1981

Physical Database Design: A DSS Approach*

J. V. Carlis
University of Minnesota

S. T. March
University of Minnesota

G. W. Dickson
University of Minnesota

Physical Database Design: A DSS Approach*

J. V. Carlis
Department of Computer Science
University of Minnesota

S. T. March
Department of Management Sciences
University of Minnesota

G. W. Dickson
Department of Management Sciences
University of Minnesota

ABSTRACT

This paper presents a working decision support system for use in the physical design of a database. Physical database design, although a structured decision problem, lends itself to a decision support approach because closed form algorithms are computationally infeasible. The paper describes the physical database design problem, presents an overview of a software system for use in solving this problem, and evaluates the use of the system in solving a sample problem.

INTRODUCTION

Decision support systems (DSS), although a relatively recent phenomena, have been applied in a variety of settings. Corporate planning (McLean & Riesing, 1977), the medical area (Davis, 1977), the determination of advertising budgets (Little, 1970), and personnel administration (Berger & Edleman, 1977; Edleman, 1981) are but a few examples of areas of DSS applications. In each of these instances a DSS approach has been adopted because of the nature of the problem. The problems have been unstructured, or semi-structured, and the decision support system has been an interactive computer based system which has helped the decision maker by providing data and models for application in the problem solving process (Carlson, 1977; Sprague, 1980).

There is another related setting in which a decision support approach is very appropriate. This is a situation in which the decision is structured and a single decision criterion is present, say, minimum cost, but no analytical optimization algorithm exists and there are too many alternative solutions to test them all in a feasible time period. The well known police beat allocation problem addressed by Carlson and his colleagues (Carlson, Grace, & Sutton, 1977) is such a situation, at least for a given decision criterion.

Another problem having these characteristics is that of the design of a large physical database. Theoretically the

*This work was supported in part by the David W. Taylor Naval Ship Research and Development Center under Research Contracts N00167-79-00140 and N00167-80-C-0061

designer, call this person the Database Administrator (DBA), can choose a design which minimizes the total operating cost over some time period. This can be done by evaluating all possible combinations of design parameters and choosing the one having the lowest cost. Unfortunately, even a modestly sized problem has so many possible combinations that a manual approach is clearly infeasible.

In such cases, a computer is frequently employed to facilitate the evaluation process. Here again, however, the sheer number of solution alternatives makes even this approach infeasible. A solution to the small, sample problem which will be discussed in this paper, for example, is estimated to take thirteen centuries on the CDC Cyber 74 computer to evaluate all possible combinations of solution (design) parameters. A "real world" problem which is being addressed by the authors for the U.S. Navy is conservatively estimated to require 10^{18} centuries of time on the same computer system for its solution, were an enumerative procedure to be employed. An approach to the solution of problems of this sort is to develop a decision support system to assist a human decision maker in arriving at a "good" solution to the problem with a reasonable expenditure of computer and human resources.

This paper describes how a decision support approach was applied to the physical design of a database--a problem type not previously discussed in the DSS literature.[1] The following section will describe, in non-technical terms, the nature of the problem and the decision parameters involved in its solution. The next section will present an overview of the solution procedure with emphasis on those aspects having DSS properties. The concluding section will evaluate the efficiency and effectiveness of the DSS approach in this area of application.

THE DATABASE DESIGN PROBLEM

Database design is a challenging task. Because of the size and complexity of the problem and interdependence among various aspects of the design, the best data organization is seldom obvious. As James Martin says:

> ...the designer is confronted with a complex array of alternatives. The more alternatives he can consider in a rational fashion, the more likely he is to produce an optimal design. Many of the poor designs of database systems (and there are many) result from a design considering only certain of the alternatives. The majority of systems analysts have a limited range of knowledge and are sometimes enthusiastic about particular techniques which they understand well to the exclusion of others, some of which might be better (Martin, 1975).

A myriad of different data organizations can satisfy a given set of required uses. Evaluating the performance of even one design involves thousands of calculations involving such factors as:

1. volume and volatility of the data,
2. characteristics of data retrieval,
3. physical equipment attributes,
4. data redundancy, and
5. intrinsic structure of data.

---

[1] The DSS approach to physical database design has been suggested by writers in the Very Large Database literature, but not many persons working in the DSS area would encounter this material (Gambino and Gerritsen, 1977; Hoffer, 1980).

It should also be noted that design performance has several measures (e.g., access time, costs, storage requirements, and complexity), many of which can be in conflict.

Because of the number of design parameters and performance tradeoffs, it is impractical to expect a human designer to select, unaided, an optimal or nearly optimal physical design. The result of "poor" database design is excess cost to organizations in terms of poor system performance or excessive computer resource requirements and, in some cases, avoidance of database applications because of perceived impossibility. In order to appreciate the complexity of the database design process, an overview of the process will be helpful.

The Database Design Process

Navathe and Schkolnick suggest that, "for any information system it is generally accepted that there are two levels of design, logical and physical." They define each as follows:

1. Logical Database Design--It consists of integrating the requirements of a number of applications/users to arrive at a centrally controlled and maintained logical database structure. The central structure must support individual user views of the data and support their processing needs. In order to store the database in a particular database environment, the structure should be defined in terms of the facilities, features, or constraints existing under that environment.

2. Physical Database Decision--Definition of a logical database still leaves a number of possibilities for its implementation in a certain database environment. Physical database design involves the evaluation of such alternatives and a choice of storage structure, placement strategies, and searching mechanisms, etc. (Navathe and Schkolnick, 1978)

The research reported upon in this paper addresses point Number 2. Figure 1 is an overview of the physical database design process.

The physical database design process has three inputs and one output. The inputs are:

1. Data description: Specification of the objects contained in the database (entities), their attributes, relationships, identifiers, cardinality, etc.

2. Uses of data: Specification of updates and report content, order, frequency, etc.

3. Computer system environment: Specification of channel transfer rate, disk access times, etc.

The output of the process is a database design characterized by a grouping of elementary data items into records (types) and a set of storage structures on the records which together form access paths to the data. The process chooses a design which will yield a good performance when implemented. Performance is measured by the sum of storage, retrieval, and maintenance costs.

