

**a quarterly
bulletin
of the IEEE
computer society
technical
committee
on**

Database Engineering

Contents

Letter from the Editor	1	Systems and Techniques for Research in Physical Database Design at the University of Michigan	22
Research on File and I/O Systems at the University of California, Berkeley	2	T.J. Teorey, H. Aghili, R. Cobb, J.P. Fry, D.G. Severance, M.E. Wilens	
A.J. Smith		An Overview of Physical Database Design Research at the University of Minnesota	31
Physical Database Research at the University of Florida	5	S.T. March, J.V. Carlis	
D.S. Batory, S.Y.W. Su, S.B. Navathe		Physical Database Research at Stanford	39
DBDSGN—A Physical Database Design Tool for System R	9	G. Wiederhold, S.J. Kaplan, D. Sagalowicz	
S.J. Finkelstein, M. Schkolnick, P. Tiberio		Database Design Research at the University of Toronto	42
Physical Database Research at the Lawrence Berkeley Laboratory	12	S. Christodoulakis, A. Garg, C.C. Gottlieb, G.C. Magalhaes	
J. McCarthy, A. Shoshani		Database Storage Structures Research at the University of Waterloo	49
Physical Database Design Research at the University of Maryland	16	G.H. Gonnet, P.A. Larson, J.I. Munro, D. Rotem, D.J. Taylor, F.W. Tompa	
S.B. Yao and A.R. Hevner		Announcements	53
		Keyword Index	55

**Chairperson, Technical Committee
on Database Engineering**

Prof. Jane Liu
Digital Computer Laboratory
University of Illinois
Urbana, Ill. 61801
(217) 333-0135

**Editor-in-Chief,
Database Engineering**

Dr. Won Kim
IBM Research
K55-282
5600 Cottle Road
San Jose, Calif. 95193
(408) 256-1507

**Associate Editors,
Database Engineering**

Prof. Don Batory
Dept. of Computer and
Information Sciences
University of Florida
Gainesville, Florida 32611
(904) 392-5241

Prof. Alan Hevner
College of Business and Management
University of Maryland
College Park, Maryland 20742
(301) 454-6258

Dr. David Reiner
Sperry Research Center
100 North Road
Sudbury, Mass. 01776
(617) 369-4000 x353

Prof. Randy Katz
Dept. of Computer Science
University of Wisconsin
Madison, Wisconsin 53706
(608) 262-0664

Database Engineering Bulletin is a quarterly publication of the IEEE Computer Society Technical Committee on Database Engineering. Its scope of interest includes: data structures and models, access strategies, access control techniques, database architecture, database machines, intelligent front ends, mass storage for very large databases, distributed database systems and techniques, database software design and implementation, database utilities, database security and related areas.

Contribution to the *Bulletin* is hereby solicited. News items, letters, technical papers, book reviews, meeting previews, summaries, case studies, etc., should be sent to the Editor. All letters to the Editor will be considered for publication unless accompanied by a request to the contrary. Technical papers are unrefereed.

Opinions expressed in contributions are those of the individual author rather than the official position of the TC on Database Engineering, the IEEE Computer Society, or organizations with which the author may be affiliated.

Membership in Database Engineering Technical Committee is open to IEEE Computer Society members, student members, and associate members. (Application form in this issue.)

Letter from the Editor

This issue of Database Engineering is entitled "Directions in Physical Database Research". For the first time, status reports of significant research activities in files and physical databases have been assembled. The universities and research labs that are represented in this newsletter were invited to participate because of their long standing or recent contributions in this field. Owing to time and publication limitations, not all institutions and major research projects on physical databases could be represented. However, some of the most visible and significant projects, I feel, are described herein.

