy) should be such that data may be secured at the data base, data file, or data element level. Since security will change over time, easy techniques for implementation and modification are essential. Further, the unique security requirements of users will vary so that if desirable they should be able to accommodate these requirements through their own security program interface.

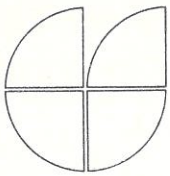
THE SYSTEM MUST BE ENVIRONMENT INDEPENDENT

Environment

Since the user will evolve over time through a number of environments, batch, on-line inquiry, on-line updating or may be functioning in a number of different environments concurrently, the DBMS must be independent or transparent of environment. Conversion to take advantage of new hardware or software technology must be eliminated.

As an example, the DBMS and all applications should be able to function in a real-time environment, but must not be locked to the specific techniques of a particular terminal monitoring and control system. This particular area has been and will continue to be dynamic. Without freedom to be responsive to the break-throughs as they occur--either hardware or software "front-ends", the early data base implementors, will find themselves very shortly frozen into obsolete technology and forced once again into a traumatic conversion effort.

The only safeguard against this serious penalty of pioneering is to be certain that the DBMS approach is one of complete environment, equipment and operating system independence.



THE SYSTEM MUST BE HOST LANGUAGE AND EQUIPMENT INDEPENDENT

Language and Equipment

Selection of a host language (COBOL, PL/1, etc.) is made to best satisfy the current and intermediate requirements of the user. Certain businesses may have mixed requirements using both Fortran and COBOL for example. Users have also evolved from one language to another; Autocoder to COBOL to PL/1. Since these conditions will probably exist for some time to come, it is imperitive that the DBMS must be independent of the host language. Technical personnel using Fortran must be able to work through the DBMS with the same data files as the COBOL or PL/1 programmer. Businesses currently using COBOL or PL/1 must be able to evolve forward as new, more efficient languages are made available in the future.

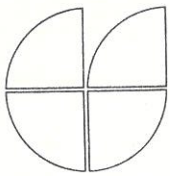
This same independence should be available with respect to type of equipment and vendor. For example, companies which have multiple equipment vendors or different operating systems should be able to utilize a standard across the board data base approach. The DOS user and the OS user should be compatible with true program and data base transferability. The current conveniences are important, but the future freedom of choice is vital. Vendor and hardware independence is essential so that the user may take advantage of new technology that his current vendor will offer or to evaluate and choose between vendors in the future without massive conversion effort penalties.

Without independence at the DBMS level, the user is "locked in" to today's technology even though he may have chosen COBOL in the hope of avoiding this disadvantage.

OPTIMUM PERFORMANCE AND EFFICIENCY ARE CRITICAL

In conventional systems and data file design much attention is usually given to speed of performance and efficiency of storage space. This is important even though only a few applications or programs may be affected. With a data base approach, since all programs and applications are affected, speed and efficiency are imperative. The problems are further complicated because the objectives typically work against each other--minimum disk and core space but maximum through-put. Further, degradation and data base reorganization must be avoided--again for the reasons that all programs are adversely affected in a less than optimum data base environment.

Because of the critical requirements of performance and efficiency in a DBMS, trade-offs or compromises in this area should not be tolerated. The DBMS must provide extremely efficient and high-speed performance in all environments.



The level of sophistication of the DBMS space management and allocation is extremely critical. Frequently, approaches are taken where additions to the data base are stored in an overflow area. Since this has serious undesirable degrading effects, systems designers in an attempt to overcome this problem provide for record storage capacities well in excess of normal requirements. The penalty here of wasted storage space can become very expensive. As discussed earlier compromises can provide gains in one area which may be lost many times over in another. Ideally, space management and allocation techniques must be inherently available within the DBMS which overcome all of these problems--with little, if any, concern on the part of the user.

THE SYSTEM MUST PROVIDE AN EASY TRANSITION APPROACH

The ultimate conversion from the traditional application oriented non-integrated systems to an integrated data base approach will take place over an extended period of time. Most users already have a major investment in existing systems which should be protected. Although the advantages of data base may be obvious--so are the potential problems of conversion. In order to eliminate the problems of redundant data files to serve these existing systems while similar data is being placed in the new data base files for new systems, a technique or approach must be available which will enable the user to efficiently protect his investment with a minimum conversion effort.

The DBMS ideally would enable the user to make only minor modifications to his existing systems in the data access areas (READS and WRITES) and begin to use the new data base files. In this way the user would gain a number of very desirable advantages:

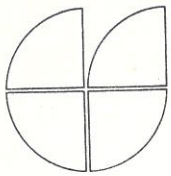
1. He would protect the investment in existing programs. In reality he may even enhance that old investment since these programs would utilize the same data files as all new systems and many advantages of the data base environment would immediately be gained.

2. He could concentrate more time on new systems design and implementation thereby gaining the savings potential or operational benefits sooner.

3. Redesign of old systems could be done in a planned, organized manner rather than a "crash basis"--the result should be much better systems.

4. He would eliminate the prolonged running of dual systems approaches with the added burden of interfacing between the old and the new.

5. He would immediately eliminate redundant data.



SYSTEMS AVAILABILITY

The TOTAL DBMS offered by Cincom Systems of Cincinnati, Ohio is one system which accomplished all of the above design objectives - and in a very efficient manner. Due to the successes of the users of TOTAL, it has become the most widely and successfully implemented system of its type in the country, and is currently available for IBM, Honeywell and RCA equipment.

Frank H. Veith, Jr.
Cincom Systems, Inc.
2181 Victory Parkway
Cincinnati, Ohio 45206
513/961-4110