The inputs to the process are assumed to be known and available. The first two input types come from the logical database design process. The third comes from the given characteristics of present or planned equipment. The methods available for logical database design are primitive and the determination of the logical problem is

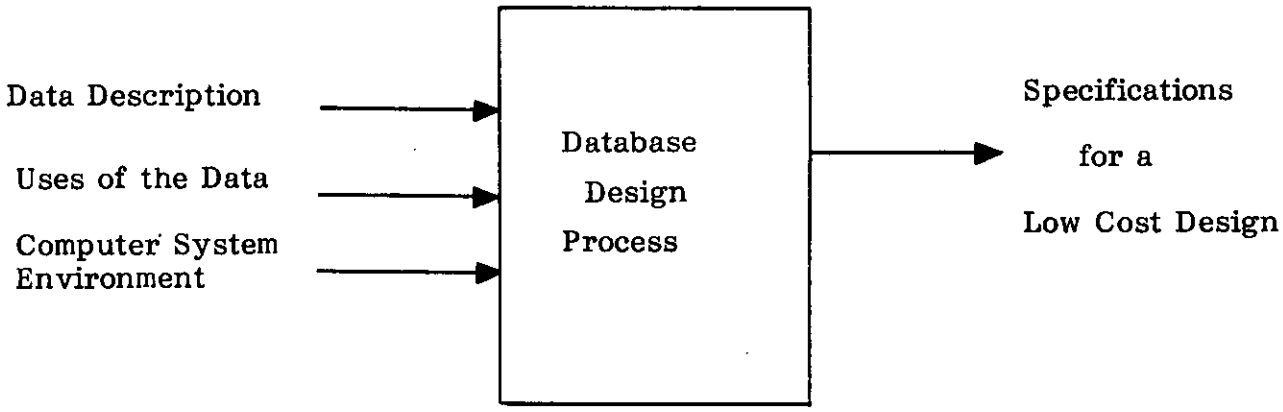

Figure 1. The Physical Database Design Process

difficult (IBM 1978 New Orleans Database Design Workshop). However, much research is focused on the process of "problem definition" or "information requirements analysis" (Teichroew & Sayani, 1971; Teichroew & Hershey, 1977; Rockart, 1979; Gane & Sarson, 1979; Munro & Wheeler, 1980) and progress in this area is being made. The resulting logical description of the problem specifies the types and cardinal number of entities in the database, descriptors of these entities, and the relationships between entities as well as the uses of this data.

Figure 2 shows the logical data structure of a sample problem involving workers within an organization. The particular graphical representation of the problem (one of several available methods) is based upon Senko's "infological model" (Senko, 1975). Figures 3.1, 3.2 and 3.3 show partial listings of other types of input data needed to develop a physical design from the logical data structure depicted in Figure 2. The characteristics of the computer environment are given in Figure 4.

From this type of problem description, the designer (DBA), must select a good physical level design which satisfies all of these logical level requirements. The logical content of the database, for example, must be organized into datasets composed of record instances at the physical level. Retrievals must be supported by access paths and updates to the database must be accomplished via some type of maintenance mechanism. A single dataset with its associated access paths and maintenance mechanisms we call a file organization. A physical database consists of a set of interconnected file organizations. Thus, the DBA must specify the following types of design parameters:

1. Some numbers of records (types), the union of which covers all elements of the logical data structure--each record defines the content of a file organization (its dataset);

2. For each file organization thus defined:

    a. Record formats describing the grouping of data items into record segments (for efficiency purposes "smaller, high use"