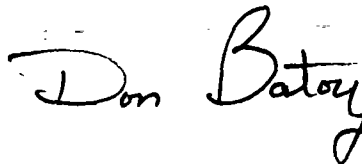
Research in files and physical databases is highly diversified. Topics range from theoretical to empirical studies of database performance; development of new file structures and database software to the development of models that relate and simplify recognized database structures, algorithms, and theories. Research on query processing and distributed databases, while closely related to physical database research, is given only cursory emphasis in this newsletter since they are the subjects of the September and December issues of Database Engineering.

There are ten different status reports covered in fifty pages, each report briefly describes several different projects. To maximize the utility of this publication, an index to key terms has been provided at the end of this newsletter. This index should facilitate the random accessing of discussions on selected topics and should provide a useful cross-reference of research interests.

I want to take this opportunity to thank again the contributors to this issue. Without their enthusiasm, dedication, support, and very hard work, this newsletter would not have been possible.

For those readers who care to comment on this issue, selected comments will appear in the September issue of Database Engineering. Please direct all letters to me.

It is my hope that the readers of this issue will find it as informative as I have discovered it to be.

A handwritten signature in cursive script that reads "Don Batory". The signature is written in dark ink and is positioned above the printed name.

D. S. Batory

RESEARCH ON FILE AND I/O SYSTEMS AT THE UNIVERSITY
OF CALIFORNIA, BERKELEY

Alan Jay Smith
Department of Electrical Engineering and Computer Sciences
415-642-5290

The author and his graduate students have been involved in research on a number of topics that can generally be covered under the subjects of file and I/O systems. In this report we summarize progress and results to date and sketch some future research plans. We also mention briefly some related work by other researchers at UC Berkeley. Included are some comments on our work on file migration, both hierarchical and across networks, cache disks and I/O system optimization.

In most computer systems the volume of files that users would like to have access to exceeds the capacity of the disk storage volumes available. Traditionally, the user has been responsible for storing his personal overflow on tape or other mass storage and recalling it when needed. File migration is concerned with automating this function, so that the system will automatically migrate inactive files to mass storage and will restore them, either on demand or on anticipated demand, to disk storage. There are a number of research problems connected with doing so. These research problems concern issues such as the creation of efficient (low miss ratio) algorithms for the selection of files to push to mass storage, the placement of such files on mass storage (especially with regard to fragmentation and compaction issues) and the selection of files (if any) to prefetch. Such problems must be studied using either real file systems or traces taken from real file systems, since file access patterns are an empirical phenomenon and modeling efforts will accomplish little unless they accurately represent the behavior of the real system. An empirical study of the replacement algorithm problem has been conducted by the author for one system ([SMIT81a-b]). Summaries of some of that work also appear in ([SMIT81c], [SMIT82a]). No work has yet been done for the fragmentation nor compaction problems nor for anticipatory fetching. We plan to study these issues as well as to study data for additional systems. (The author is very interested in obtaining additional file system trace data and would welcome any useable contributions.)

File migration can also be applied across distributed computer systems. In such a system, files may be located at either a central file server or at the devices attached to one of the CPUs. The issue then is to determine at any given time whether a remote file access should cause the file to be moved to the requesting CPU, whether the reads and writes should take place remotely, whether the process should be moved or whether some mixture of these schemes is appropriate. Research on this topic is in progress by the author and one of his graduate students.

Just as cache memories cache the contents of main memories in the CPU, cache buffers can be created which will retain the contents of portions of the disk resident file system. The idea is to cache tracks and cylinders from the disk address space in a level of storage intermediate between main memory and the disk; this storage can be made of CCDs, magnetic bubbles or dynamic MOS RAM. The locus of such a cache can be in the disk spindle, disk storage controller, the channel or at the CPU. Such caching was first shown to be effective in ([SMIT78a]) and has since been introduced in a number of commercial products. There are a number of research problems associated with the design of such a cache including those of cache location, capacity, block size, fetch and replacement algorithms, static vs. dynamic allocation, etc. A number of these issues have been extensively studied (see [SMIT82a-b]). Each of these problems will be considered further in studies to be heavily based on file system trace data. (The author is again very interested in obtaining additional trace data and would welcome any contributions.)

