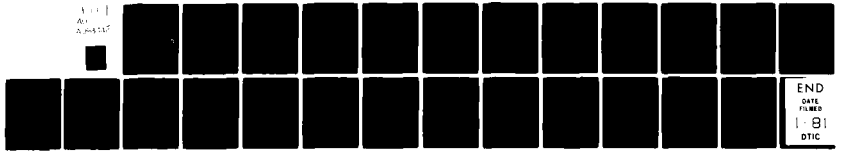
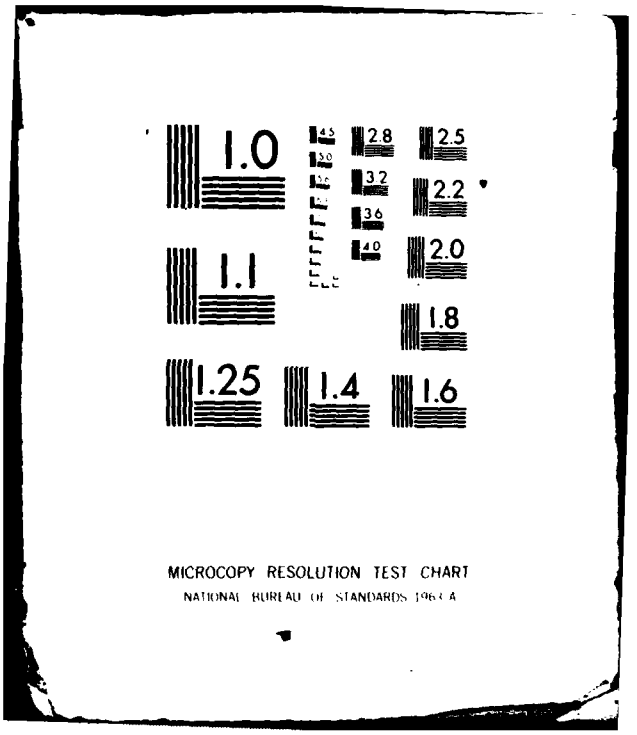
AD-A093 442 WHARTON SCHOOL PHILADELPHIA PA DEPT OF DECISION SCIENCES F/G 5/2
DATA BASE DESIGN FOR DECISION SUPPORT,(U)
OCT 80 E K CLEMONS N00014-75-C-0462
UNCLASSIFIED 80-10-02 NL



END
DATE FILMED
1-81
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
32
36
40
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 80-10-02	2. GOVT ACCESSION NO.: AD-A093442
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): DATA BASE DESIGN FOR DECISION SUPPORT	5. TYPE OF REPORT & PERIOD COVERED:
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Eric K. CLEMONS	8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0462
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Decision Sciences, The Wharton School, U. of PA, Phila., PA 19104	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Task NR049-272
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research.	12. REPORT DATE: October 1980
	13. NUMBER OF PAGES: 24
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 12 27	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report):

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report):

DISTRIBUTION UNLIMITED

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number):

database management; DSS; index selection;

20. ABSTRACT (Continue on reverse side if necessary and identify by block number):

A satisfactory user interface is essential for a decision support system to gain acceptance, and system performance is an important component of the user interface. Several suggestions for data base design are presented to improve system performance and response time, and thus to improve the user interface.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601 | SECURITY CLASSIFICATION OF THIS PAGE (When Data

408757

AD A093442

DDC FILE COPY

DTIC ELECTE JAN 5 1981 D C

80 12 30 005

Data Base Design for Decision Support

Eric K. Clemons

80-10-02

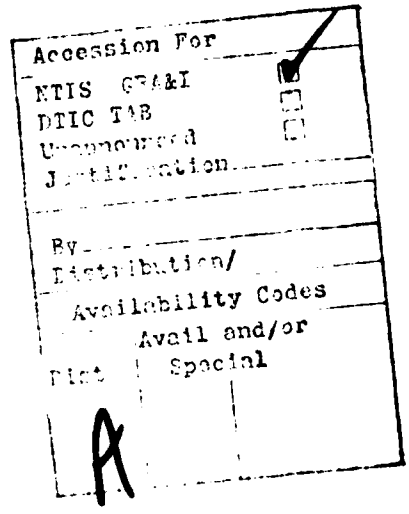
Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, Pa 19104