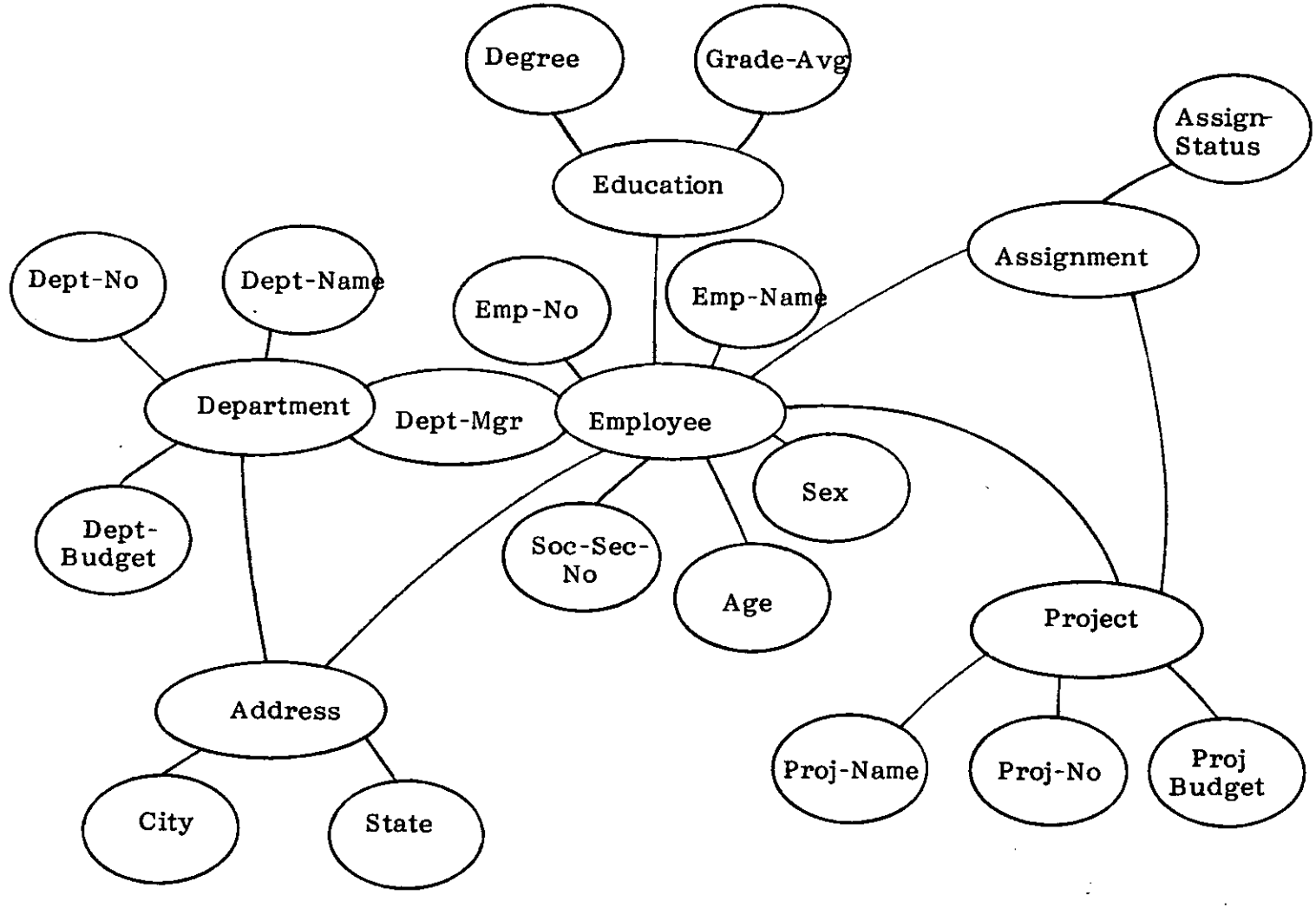

Figure 2. A Graphical Representation of a Sample Logical Data Structure (LDS)

3.1 Data Description (partially shown)

Entity Descriptor	Identifier	Average Length (Characters)
Department		
Dept-no	Dept-no	4
Dept-name		30
Address		
City name	City-name	20
State-name	State-name	2
Employee		
Emp-no	Emp-no	10
SSN		9
Emp-name		30
Salary		10

3.2 Data Volume and Update (partially shown)

Entity	Quantity	Additions/ Month	Delections/ Month	Modifications/ Month
Department	150	0	0	1
Employees	30,000	500	500	500
Assignments	60,000	1,500	1,200	5,000

3.3 Typical Reports (partially shown)

- Departments and their employees, ordered by department number then by employee name.
- Assignments ordered by employee.
- Employees in the Shipping Department ordered by employee name.

Figure 3. Data Description and Use (Partial Example)

Random Access Time	43.3 milliseconds
Local Access Time	36.3 milliseconds
Data Transfer Rate	1.13 MB/second
Length of a Memory Block	13030 characters
Memory Blocks per Locality	19
Length of a Memory Pointer	4 characters
Maximum Sort/Merge Order	2
Cost of Storage	$.68/locality-month
Cost of Retrieval Time	$.01/second
Cost of Maintenance Time	$.01/second
Cost of Buffer Space	$2.27 x 10 /char-sec
Expected Life	12 months

Figure 4. Computer System and Operating Environment Description

data items may be stored on primary segments while "larger, low use" data items are stored on secondary segments);

b. A dominant access path which establishes the ordering and positioning of record instances;

c. Auxiliary (secondary) access paths which connect selected subsets of record instances;

d. Maintenance mechanisms (procedures and system data) to manage storage space for update operations; and

3. Interconnections among file organizations (typically implemented by direct or symbolic pointers between record instances in different file organizations).

Many alternatives exist for each of the above. For example, the number of alternative records is exponential in the number of relationships and for each record the number of alternative record formats is exponential in the number of data items in that record. Clearly the solution space from which a design must be selected is enormous.

Design Complexity and the DSS Approach

The fact that the DBA must choose a set of physical database design parameters from among such a population of solution alternatives results in a large combinatorial problem. Obviously, the selection of a set of physical storage structures must be based upon the logical data structure, its volume, the frequency with which data is updated, the formats and frequency with which it is retrieved, and the access and

cost characteristics of the media on which it is stored. All these parameters affect system performance, measured by storage and processing costs, in interdependent and complex ways.

Consider the following example. Suppose two different users need the information about the entity, "assignment" (from Figure 2), but need it in a different order. Ordering assignment by project may yield fast access for one user, but it may be very slow for another. Ordering assignment by employee may produce opposite results. One solution is to keep two copies of the record instances, one in each order. This would produce fast access, but high storage and update costs. Another possibility the DBA could choose would be to order assignments by some other criterion (e.g., to reduce update costs) and form indexes according to project and employee.

Many other solutions could be chosen by the DBA, even for this simple illustration. Many more possibilities exist as the problem complexity expands. The question is--how is the DBA to pick a set of design parameters which has optimal, or nearly optimal, performance and cost from among the vast array of possibilities?

We suggest that the DBA can make these choices more intelligently if supported by a machine system that performs calculations to aid in the evaluation of solution alternatives and that contains mechanisms which reduce the size of the problem and help to structure the solution process. In terms of the well-known Simon model of decision making, the DBA needs machine assistance in the phases involving design and choice. The first phase, intelligence, which deals with searching the environment for conditions calling for a decision, is part of the logical database design process and is outside of the scope of this research. Design (developing and analyzing possible courses of action) and choice (selecting a course of action) are the phases of decision making addressed by this research. In Figure 5, we show the activities of the person and the machine (and their relationship in the physical database design process as reflected in the database design system whose characteristics are described in the following section.

THE DATABASE DESIGN SYSTEM

To date, traditional mathematical optimization techniques have not been successfully applied to the solution of the physical database design problem. The problem contains integer decision variables, nonlinear constraints, and discontinuous objective functions. Furthermore, there are many nonquantifiable factors which must be considered. As an alternative approach, a decision support system has been suggested.

In particular, Gambino and Gerritson, 1977, and Hoffer, 1980 suggest the utility of a DSS approach to the solution of the physical database design problem. Although file design tools have been developed, the only operational DSS addressing database design is that of Gerritsen (1975), but this system is limited in its ability to model logical data structures and physical access paths. Furthermore, this system simply evaluates designs in contrast to optimizing parts of the design.