The last topic of relevance to physical data base systems is the consideration that the author has previously given to I/O system optimization and I/O architecture. A survey on this topic has been published in ([SMIT81d]) and an extensive bibliography appears in ([SMIT81e]). Readers may find those useful as a guide to the principal results and to the literature in the area.

There are other research projects at UC Berkeley which are also of some relevance. We include here the large ARPA funded project under the supervision of Professor Robert Fabry to extend and improve the UNIX operating system. Among the goals of that project is to improve and optimize the file system. There is also the Ingres Database System project led by Professors Stonebraker, Wong and Rowe which has been at times concerned with physical file structures. Readers interested in these projects should contact the researchers on those projects directly.

REFERENCES

- SMIT78a Smith, Alan Jay "On the Effectiveness of Buffered and Multiple Arm Disks", Proc. Fifth Computer Architecture Symposium, April 1978, Palo Alto, Ca., pp. 242-248.
- SMIT78b Smith, Alan Jay "Directions for Memory Hierarchies and Their Components: Research and Development", Proc. COMPSAC Conference, Chicago, Ill., November, 1978, pp. 704-709.
- SMIT81a Smith, Alan Jay "Analysis of Long Term File Reference Patterns for Application to File Migration Algorithms", IEEETSE, SE-7, 4, July, 1981, pp. 403-417.
- SMIT81b Smith, Alan Jay "Long Term File Migration: Development and

Evaluation of Algorithms", CACM, 24, 8, August, 1981, pp. 521-532.

- SMIT81c Smith, Alan Jay "Algorithms and Architectures for Enhanced File System Use", presented at the Second International Summer School on Computer Systems Performance Evaluation, Urbino, Italy, June, 1980. Published in Experimental Computer Performance and Evaluation, ed. D. Ferrari and M. Spadoni, North-Holland, 1981, pp. 165-193.
- SMIT81d Smith, Alan Jay "Input/Output Optimization and Disk Architecture: A Survey", Performance Evaluation, 1, 2, 1981, pp. 104-117.
- SMIT81e Smith, Alan Jay "Bibliography on File System and Input/Output Optimization and Related Topics", Operating Systems Review, 15, 4, October 1981, pp. 39-54.
- SMIT82a Smith, Alan Jay "Optimization of I/O Systems By Cache Disk and File Migration: A Summary", to appear, Performance Evaluation.
- SMIT82b Smith, Alan Jay "Cache Disks", in preparation.

Physical Database Research at the University of Florida

D. S. Batory, S. Y. W. Su, and S. B. Navathe

Department of Computer and Information Sciences
512 Weil Hall, University of Florida
Gainesville, Florida 32611
(904) 392-5241
(904) 392-2371

The University of Florida's Database Research and Development Center is actively engaged in a number of diverse projects in the database area. In a recently completed study, a new approach to the cost/benefit analysis of DBMSs was proposed as a way to simplify the problem of determining which database management system, out of a number of candidate systems, would best satisfy requirements and expenditure constraints for a specified business application ([SU81a]). Projects on database conversion ([SU81b]), distributed databases ([CERI81], [NICK80]), database machines ([HONG81]), and the logical and physical modeling of scientific databases ([SU81c]) are ongoing. Connected with some of these projects are studies on physical databases. These studies are briefly summarized below.

1. Decomposition of Physical Databases

If there are any major goals in the area of physical database research, one is certainly the development of practical design and tuning aids for commercial database systems. The need for such aids has long been recognized; unfortunately, their realization still is probably years away. A prime reason for the delay is the sheer difficulty in modeling complex networks of interconnected file structures. Although the knowledge for constructing physical databases has been around for many years, the theoretical underpinnings of this knowledge, fundamental to the development of design and tuning aids, are only now becoming better understood. Our work is aimed at a comprehensive and fundamental understanding of physical databases. The approach, some recent results, and current investigations are described below.