Data Base Design for Decision Support

Eric K. Clemons

ABSTRACT

A satisfactory user interface is essential for a decision support system to gain acceptance, and system performance is an important component of the user interface. Several suggestions for data base design are presented to improve system performance and response time, and thus to improve the user interface.



Data Base Design for Decision Support

Eric K. Clemons

1. Introduction

Information is required for decision making. This is true whether the information is quantitative or qualitative, and whether the decision is subjective and based on intuition, highly structured and model-based or algorithmic, or somewhere in between. The nature of both the decision and the decision maker will vary, thus nature of the information used will vary; still, some contact with the surrounding reality will be required.

The evolving discipline of data base management does have something to offer to the implementor of decision support systems. Its forefront or leading edge of theory need not be considered yet: data dictionary/directory systems, information resource management, or semantic data models for conceptual schemas will be of importance to decision support systems in the future but are not yet ready for export outside the data base community. Data base design, however, is an important component of decision support systems.

Unsuccessful DSS implementations are frequently defeated, not by technical failure, not by use of the wrong models or algorithms, not by software development errors on the part of the development team. Rather, DSS implementations that fail are often killed by a poor user interface, leading to poor reception by the user of the system or reluctance of the user to conclude that his personal performance using the DSS will improve enough to justify his commitment of time. This belief that use of the DSS will improve performance sufficiently to compensate the user for his investment in time to learn and to use the system is one of the rationales for DSS development (see Keen, "Decision Support and the Marginal Economics of Effort" [5]). In the absence of this perception of the DSS as worthwhile, it will ultimately be rejected.

While good system performance and good response time do not guarantee a good user interface, intolerable system performance and intolerable response time are sufficient to preclude an acceptable interface, and thus to preclude user acceptance. Here data base design enters. Quite simply, when data and supporting auxiliary structures such as indices are properly designed, performance improves.

As is well known, it is extremely difficult to predict the information requirements for the unstructured analyses and unanticipated requests of strategic decision making [3]. Fortunately, the average DSS imposes some structure: It is for advertising budget preparation or ambulance and fire service

redistricting or portfolio management; it is not for all three plus the study of mergers and acquisitions. We offer here some guidelines for designing decision support data bases, exploiting this structure. We also draw on experience with file design for machine peformance, with interactive systems, and with clients' solutions to their problems. Some recent data base research is incorporated where directly applicable. Common sense is also enthusiastically endorsed.

2. An Overview of Five Guidelines

Five guidelines for data base design for decision support are presented below in summary. They are treated in greater detail, with examples, in the subsequent section.

1. Exploit knowledge of traditional, file-based inquiry systems

2. Design for the specific nature of DSS use

3. Keep auxiliary data for DSS use

4. Note when the auxiliary data become obsolete

5. Recalculate auxiliary data rapidly

Traditional, file-based sytems use auxiliary data structures to permit rapid access to subsets of the file that are of interest, that is, records that are required to respond to a specific query or generate a specific report. These structures are usually inversions or indices and lists threaded through the principal file. They are useful when the approximate distribution of requests -- what is being requested, by whom, and how often -- is known in advance. They are most useful when the approximate distribution of data values is also known.

But decision support systems are not inquiry systems and have different requirements. Each DSS will impose some structure upon the decision maker and upon his requests. The implementor should design for the requirements of the DSS being built. In particular, the data base should be designed for the verbs supported by the DSS: e.g., PLOT for MONTH, HISTOGRAM, SEARCH for MAXIMUM. Don't design the data base to support a general inquiry facility unless this will also be provided.