The research reported in this paper goes well beyond previous efforts (e.g., SODA, see Nunamaker, Konsynski, Ho, & Singer, 1976). Our research provides a multiple level descriptive model of the database design problem space and its solution space. In addition to the descriptive model, a software system is provided to automatically generate "good" physical database designs. In addition the software may be used to evaluate particular designs suggested by the DBA.

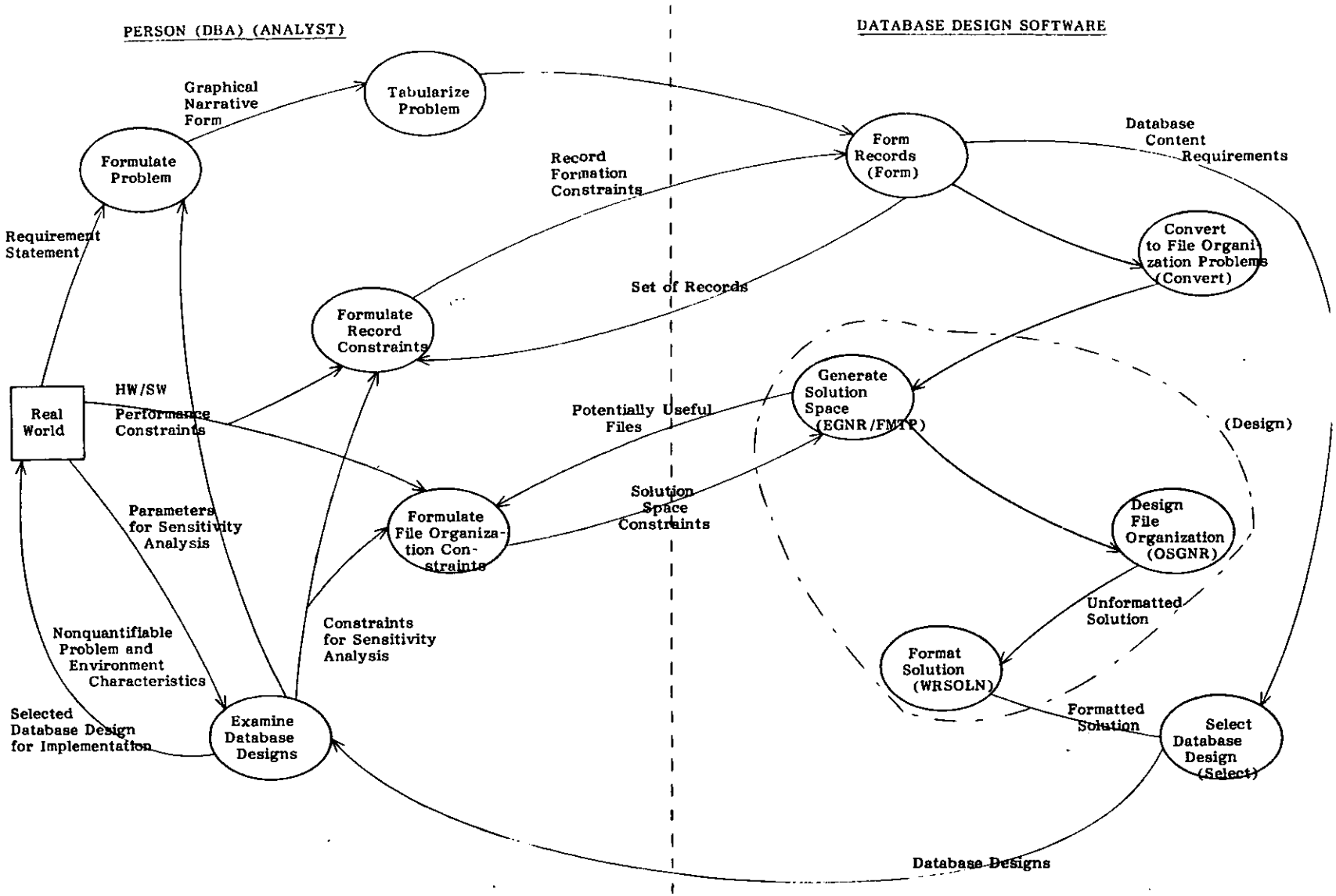

Figure 5. The Activities of the Person and Machine System

This person/machine system is built upon the strengths of both of its components. The person (DBA) exercises judgment and constrains the solution space. The machine is used to optimally solve various design subproblems and to evaluate the performance of alternative designs constructed from those solutions and the constraints imposed by the DBA. The specific roles of each party will be shown in the following sections of this paper.

Figure 6 shows the four major modules of the design system and the flow of information among them. Input to the system is: (1) a statement of the information requirements of the user community including the logical data structure (LDS), data volumes, and retrieval and update characteristics as discussed earlier and, (2) commands and constraints, e.g., controlling the level of detail of the output produced by the system, and/or dictating generic types of solutions to be considered (record structures, access paths, etc.), or specifying partial solutions (records, individual access paths for specific retrievals, etc.). Output from the system is a database design and performance estimates for this design.

Ideally, the system will produce optimal file organization designs for an optimally generated set of records. However, problem and solution characteristics affect system performance in such complex ways that they defy exact analysis. Thus, in order to guarantee the selection of a globally optimal design, all possible combinations of model parameters would have to be evaluated. Unfortunately, as discussed above, the number of possible design alternatives is so large that, even with an automated design evaluator, such a brute force approach is computationally intractable.