Physical databases are large networks of interconnected files. To describe them simply, the technique of physical database decomposition has been introduced. Physical databases can be decomposed into a collection of simple files and linksets. A simple file is a structure that organizes the records of a single file. Classical simple files include hash-based, indexed-sequential, B+ trees, and unordered files. A linkset is a structure that connects records of one simple file to another. Classical linksets include pointer arrays, inverted lists, multilists, and ring lists. By specifying the structure of each simple file and linkset, the structure of a physical database can be precisely defined.

In a similar manner, decomposition can be used to simplify the descriptions of transactions (i.e., complex operations) on physical databases. Transactions can be decomposed into a sequence of basic operations on simple files and linksets. Once the basic operations have been determined and expressions for estimating their execution cost have been developed, the cost of processing a transaction can then be calculated. With estimates of transaction performance, database performance can be predicted. Such predictions are useful in determining, for example, good database implementations and efficient query processing strategies.

Our approach has been quite successful in consolidating numerous disparate works on physical database design and performance. Works on batched searching, transposed files, index selection, dynamic hash-based files, generalized access path structures, and differential files have been related and extended ([BAT082a]). New techniques for determining optimal reorganization points (not heuristically chosen points) for files and techniques for predicting the performance deterioration of files undergoing record insertions and deletions have been developed ([BAT082b], [BAT081a]). General descriptive and analytic tools for modeling and understanding the behavior of transactions on physical databases have been formalized ([BAT081b]); the use of special instances of these tools were essential to recent work on query optimization and database design. And last, but not least, our approach has been successfully used in modeling of the physical structures of a commercial database system ([CASA81]). Much of this work is now being extended.

A number of studies in addition to the above are just beginning. Software modules for simple files and linksets are being written with the view of developing powerful experimental tools for implementing and testing physical database designs. This software will be integrated with modules for monitoring and predicting database performance. These monitors should provide the means by which the accuracy and validity of underlying assumptions and approximations used in current models of database performance may be tested. It is believed that such tests are constructive first steps toward the realization of design and tuning aids for commercial databases.

Finally, linear splitting is a recently discovered method for accommodating file growth in hash-based files ([SCH081]). From the generalized perspective that our modeling approach imposes, it seems clear that linear splitting can be applied to any file structure that uses overflow to accommodate file growth. Results of this investigation are forthcoming.

2. Implementation of Scientific and Statistical Databases

Scientific and statistical databases (SSDs) are collections of flat files that are characterized by large volumes of numeric data, a rela-

tively static existence (i.e., they rarely experience record update, insertions, or deletions), and hundreds of attributes per file. SSDs maintained at Lawrence Berkeley Labs, for example, contain static census data. It is quite common for census files to have several hundred attributes. However, there are some files that contain thousands of attributes (see ([McCA81])). Because SSDs are usually static, operations of SSDs are primarily retrievals. Owing to its predominantly numeric composition, data that is retrieved is often processed further by well-known statistical packages, such as SAS or SPSS. Furthermore, unlike conventional databases, users make frequent requests to access and process selected attribute data for all records in a file. It is clear from this brief description that conventional DBMSs are not well suited for SSDs. SSDs require unconventional means for storing and retrieving data.

Some of the most effective techniques for storing and processing SSDs involve data compression and file transposition. Both facilitate the reduction of the amount of data that needs to be accessed in order to process retrieval requests. We have recently developed some general techniques for storing and processing SSD files using a rather simple compression scheme. (We have yet to consider the impact of file transposition.) Unlike other data compression methods that require compressed data to be expanded before data processing can be done, our method allows file searches and most statistical processing to be done on compressed data. This alone can result in substantial savings in data processing. Moreover, the compression seems significant; in the few files we have considered so far, the compressed files had one third the volume of the original. A formal report on this work is forthcoming.