Keep auxiliary, derived data in summary form for use by the DSS. The transaction processing data base will frequently be enormous. This full detail is almost never required by a DSS for strategic planning; rather, averages, totals, maxima, or rates of change are needed. Where possible, extract these interesting data items in off-line, off-hours runs using the transaction data base. These derived data are available for DSS use without the slow, time-consuming, and expensive scan of the full data base. Wherever possible, keep the DSS away from the transaction

processing data base.

Unfortunately, while this separation of decision support and transaction processing permits improved response time by the DSS, it does so at a cost: some of the derived data computed off-line and available to the DSS will be rendered inaccurate or out-of-date by the updates of the transaction processing system. For those derived items for which currency is essential, it is reasonably inexpensive to add to the transaction processing system the capability to detect when an update to the data base will alter the item so that recalculation is advisable. Three options are then available:

1. recalculate immediately

2. recalculate the item if it is requested by the DSS

3. recalculate the item only if it is requested by the DSS and the user indicates that complete accuracy is essential and the user indicates willingness to wait for the item to be recalculated or to return for the result

Alternatively, we may keep still additional auxiliary data to permit the transaction processing system to recalculate the derived data rapidly. Thus the transaction processing system can update both the transaction processing data and the decision support data at the same time. If the auxiliary data are well-selected, this requires no degradation of transaction processing performance, while providing timely and accurate data

rapidly for DSS use. A cautionary note: as we shall see in the following section, this can sometimes be done, and done quite easily. However, it requires that we analyze very carefully the data that will be useful to the DSS.

3. Some Examples of Data Base Design for Decision Support

Five guidelines for data base design for decision support were suggested in the previous section. Here, each is described in greater detail and illustrated by example.

3.1. File-Based Inquiry and Decision Support

File-based inquiry techniques use auxiliary data structures to extract rapidly subsets of a large file needed for a report or to respond to a query. Such techniques are useful when the DSS requires that the decision maker examine subsets of far larger collections. They can be employed only when some predictions are available for the distribution of requests and the distribution of data values.

Such file-based systems rely on direct access devices, which provide the ability to access requested records if a record key or device address is known, and upon two data structures: list or multi-list structures and inversions. In a conventional file, each record comprises a list of (attribute, value) pairs, e.g.,

LOAN:(LOAN-ID:127,542), (LOAN-OFFICER:CLEMONS),
(ORIGINAL-PRINCIPAL:12,000,000). . .

In an inverted file, this conventional organization is turned upside down. Instead, for a given (attribute, value) pair, we have a list of records that contain the pair. These records are represented by their addresses on a storage device or, more frequently, by an identifying attribute or key. Thus, an inversion on loan officer might include the following.

(LOAN-OFFICER:CLEMONS): 111,202, 123,411, 123,602, . . .

Thus, for a collection of records of interest, for example, those accounts with Clemons as approving loan officer, we have immediate and direct access. An example of an inverted file is given in figures 1 and 2.

In a list structured file, an interesting subset is identified and each record is tied to its successor by means of a pointer. The collection of attributes used to select a list subset is usually more complex than that used to construct an inversion. An example of a list structured file is also given in figures 1 and 2. For example, list 1 provides large British industrial loans for which Clemons was the original loan officer, linked together by key values in pointer 1.

Inversions provide for efficient access to a wide variety of queries, even those where details of the queries are not fully known in advance. For example, it is useful to know which attributes, or which collection of (attribute, value) pairs, will be used to describe subsets requested. It is not always necessary to know precisely how these attributes will be combined to form queries. This form of retrieval support is particularly valuable when some assumptions can be made on the distribution of values present in the data base for each value.

Some examples should illustrate the power of these simple techniques. Refer to the loan file shown in figures 1 and 2, containing 50,000 entries. Consider calculations to determine the number of records examined to retrieve the following subsets:

1. British loans
2. Large British loans for heavy industry
3. Large British loans for heavy industry with Clemons as loan officer

The calculations are shown in figure 3.