The current design system has both analytic and heuristic parts. It requires the designer to guide the search of the solution space and includes a number of design heuristics as well as optimization and evaluation algorithms which are incorporated into the modules shown in Figure 6. The approach is, in essence, a classic example of a DSS. The operation of the design system is briefly described below (for a more detailed discussion of the exact nature of the human interface and the procedures used by the various modules, the reader is directed to Carlis, 1980):

1. FORM creates sets of records (types) for which file organizations will be designed. Each set contains or "covers" all of the logical data structure; each record covers one or more entities. We restrict the formation of records as follows:

   a. all instances of a record are in the same file organization,

   b. all attributes of an entity are in the same record,

   c. relationship descriptors[2] are represented by one of five representations; symbolic pointer, direct pointer, both symbolic and direct pointers, absorption (repeating groups of attributes or relationship descriptors) and no representation.

These five representations allow us to bound the computing task facing FORM. If there are NE entities and NR relationships then an upper bound on the number of records and therefore on the computing

[2]For each relationship there are two relationship descriptors, one describing each entity in the relationship, e.g., in Figure 2, there are both department-of-employee and employees-of-department relationship descriptors.

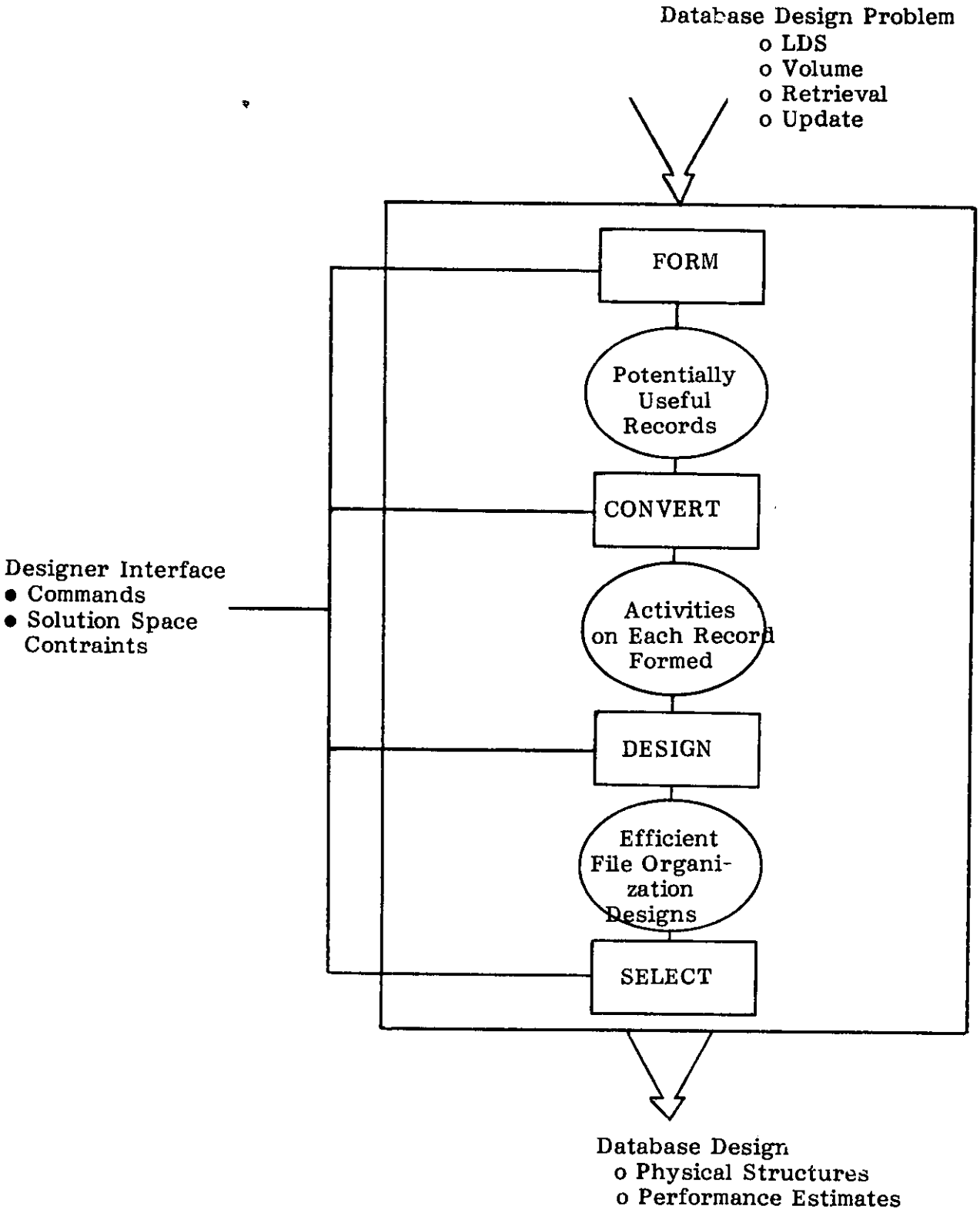

Figure 6. Database Design Modules

done by the design software is NE*5** (2*NR) (or NE*25**NR). This bound is large even for small values of NR. To reduce the computational burden we use these techniques:

i. Limit representations. Not all representations will be efficient. FORM has embedded within it heuristics (Carlis, 1980) which will eliminate most of the possible relationship representations. The heuristics lower the upper bound to NE*3**NR, which still results in a large number of records. The DBA can selectively override the heuristics, suggesting one of several representations (based on, for example, company policy, DBMS contraints, and knowledge about the problem, perhaps anticipating future needs).

ii. Identify subproblems. Using the above limited representations, FORM searches for subproblems, called connected graphs (of entities). Relationships between entities in different connected graphs have exactly one (non-absorbing) representation while those between entities in the same connected graph may have several possible representations. Connected graphs define subproblems because the cost of a file organization depends on which representations are chosen for intra-connected graph relationships, but not on which are chosen for inter-connected graph relationship is fixed by definition). If several connected graphs are constructed then the upper bound is significantly reduced; it becomes a sum of products. Let NCG be the number of connected graphs, NECG(j) be the number of entities in the i(th) connected graph, NRCG(i) be the number of relationships in the i(th) connected graph, and NREP(ij) be the number of representations for the j(th) relationship in the i(th) connected graph. Then the upper bound is

$$\sum_{i=1}^{NCG} NECG_i \prod_{j=1}^{NRCG_i} NREP_{ij}.$$

In general, the heuristics mentioned above will not produce multiple connected graphs. The DBA can reduce the computational load by examining the LDS, locating sets of entities which are nearly disconnected by the heuristics and specifying single representations for some number of relationships so that multiple connected graphs will result.