In addition to this, we also are developing a (logical) data model that is suitable for specifying SSDs containing diverse data such as n-dimensional matrices, time series data, graphs, maps, etc. An approach using abstract data types is under consideration. We also intend to investigate formal mappings between this data model and a model of physical databases ([SU81c]).

This research is supported by the U.S. Department of Energy, contract DE-AS05-81ER10977.

REFERENCES

- BAT081a Batory, D.S. "B+ Trees and Indexed Sequential Files: A Performance Comparison", ACM SIGMOD 1981, pp. 30-39.
- BAT081b Batory, D.S. "A Model of Transactions on Physical Databases", submitted to journal publication, 1981.
- BAT082a Batory, D.S. and C.C. Gotlieb, "A Unifying Model of Physical

Databases", ACM Trans. on Database Syst., 1982.

- BATO82b Batory, D.S. "Optimal File Designs and Reorganization Points", ACM Trans. on Database Syst., March 1982.
- CASA81 Casas-Raposo, I. "Analytic Modeling of Database Systems: The Design of a System 2000 Performance Predictor", M.Sc. Thesis, Dept. of Computer Science, University of Toronto, 1981.
- CERI81 Ceri, S., S.B. Navathe and G. Wiederhold "Optimal Design of Distributed Databases", to appear as Computer Science Dept. Technical Report, Stanford University, 1981.
- HONG81 Hong, Y.C. and S.Y.W. Su "Associative Hardware and Software Techniques for Integrity Control", ACM Trans. on Database Syst., September 1981, pp. 416-440.
- McCA81 McCarthy, J., et al. "The SEEDIS Project: A Summary Overview", Computer Science and Mathematics Dept., Lawrence Berkeley Labs, University of California, Berkeley, 1981.
- NAVA81 Navathe, S.B. and S. Gadil "An Approach to the Integration of Views in the Logical Database Design Process", working paper, University of Florida, 1981.
- NICK80 Nickens, D.O., T.B. Genduso and S.Y.W. Su "The Architecture and Hardware Implementation of a Prototype MICRONET", Proc. Fifth Conference on Local Computer Networks, 1980.
- SCH081 Scholl, M. "New File Organizations Based on Dynamic Hashing", ACM Trans. on Database Syst., March 1981, pp. 194-211.
- SU81a Su, S.Y.W., D.S. Batory, R. Elnicki, S.B. Navathe, A. Olagunju and J. Parkes "A DMS Cost/Benefit Decision Model", National Bureau of Standards, Contract No. NB80SBCA0449, 1981.
- SU81b Su, S.Y.W., H. Lam and D.H. Lo "Transformation of Data Traversals and Operations in Application Programs to Account for Semantic Changes of Database Systems", ACM Trans. on Database Syst., June 1981, pp. 255-294.
- SU81c Su, S.Y.W., D.S. Batory, S.B. Navathe "Logical and Physical Modeling and Design of Scientific and Statistical Databases for Energy Research", DOE Contract DE-AS05-81ER10977.

DBDSGN--A PHYSICAL DATABASE DESIGN TOOL FOR SYSTEM R

S.J. Finkelstein, M. Schkolnick, P. Tiberio
IBM San Jose Research Laboratory
408-256-7656

DBDSGN ([SCHK79], [FINK81]) is a physical database design tool developed at the IBM San Jose Research Laboratory. Given a workload for the relational database system System R ([ASTR76], [BLAS81], [CHAM81]), DBDSGN suggests which indexes should exist on each table, and which index (if any) should be clustered. A workload consists of a collection of SQL statements ([CHAM76]), with relative frequencies for them.

As an index selection tool, DBDSGN has the following distinguishing properties.

- DBDSGN accepts any SQL statement, including joins involving any number of tables, updates and queries with subqueries.
- DBDSGN obtains its cost estimates from the System R optimizer, rather than by using independent estimated costs.
- DBDSGN includes the cost of updating tuples and indexes in its evaluations.
- Design can be performed for any subset of the tables in the input statements, and specific indexes may be required in any design. However, the primary access path need not be specified in advance.