LOAN PORTFOLIO FILE

I. PRIMARY DATA RECORD: LOAN DESCRIPTION
50,000 RECORDS
600 CHARACTERS EACH
6 CHARACTERS IN KEY LOAN-ID

RECORDS CONTAIN THE FOLLOWING ATTRIBUTES:
LOAN-ID, BORROWING ORGANIZATION,
ADDRESS, FINANCIAL-OFFICER,
LOAN-OFFICER, RATE, TERMS,
PRINCIPAL, BALANCE, STATUS,
INDUSTRY, NATIONALITY, POINTERS

II. DISTRIBUTION OF VALUES

NATIONALITY			
BRITISH	15%	GERMAN	25%
FRENCH	20%	ITALIAN	25%
		SWEDISH	15%

INDUSTRY			
AUTO	10%	SERVICE	20%
FURNITURE	5%	SHIP BUILDING	5%
HOUSING/CONSTR	30%	SMALL APPLIANCES	10%
RUBBER/TIRES	5%	STEEL	5%
		TRANSPORTATION	10%

SIZE OF LOAN (IN THOUSANDS)		
SMALL	(<100)	25%
AVERAGE	(100-1,000)	45%
MODERATE	(1,000-4,000)	25%
LARGE	(4,000+)	5%

III. AUXILIARY DATA STRUCTURES

INVERSIONS: NATIONALITY
INDUSTRY
SIZE

LISTS: LARGE BRITISH LOANS FOR
HEAVY INDUSTRY WITH CLEMONS
AS LOAN OFFICER

(HEAVY INDUSTRY INCLUDES AUTO,
STEEL, AND SHIP BUILDING)

Figure 1--Description of a loan portfolio file

LOAN-ID	PRINCIPAL	INDUSTRY	OFFICER	NATIONALITY	POINTER 1
0			HEAD1		123,411
111,202	40,000	SERVICE	CLEMONS	IT	
114,653	400,000	SERVICE	KEEN	IT	
122,404	500,000	AUTO	DAVIS	IT	
122,515	350,000	CONSTR	KEEN	FR	
122,800	50,000	FURN	JONES	GER	
123,411	7,500,000	SHIP	CLEMONS	BRIT	123,602
123,602	4,250,000	STEEL	CLEMONS	BRIT	133,213
125,152	1,075,000	APPL	SMITH	GER	
125,908	500,000	SERVICE	KEEN	GER	
127,804	400,000	RUBBER	JONES	SWD	
128,614	5,000,000	STEEL	DAVIS	SWD	
131,400	750,000	TRANS	SMITH	BRIT	
133,213	8,000,000	STEEL	CLEMONS	BRIT	135,434
133,667	10,000,000	SHIP	KEEN	BRIT	
135,432	4,350,000	AUTO	CLEMONS	IT	
135,434	4,211,000	AUTO	CLEMONS	BRIT	141,006
137,707	250,000	SERVICE	DAVIS	FR	
140,876	20,000	SERVICE	HARRIS	GER	

a.) Portion of a loan file, including a list. Note use of loan with key 0 to identify head of list.

NATIONALITY				
BRITISH:	123,411,	123,602,	131,400,	135,434
FRENCH	122,515,	137,707		
GERMAN	122,800,	125,152,	125,908,	140,786
ITALIAN	111,202,	114,653,	122,404,	135,432
SWEDISH	127,804,	128,614		

b.) Portion of the inversion on nationality, corresponding to file sample shown in a above. Similar inversions would be constructed for industrial classification, principal, and other attributes of interest.

Figure 2--A sample of the loan file described in figure 1 and a corresponding inversion.

It should be clear from the calculations that these techniques offer substantial performance improvements over full sequential scan of all records in the file. It may be less clear that considerable analysis is necessary if the appropriate inversions and lists are to be selected for inclusion in the file, out of the total collection of all such structures possible. For example, it is extremely unlikely that the list used to respond to query 3 would actually have been identified as useful when the file was constructed. Rather, several inversions would be used to get large British industrial loans and the system would be programmed to ignore those for which Clemons was not loan officer.