iii. Eliminate infeasible skeletons. FORM enumerates sets of records for connected graphs by systematically varying representations. We call each set a skeleton. A skeleton may be found to be infeasible and be discarded. Infeasibility is caused by representations dictating an entity's absorption into different records (violates the first restriction above) or dictating a circuit of absorption (has no root entity for the record).

iv. Identity duplicate records. In enumerating skeletons, FORM will create records which will appear in more than one skeleton. Processing is saved since FORM identifies these duplicates and a file organization is designed (i.e., DESIGN is executed) only once for each unique record. Later, in SELECT, skeleton costs are computed by summing the proper file organization costs.

2. CONVERT changes processing requirements expressed in the logical level model to accessing patterns of the records formed in (1) effectively yielding a set of file organization design problems.

3. DESIGN produces an efficient file organization design for each problem from (2). It has three major components: (a) a probabilistic model which evaluates the expected performance of alternative main-

tenance mechanisms, (b) a bicriterion[3] mathematical programming algorithm which selects an optimal assignment of data items to primary and secondary segments (Hammer, 1979, and Hoffer, 1975, provide a motivation for and discussion of record segmentation), and (c) a branch and bound algorithm which selects an optimal set of data access paths from a set of potentially useful data access paths which are heuristically generated. The heuristics for generating potentially useful data access paths are based upon the activity which must be satisfied. Auxilliary access paths such as lists and inverted lists are generated for all retrievals requiring less than ten percent of the data records. In addition, identifier searching access paths such as hashing, ISAM, full indexes, etc., are generated for all individual record retrievals and for update operations.

The proper execution of each component of DESIGN is dependent upon the results from each of the others. Therefore, DESIGN proceeds by heuristically generating an initial record segmentation (primary segments and secondary segments are set to equal lengths) and an initial set of data access paths (all retrievals requiring less than one percent of the data record instances are directly accessed, all others are sequentially accessed). The three components are then executed in sequence using the actual results from previously executed components. The record segmentation algorithm is then executed again using the access paths selected by the branch and bound algorithm. If the assignment of data items to primary and secondary segments has changed, then the procedure iterates once.

Again, the designer may optionally constrain the solution space by specifying partial solutions to be considered or even complete solutions for evaluation only. This is accomplished, for example, by specifying a set of maintenance mechanisms to be evaluated and/or by modifying the set of potentially useful access paths generated. In addition, the record segmentation algorithm can be bypassed, further restricting the solution space to single segment records. Finally, specific access paths may be explicitly defined for all or some of the activity supported by the file organization.

Output from DESIGN is an efficient combination of maintenance mechanisms, record segmentation, and data access paths along with performance estimates for each file organization designed. Performance is measured by a composite of storage, retrieval, and maintenance costs. The content and the form of the output produced by the system are discussed in the following section of this paper.

4. SELECT picks a set of file organizations which efficiently meet the user information requirements. This module automatically selects the most efficient (lowest cost) set of file organizations from among those produced by DESIGN. The total cost of a database is the sum of the skeleton costs for each connected graph. A skeleton's cost is the sum of the costs of the file organization for each of the skeleton's records. Since there are many non-quantifiable factors which are not considered by DESIGN and which may have considerable impact on overall database performance, the designer may use SELECT to display a set of alternatives produced by DESIGN and which is within some percent of the least cost design. The DBA may then subjectively evaluate this set of design alternatives and/or perform

---

[3]The algorithm trades off the storage and retrieval costs associated with a large primary segment against the volume of retrieval activity which would have to access secondary segments were the primary segment smaller (March 1976).

sensitivity analysis by varying critical design factors prior to selecting a design for implementation.

FORM, CONVERT, DESIGN, and SELECT are the modules of our database design system (DBDS). These programs represent approximately 10,000 lines of executable FORTRAN code. The DBDS is operational on both IBM and Control Data computer systems with the test results presented in this paper being generated on the latter system (CDC Cyber 74).

While the design system cannot guarantee optimality, database designs produced by it have been deemed intuitively reasonable and substantially more efficient than simplistic designs (e.g., a set of flat files) generated for the same problem. In addition, designs are quickly produced (see the next section for timing) and evaluated by the system, thus providing a vehicle for sensitivity analysis on critical design parameters and a benchmark against which to evaluate the performance of proposed alternative solutions.

APPLICATION OF THE DESIGN SYSTEM

While the authors serving as the human component (the DBA function), the system has been applied to a number of design problems. To illustrate a design solution, the problem described in Figures 2 and 3 was input to the system together with a description of the computer system and operating environment in which the database was to be implemented (see Figure 4). With no additional input (i.e., using only the design heuristics described above), a database design was automatically generated in approximately thirty seconds of computer time.[4]

A summary of the database design selected and estimates for fixed storage and maintenance costs for each of its component file organizations are given in Figure 7. Figure 8 shows the assignment of retrieval activities to file organization access paths and summarizes total costs by file organization. The DBDS also provides additional design and performance details which are omitted here.

The database design process described above is based upon the standard decision support system procedure of imbedded models and heuristics coupled with an interactive process. The division of labor between human and machine is as follows. In summary, the role of the designer (see Figure 5) is to: (1) abstract the problem from the real world and express it in the form of the logical level model, (2) judiciously constrain the solution space and execute the DBDS software (machine), and (3) subjectively evaluate the set of efficient database designs produced by the DBDS (machine) and select a design for implementation. The role of the DBDS software (machine) is to quickly search the constrained solution space for efficient database designs (by optimally solving various design subproblems) and/or evaluate subsets of the solution space as directed by the designer.