By running as an application program for System R, DBDSGN can extract information about statements and their estimated costs in different index configurations. We obtain this information by submitting statements to the System R optimizer, without actually running the statements. (The optimizer analyzes different plans for executing queries, and selects the plan with the smallest estimated cost.) Hence we need not change the tool if we change the optimizer's cost formulas. Also, we know that the optimizer must choose the access paths indicated by DBDSGN with the indicated costs. Normal operation of the optimizer, when we are not doing physical design, is not affected by DBDSGN's requirements.

In System R, each table in a statement is accessed using some single access path, which may be an index, or a scan of the segment in which the table resides. We call physical configurations in which there is (at most) one index per table atomic configurations, and we call the costs of executing statements in these configurations atomic costs. Given any configuration, there is a set of atomic configurations which are subsets of it. The cost of a query for a configuration equals the minimum of the costs of that query over all subset atomic configurations. For updates, this includes both tuple selection cost and tuple update

cost; the additional cost of updating indexes, which is sometimes significant ([SCHK81]), also must be computed.

We need not evaluate every atomic configuration for every statement. An index on AGE, for example, will usually not help process a query that does not mention AGE. For a given statement and table, some columns of the table are plausible for indexing. The costs for implausible indexes on a table must be nearly equivalent, no matter what other indexes exist. For each table in the statement, we choose one representative implausible column. Only atomic configurations in which all indexes are plausible (or the representative implausible) need be evaluated. This significantly reduces the number of configurations that we must consider, since only a few columns are plausible for each table in most statements.

Costs are obtained by creating skeleton replicas of each table, which have no tuples in them, and creating one index per replica. We simulate atomic configurations by putting statistics in the system catalogs that describe the actual tables. When we wish to simulate a particular index, we also alter the system catalogs, entering a description of that index. When we wish to simulate no index on a table, we enter very large values for the number of leaves and levels for the replica index.

Statistics for indexes may be supplied by the DBA or obtained from the system. DBDSGN can build a re-usable statistics file by creating indexes one at a time, reading their statistics, and performing a restore. A faster statistics generation tool, involving a small number of passes through each table, has also been developed. When tables do not exist the DBA must supply the statistics. The DBA may also want to alter the statistics file, to reflect expected changes in the database.

We obtain costs for configurations in which there is only one index on one table first, and apply heuristic criteria to eliminate impractical indexes. This index elimination step further reduces the number of configurations which must be submitted for cost evaluation. Only survivor indexes are considered in solution generation. Survivors are ordered on a survivor list, according to a measure of overall benefit.

Simulating configurations is itself a significant expense, so whenever we simulate a configuration, we submit all of the statements for which the configuration is plausible for cost evaluation.

Finally, in solution generation we consider a tree in which each node corresponds to a physical design. No design can have more than one clustered index per table. The root of the tree is the design in which there are no indexes. Nodes at level k have a total of k survivor indexes. Any particular set of indexes appears in only one node. (This is accomplished by requiring that new indexes added to a node appear later in the survivor list than any index already in the node.) The cost of any design solution is evaluated using the atomic costs for plausible solutions. The user can specify a maximum amount of storage for indexes, the number of solutions desired (N), and an expansion cutoff

level (EL). The tree is expanded, using breadth-first search, to level EL, and we keep only the best N solutions. These are further expanded up to EL additional levels, and this process continues until no more expansion is possible. The best N solutions found are then displayed.

DBDSGN allows computed costs to be stored for re-use in later runs. For example, we might want to run with different frequencies, or with certain indexes required in solutions.