Several references are available for this material in tutorial form [7]. Recent research in the automatic selection of attributes on which to index will probably not be directly applicable to the DSS designed, but will help the designer develop his own procedures for manually identifying those attributes [4].

QUERY1: BRITISH LOANS

OPTION1--FULL SCAN OF 50,000 RECORDS

OPTION2--USE BRITISH INVERSION

EXPECTED NUMBER OF BRITISH LOANS
15%x50,000 = 7,500
EXPECTED SIZE OF BRITISH INVERSION
15%x50,000x1/100 = 75

TOTAL COST OF OPTION2:7,575
SAVINGS OF 85% OVER OPTION1.

QUERY2: LARGE BRITISH LOANS FOR HEAVY INDUSTRY

OPTION1--FULL SCAN OF 50,000 RECORDS

OPTION2--USE BRITISH INVERSION, 7,575 RECORDS READ

OPTION3--USE BRITISH INVERSION INTERSECTED WITH LARGE LOAN INVERSION

SIZE OF INVERSIONS:
BRITISH:15%x50,000x1/100 = 75
LARGE LOANS:5%x50,000x1/100 = 25
100

EXPECTED NUMBER OF LARGE BRITISH LOANS
15%x5%x50,000 = 375

COST OF RECORDS PLUS INVERSIONS READ IS 475

OPTION4--USE BRITISH INVERSION INTERSECTED WITH LARGE LOAN INVERSION INTERSECTED WITH THE UNION OF (AUTO, STEEL, AND SHIPBUILDING INVERSIONS)

SIZE OF INVERSIONS
BRITISH: 15%x50,000x1/100 = 75
LARGE LOANS: 5%x50,000x1/100 = 25
AUTO: 10%x50,000x1/100 = 50
STEEL: 5%x50,000x1/100 = 25
SHIPBUILDING: 5%x50,000x1/100 = 25
200

EXPECTED SIZE OF SUBSET:
15%x5%x(10%+5%+5%)x50,000 = 75

COST OF RECORDS PLUS INVERSIONS READ IS 275

TOTAL COST OF OPTION4:275
SAVINGS OF 99.45% OVER OPTION1.

QUERY3: LARGE BRITISH LOANS FOR HEAVY INDUSTRY
WITH CLEMONS AS LOAN OFFICER

OPTION1--FULL SCAN OF 50,000 RECORDS

OPTION2--USE OF INVERSIONS AS IN OPTION4
OF QUERY3, COST OF 275 READS

OPTION3--USE LIST1. COST CANNOT BE DETERMINED
WITHOUT KNOWLEDGE OF DISTRIBUTION OF
LOANS BY OFFICER, BUT WORST CASE
ASSUMING ALL LARGE BRITISH LOANS
FOR HEAVY INDUSTRY ARE THROUGH
CLEMONS IS COST OF 75.

TOTAL COST USING OPTION3: AT MOST 75
SAVINGS OF 99.85% OVER OPTION1
SAVINGS OF 72% OVER OPTION2.

Figure 3--Sample cost calculations for three queries, using the file presented in figure 1.

3.2. Design for the Specific Use of the DSS

Design for the specific use of the verbs supported by the DSS being constructed and not for general inquiry and reporting, unless these facilities are also to be supported. We illustrate this principle with a simple example. Consider development of a DSS to compare stocks for inclusion in a portfolio. Our system will include high resolution full color graphics display, now available on several inexpensive micro-computers. We will compare stocks by plotting some statistic like daily closing price for a month or monthly high for two years.