The design system also provides a convenient means for comparing the performance of various design alternatives. For example, the solution selected by the design system has three file organizations, one of which (the employee record) is hierarchically structured (contains repeating

---

[4]The time is, of course, for one iteration by the DBA testing and combination of variables. This timing figure is drastically affected by the problem size. Since the illustrative problem used here is small, this figure is low. As can be imagined, a real problem of, say, ten times the size would take substantially more computer time. The judgment of the DBA is critical in controlling total computing time.

FILE ORGANIZATION
Contains DEPARTMENT information:
DEPT-NO
DEPT-NAME
DEPT-BUDGET
MANAGER-OF-DEPARTMENT (EMP-NO)
ADDRESS-OF-DEPARTMENT (CITY,STATE)
Data Records are Unsegmented

Access Paths	Key Data Items	Selection Criteria	Fixed Costs (Storage and Maintenance)
1 Unordered Sequential File			

FILE ORGANIZATION 2
Contains EMPLOYEE information:
EMP-NO
EMP-NAME
AGE
SEX
SOX-SEC-NO
DEPARTMENT-OF-EMPLOYEE (DEPT-NO)
EDUCATION-OF-EMPLOYEE ([DEGREE, GRADES]*)
ASSIGNMENT-OF-EMPLOYEE ([STATUS, PROJ-NO]*)
ADDRESS-OF-EMPLOYEE (CITY, STATE)

Data Records are Unsegmented

Access Paths	Key Data Items	Selection Criteria	Fixed Costs (Storage and Maintenance)
1 Hased Full Index	EMP-NO	All	$1.83
2 ISAM	EMP-NAME	All	2.02
3 Hasing via a Scatter Table	EMP-NO	All	.04
4 Cellular I/L	DEPT-NO	All by Dept	.02
5 Cellular I/L	DEPT-NO	All by Proj	.01
6 File Scan	None	All	.00
			$3.92

FILE ORGANIZATIONS 3
Contains PROJECT information:
PROJ-NO
PROJ-NAME
PROJ-BUDGET
MANAGER-OF-PROJECT (EMP-NO)

Data Record Are Unsegmented

Access Paths	Key Data Items	Selection Criteria	Fixed Costs (Storage and Maintenance)
1 Unordered Sequential File	None	All	$ .26
			$ .26

Figure 7. A Database Design Summary

*The notation [ ] indicates a nested repeating group.

Retrieval	Retrieve From File Organization	Retrieve From Access Path	Retrievals Per Month	Retrieval Cost Per Month
1	1 2	1 4	5 150	$ .01 2.44
2	1 2	1 1	10 500	.02 .32
3	2	2	20	.27
4	2	6	1	.26
5	2	3	3000	2.26
6	2 3	5 1	18 2	.29 .01

Total Retrieval Cost per Month	$ 5.88
Total Fixed Cost per Month	4.23
Ttoal Operating Cost per Month	$10.11

Figure 8. Assignment of Retrieval Activities

groups for education and assignment). Clearly, a database containing only flat files (i.e., normalized relations) is simpler and more flexible (many extensions to the database content will not disturb the existing file organizations). In order to determine the operational cost of obtaining this design simplicity and added flexibility, the design system was invoked again and the solution space constrained to flat file representations (by limiting the relationship representations as discussed above). The resulting design and its costs are summarized in Figure 9. Total monthly operational costs are seen to have increased by an order of magnitude from $10 to $111 per month. Provided with such performance estimates, a designer can more objectively decide if increased design complexity and its associated implementation costs are justified by significant savings in operating costs.

Sensitivity Analysis

As discussed above, the design system requires as input a parametric description of the data, its uses, and the computer system environment in which the database will be operated. The process of determining these parameters is ill-defined and the problem description is likely inexact. Therefore a DBA is concerned with the sensitivity of the database design to fluctuations in these parameters. The design solution presented above is relatively stable for a number of variations in the problem parameters. We executed the DBDS for nine candidate sets of records and for each candidate we tried twenty different variations in the problem parameters. The parameters varied were: blocking factor (one or four blocks/track; reorganization interval (between six and thirty-six months); device contention (random and local accesses were nearly the same, or widely different); and maintenance mechanisms (six different ones were employed).

The following results were observed.

1. The design software worked. The results were deemed reasonable by the authors.

2. The ranking of the candidate sets of records was stable across the variations in parameters. Thus, at least for this simple problem, the choice of records is the most important decision.

3. For the best candidate set of records the access paths were stable across variations of parameters with only minor exceptions.

4. Varying parameters primarily affected costs rather than structure.

    a. A longer block has the effect of increasing costs for individual record retrieval, due to a longer block retrieval time, but of decreasing costs for scanning since the number of blocks which must be accessed is reduced.

    b. A longer reorganization interval has the effect of increasing costs due to an increased use of overflow areas for updates.

    c. Where there was contention for a device the time for local access and, therefore, its cost, was higher.

    d. Maintenance mechanisms using direct pointers were found to be cheaper where there was little update.

This sensitivity analysis is typical of the kind of testing a DBA might do. Other parameters which might reasonably vary

Retrieval	File Organization		Retrievals per Month	Retrieval Cost per Month
1	DEPARTMENT EMPLOYEE	Unordered Sequential Cellular I/L (DEPT-NO)	5 150	$ .02 4.23
2	DEPARTMENT EMPLOYEE	Unordered Sequential Hashed Full Index (EMP-NO)	10 500	.03 .50
3	EMPLOYEE ASSIGNMENT	ISAM (EMP-NO) Cellular I/L (EMP-NO)	20 2000	.33 12.20
4	EMPLOYEE ASSIGNMENT education	Scan Scan Cellular I/L (EMP-NO)	1 2 3000	.39 .06 21.97
5	EMPLOYEE ASSIGNMENT	Hashed Full Index (EMP-NO) Cellular I/L (EMP-NO)	3000 3000	2.97 25.55
6	ASSIGNMENT PROJECT	Unordered Sequential Unordered Sequential	18 2	.51 .01

Total Retrieval Cost per Month	$ 68.77
Total Fixed Cost per Month	42.41
Total Operating Cost per Month	$111.18

Figure 9. A Flat File Database Design

are the frequency of updates or retrievals, the number of entity instances, attribute lengths, relationship degrees, etc.