The cost of using DBDSGN is very small compared to the cost of actually running statements on a large database many times. DBDSGN might be used at initial physical design time, when tables are added, when major updates have occurred, or when statements or their frequencies change significantly. Our experience with DBDSGN shows that it is possible to build a practical tool for index selection that recognizes all costs of all statements, and uses the system optimizer as its cost estimator.

REFERENCES

- ASTR76 Astrahan, M.V. et. al. "System R, a relational approach to database management," ACM Trans. Database Syst., 1, 2, June 1976, 97-137.
- BLAS81 Blasgen, M.W. et.al. "System R: An architectural overview," IBM Syst. J., 20, 1, 1981, 41-62.
- CHAM76 Chamberlin, D.D., et. al. "SEQUEL2: a unified approach to data definition, manipulation and control," IBM J. Res. Dev., 11, Nov. 1976, 560-575.
- CHAM81 Chamberlin, D.D., et.al. "A history and evaluation of SYSTEM R," Comm. ACM, 24, 10, Oct. 1981, 632-646.
- FINK81 Finkelstein, S.J., M. Schkolnick, and P. Tiberio "A physical database design tool for relational databases," To appear as an IBM San Jose Research Report.
- SCHK79 Schkolnick, M. and P. Tiberio "Considerations in developing a design tool for a relational DBMS," Proc. IEEE COMPSAC Conf., Chicago, Nov. 1979, 228-235.
- SCHK81 Schkolnick, M. and P. Tiberio "A note on estimating the maintenance cost in a relational database," To appear as an IBM San Jose Research Report.

PHYSICAL DATABASE RESEARCH AT THE LAWRENCE BERKELEY LABORATORY*

John McCarthy
Arie Shoshani

Building 50B -- Room 3238
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720
(415) 486-5181

This note summarizes two ongoing research efforts concerning physical database design in the Computer Science and Mathematics Department at the Lawrence Berkeley Laboratory (LBL). Section 1 describes an existing system (SEEDIS), and physical structures it employs for statistical databases. Section 2 describes a research activity for efficient compression and access of statistical data.

1. The SEEDIS Project

SEEDIS is a research and development project on Social, Economic, Environmental, and Demographic Information Systems ([McCA81]). The project began nearly ten years ago to provide quick, low cost access to large databases from the 1970 U. S. Census -- over ten billion individual data values for some 300,000 individual geographic areas.

1.1 Computer Independent Binary Compression and Storage

In order to minimize costs associated with storage, communication, and reformatting of these very large databases as hardware evolved, the SEEDIS project developed a computer independent binary format ([HEAL78]). The "SEEDIS Compressed Format" is a relatively simple type of run-length encoding for a standard "virtual machine" with two characteristics:

- + strings are stored in standard ASCII encoding
- + storage media is divided into 8-bit byte segments

The scheme currently provides for three basic types of data: alphanumeric strings, integers, and floating point numbers. Each data value is

*This work was supported by the Employment and Training Administration, U.S. Department of Labor, and the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy, under contact W-7405-ENG-48.

stored as a variable length sequence of 8-bit bytes, preceded by a single byte containing a 4-bit type code and a 4-bit byte count. All string data values are stored as ASCII character strings, integers are stored in signed binary form within an integral number of 8-bit bytes, and floating point numbers are merely two successive integers representing the exponent and mantissa. Two other key features round out the storage scheme and provide for data compression:

- + if a number has value zero, its byte count is zero and no data value is stored,

- + a fourth data "type" indicates a repeat count of the data value which follows it.

Using these conventions and its own read and write routines, SEEDIS produces binary compressed tapes for both CDC 6000 and VAX 11/780 computers. Since census data typically contains large numbers of small integers, as well as repeating strings of zeros and suppression indicators (missing data), SEEDIS compressed files typically require only about one third as much storage as the raw data.