Traditionally, the data base used to capture stock exchange information stores the data for a single stock on a single day in a single record; this record contains numerous data items, such as the following:

```
STOCK:  (EXCHANGE CODE, NAME DATE, YEAR,
     OPENING PRICE, CLOSING PRICE, DAILY VOLUME,
     DAILY HIGH, DAILY LOW, 12 MONTH HIGH,
     12 MONTH LOW, YEAR-AGO PRICE, DIVIDEND)
```

This organization was selected probably because it closely parallels the format in which the data were originally acquired. It has been maintained because it is adequate for the pattern of use to which the data are put in a general inquiry system on a high speed main-frame computer.

However, to plot two stocks' closing price over a month we recall approximately 40 records. On a micro-computer with diskette or on a heavily congested main-frame with disk contention from other users we may wait several minutes for the full plot, which is clearly intolerable response time in a system intended for interactive use in decision support. Worse still, note that plotting monthly highs for the two stocks over two years will be twenty-four times slower, requiring perhaps one or two hours.

An alternative data structure readily suggests itself, once the verbs of the DSS are recognized: rather than place the ten or fifteen data items for the stock for a single day in one record, use one record of the same length to capture three weeks of one data item such as closing price. Plot of two stocks' closing prices for one month will then require four or five record accesses in place of 40, for perhaps a ten-fold increase in speed.

We note that this is not the ideal layout for a system that is to respond to such requests as: DESCRIBE in FULL, IBM, 9-OCT-80. However, such a request is neither required nor permitted by the DSS being considered.

9. average lateness, delinquent loans

Clearly, this full collection can be generated off-line in a single pass through the transaction data base. And, equally clearly, several other derived data items may as plausibly be argued to be useful. The designer of the DSS should determine what, in fact, will actually be needed.

Assuming a moderately sized loan portfolio of 50,000 loans, queries can be answered:

1. in many minutes, using a full scan of all records in the folio

2. in only a few minutes if the necessary inversions are available and only a subset of the loans need be investigated to prepare a response

3. in at most a few seconds, using derived data stored on-line and updated nightly

There is one disadvantage of this approach: if we update derived-data nightly, then we may introduce a one-day lag between the time a transaction updates the main data base and the time this update is reflected in the derived data. Is this serious? Actually, it's usually not even significant. Since the items in the decision support data base are aggregate, derived from hundreds or thousands of entries in the transaction processing system, effects of individual updates are usually lost in the

mass. An on-line inquiry system requires up-to-date information if it is to permit approval of, for example, Clemons using his VISA card to purchase a ticket to Hawaii. Decision support data, permitting bank officers to decide if the bank should continue to offer VISA, need not be as current.

There will sometimes be a measurable effect due to the lag in updating decision support data, particularly if the sample used to derive them is a small fraction of the transaction base. Still, we must consider if the loss in management performance due to the lag exceeds the loss in performance due to early termination of the search for solutions caused by frustration over lengthy response time. It is the experience of some large commercial users that improved decision making permitted by adequate system speed more than compensates for the inaccuracies introduced by delay in updating derived data.

Sometimes, of course, lag in updating may significantly affect the resulting decision. Events such as prime reaching 24%, collapse of the stock market, or failure of a U.S. auto maker would no doubt rapidly affect a bank's lending policies. These are events that the manager would discover from other sources long before effects started showing up in his loan portfolio!

3.4. Let the System Warn When the Derived Data are Obsolete

Some updates by the transaction processing system clearly will, or should, alter the derived data items maintained for use by the DSS. A missed payment should affect the number of loans delinquent, the number of loans delinquent in its specific industry, and several other data items, yet these items are not recalculated until the next off-line run that rederives the entire collection. For some data and some decisions, this lag may be important.

The transaction processing system can be designed to determine automatically when an update to the transaction processing data base should also modify the decision support data. Several options are then available to the DSS designer. First, the derived data may be recalculated when the need is detected, without awaiting the next off-line run. Secondly, the recalculation can be begun when the user's request of the DSS causes it to reference derived data that are no longer current. Thirdly, the user may be notified that a request references data in need of recalculation; the user will then have the option of completing his DSS use rapidly with the existing derived data or awaiting recalculation of affected items. Unfortunately, any recalculation is likely to be slow and to make numerous disk accesses.