A DSS EVALUATION

As can be inferred from the above discussion, a DSS approach is appropriate for application to the database design problem. The methodology presented involves man/machine interaction at several stages in the process. Heuristics are employed to further reduce the problem sapce and the computational complexity. Imbedded models of an optimizing type are also utilized where appropriate. These characteristics, we argue, establish the presence of a decision support system.

The DSS provides a problem solution and, as far as analysis has shown, a reasonably good one. Sensitivity analysis supports the reasonableness of generated solutions and demonstrates the flexibility of the process.

As computer analysis is applied to large and complex problems, even of a structured nature, our experience with the database design problem suggests that a DSS approach is an attractive one to pursue. The combination of human judgment and machine efficiency is powerful even when heuristics embedded in the solution procedure do not allow the optimality of the solution to be measured. We suggest that it is possible that the problems being addressed with the tool described in this paper are among the most complex worked on to date with a DSS procedure. That the DSS approach works so well here suggests that it is a fruitful method to consider as we expand computer application to new areas involving large, complex problems.

REFERENCES

Berger, P. and Edleman, F. "IRIS: A Transaction-Based DSS for Human Resources Management," Data Base, Volume 8, Number 3, Winter 1977, pp. 22-29.

Carlis, J. V. An Investigation into the Modeling and Design of Large, Logically Complex Multi-User Databases, Unpublished Ph.D. Dissertation, University of Minnesota, Minneapolis, Minnesota, 1980.

Carlson, E. D. "DSS Conference Overview," Data Base, Volume 8, Number 3, Winter 1977, p. 2.

Carlson, E. D., Grace, B. F., and Sutton, J. "Case Studies of End User Requirements for Interactive Problem-Solving Systems," MIS Quarterly, Volume 1, Number 1, March 1977, pp. 51-63.

Davis, R. "A DSS for Diagnosis and Therapy," Data Base, Volume 8, Number 3, Winter 1977, pp. 58-72.

Edleman, F. "Managers, Computer Systems, and Productivity," MIS QUARTERLY, Volume 5, Number 3, September 1981, pp. 1-20.

Gambino, T. J. and Gerritson, R. "A Data Base Designs Decision Support System," Proceedings of the Third International Conference on Very Large Databases, Tokyo, Japan, October 1977, pp. 551-557.

Gane, C. and Sarson, T. Structured Systems Analysis, Prentice-Hall, Englewood Cliffs, New Jersey, 1979.

Gerritsen, R. "A Preliminary System for the Design of DTBG Data Structures," Communications of the ACM, Volume 18, Number 10, October 1975, pp. 551-557.

Hammer, M. and Niamir, B. "A Heuristic Approach to Attribute Partitioning," Proceedings ACM SIGMOD, June 1979, Boston, Massachusetts, pp. 93-100.

Hoffer, J. "Database Design Practices for Inverted Files," Information and Management, Volume 3, 1980, pp. 149-161.

Hoffer, J. A. and Severance, D. G. "The Use of Cluster Analysis in Physical Data Base Design," Proceedings of the International Conference on Very Large Data Bases, Framingham, Massachu-

setts, September 1975, pp. 69-86.

IBM 1978 New Orleans Database Design Workshop, IBM Technical Report Number RJ2554, IBM Corporation, San Jose, California, 1978.

Jefferson, D. "The Development and Application of Data Base Design Tools and Methodology," Proceedings of the Sixth International Conference on Very Large Databases, Montreal, Canada, October 1-3, 1980.

Little, J. D. C. "Models and Managers: The Concept of a Decision Calculus," Management Science, Volume 16, Number 8, April 1970, pp. 466-485.

March, S. T. and Severance, D. G. "A Mathematical Modelling Approach to the Automatic Selection of Database Designs," Proceedings of the ACM SIGMOD, Austin, Texas, 1978, pp. 112-126.

March, S. T. and Severance, D. G. "The Determination of Efficient Record Segmentations and Blocking Factors for Shared Data Files," ACM TODS, Volume 2, Number 3, September 1977, pp. 279-296.

Martin, J. Computer Data-Base Organization, Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

McLean, E. R. and Reising, T. F. "MAPP: A DSS for Financial Planning," Data Base, Volume 8, Number 3, Winter 1979, pp. 9-14.

Munro, M. C. and Wheeler, B. R. "Planning Critical Success Factors and Management's Information Requirements," MIS Quarterly, Volume 4, Number 4, December 1980, pp. 27-38.

Nunemaker, J. F., Konsynski, B. R., Ho, T., and Singer, C. "Computer-Aided Analysis and Design of Information Systems," CACM, Volume 19, Number 12, December 1976, pp. 674-687.

Navathe, S. and Schkolnick. "View Representation in Logical Database Design," SIGMOD Proceedings, Austin, Texas, 1978, pp. 144-156.

Rockart, J. F. "Chief Executives Define their Own Data Needs," Harvard Business Review, Volume 57, Number 2, March-April 1979, pp. 81-93.

Ross, D. T. and Schoman, K. E. "Structured Analysis for Requirements Determination," IEEE Transactions on Software Engineering, Volume 3, Number 1, January 1977, pp. 53-67.

Senko, M. E. "Specification of Stored Data Structures and Desired Output Results in DIAM II with FORAL," Proceedings of the Conference on Very Large Data Bases, Framingham, Massachusetts, 1975, pp. 557-587.

Sprague, R. J., Jr. "A Framework for the Development of Decision Support," MIS Quarterly, Volume 4, Number 4, December 1980, pp. 1-26.

Teichroew, D. and Sayani, H. "Automation of System Building," Datamation, Volume 17, Number 3, August 1971, pp. 25-30.

Teichroew, D. and Hershey, E. A. "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems," IEEE Transactions on Software Engineering, Volume 3, Number 1, 1977, pp. 15-27.