Alerting [2] is the process used to detect when derived data should be modified after update by the transaction system. Quite complex conditions can be detected, and detected very efficiently [1]. Alerting will be more valuable to the DSS designer when combined with the suggestion of the next section.

3.5. Keep Auxiliary Data to Permit Rapid Recalculation of Derived Data

Keeping additional auxiliary data, in addition to the derived data actually required by the DSS, will often facilitate recalculation of the useful derived data when the need is detected by alerters. In fact, this recalculation may be so rapid that it can be performed by the transaction processing system at the time the update is performed, eliminating the lag between transaction data and decision data updates.

This concept is illustrated by a single, simple example. Let our interest be in the average size of loans in our portfolio, by industry. Derived data are maintained that provide these averages for each industry. And additional data are kept, even if these are never directly referenced by the DSS: number of loans and total value of loans, for each industry. Creation of a new loan should, for the appropriate industrial group, alter the average loan amount kept for the DSS. This can be done rapidly, without reference to the transaction data base: For the industrial group affected by the new loan the number of loans is

increased by 1, the total value of loans is increased by the value of the new loan, and the number divides the total value. Comparably direct recalculation processes can be described for totals, maxima, minima, and numerous other derived data of use to a DSS. These processes permit recalculation of derived data by the transaction processing system at the time transaction data are updated, without access of the transaction data base and without degradation of transaction processing performance; this successfully eliminates potentially misleading loss of currency in the decision support data.

4. Conclusions

What conclusions can we draw from the preceding material?

The user interface can kill a decision support system, and inadequate system performance can kill a DSS. Therefore, the selected data base design must support system performance.

Several suggestions have been presented to guide data base design. All require some knowledge of the applications to be run and the problems to be solved. Thus, as always, internal involvement and user involvement will be essential for successful DSS implementation. No team of outside specialists, regardless of their competence or credentials, can design and install a DSS without considerable assistance from the ultimate users of the system.

Unfortunately, much of managerial work is unprogrammed and unpredictable: for many of the problems for which DSS is the most appropriate decision-aiding tool, accurately predicting use and information requirements does not appear possible. This problem is exacerbated by a sort of managerial "uncertainty principle": The presence of a successful DSS alters the type of problem that can be attacked and the solution procedures that can be employed. The DSS itself alters the nature of DSS use. Thus, even if the implementors' original data base design was ideal for the existing work patterns and information requirements, after the DSS has been in use it will probably be necessary for the implementors to return for modifications.

5. References

1. Buneman, O.P. and Clemons, E.K. "Efficiently Monitoring Relational Databases". ACM Transactions on Database Systems, Vol. 4, No. 3, Sept. 1979, pp. 368-382.

2. Buneman, O.P. and Morgan, H.L. "Implementing Alerting Techniques in Database Systems". Proc. COMPSAC 77, pp. 463-469.

3. Gorry, G. A. and Scott Morton, M. S. "A Framework for Management Information Systems". Sloan Management Review, Vol. 13, No. 1, pp. 55-70.

4. Hammer, M. and Chan, A. "Index Selection in a Self-Adaptive Data Base Management System". Proc. SIGMOD International Conference on Management of Data, June 1976, Washington, D.C., pp. 1-8.

5. Keen, P.G.W. "Decision Support and the Marginal Economics of Effort". Decision Sciences Working Paper 79-03-16, University of Pennsylvania, 1979.

6. Keen, P.G.W. and Scott Morton, M.S Decision Support Systems: An Organizational Perspective. Addison-Wesley, Reading, Mass., 1978.

7. Severance, D.G. and Carlis, J.V. "A Practical Guide to the Selection of Record Access Paths". ACM Computing Surveys, Vol. 9, No. 4, pp. 259-272.