1.2 Hierarchical Tape Storage System

The tape-based General Storage System (GSS) developed by LBL Computer Center staff ([GOK81]), has been an effective complement to the SEEDIS Compressed Format. Each GSS tape has a hierarchical directory structure which resembles that of Unix ([RITC74]). Users can access individual files or sets of files using unix-like path name notation such as:

```
/70census/tract/calif/alameda/*
```

where * signifies any file belonging to this directory.

Thus terminal nodes are accessible much as they would be on a random access device -- even though the underlying medium is sequential. Since census and many other types of geographic data have a natural hierarchical structure, representation on GSS is straightforward. GSS tapes are accessible via a tape robot, so retrieval time is relatively fast (usually minutes), and SEEDIS users need not even know that data are coming from tape.

1.3 Storage for Different Levels of Analysis

SEEDIS contains data for some sixty different types of geographic levels, some of which overlap considerably. In order to permit analysis across different levels without redundant storage for multiple levels, SEEDIS contains cross-level reference files and multiple index files for different levels which reference the same underlying physical data file. For example, if a user references 1970 census data of the "1980 County" level, access is made via a separate 1980 county index to the 1970 data, but the county data records are the same save for counties whose boundar-

ies changed between 1970 and 1980; for those few cases, additional records are created by adding and subtracting smaller areas such as minor civil divisions.

1.4 Current SEEDIS Development Areas

SEEDIS project staff are currently working on several physical database problems:

- + a non-procedural data definition language for specifying input and output formats, internal physical storage organization, and transformations between different internal storage formats (e.g., transposition or partial transposition);

- + development of caching mechanisms so that data and data descriptions can automatically migrate from GSS tapes to disk files on a distributed network of VAX computers;

- + implementation of calling sequences to provide transparent low-level access to different compression and storage schemes, such as those outlined below.

For further information, contact Fred Gey, Bob Healey, Harvard Holmes or John McCarthy.

2. Compression Techniques For Statistical Databases

We describe here work in progress on techniques for compressing statistical databases (SDBs). These techniques exploit certain characteristics of SDBs, in order to gain a high degree of compression while achieving fast (logarithmic) access time.

There are two characteristics of statistical databases that we exploit. The first is the repetition of a small set of data values, such as zero or a missing data indicator. These values, which we refer to as "constants", may represent a large portion of the database. A standard way of removing the repeating values from the data stream is by associating a count with each sequence of repeating values. When these counts are left in the data stream (a technique known as "run length encoding"), access to a particular data value involves a sequential decoding of the data stream, a process which requires a linear search time.

Our approach removes all counts from the data stream and stores them in a header. The counts are modified to be cumulative, thus allowing the header to be searched in logarithmic time. The header is used to form the base level of a B-tree index into the database. This further improves the access time by increasing the radix of the logarithmic search. In addition, the technique can be extended to support the compression of multiple constant values.

The second characteristic of statistical databases is their skewed distribution of data values. Often the range of integer values for certain attributes is large, requiring several bytes to represent them (typically 4 bytes each for integers). However, the distribution of values may be skewed toward the lower end, making smaller values more likely. Compressing data values to their minimum byte length will reduce the storage requirements considerably. The header technique used for compressing constants can be applied to multiple-sized data values, by modifying the count entries to include physical byte counts. Thus, a single entry in the header can represent a substring of data values of the same size, as well as a series of constants.

There are several applications for which it may be advantageous to use a more simplified version of the compression scheme. In particular, the version specifically designed for the case of single constant, single size data achieves a header storage savings of 2.5 times that of the general scheme. The general version of the compression scheme and other interesting special cases are discussed more fully in ([EGGE80], [EGGE81]).

For further information, contact Susan Eggers or Arie Shoshani.

REFERENCES

- EGGE80 Eggers, S. J. and A. Shoshani "Efficient Access of Compressed Data," Proceedings of the International Conference on Very Large Databases, 6, Montreal, 1980, pp. 205-211.
- EGGE81 Eggers, S. J., F. Olken and A. Shoshani "A Compression Technique for Large Statistical Databases," Proceedings of the International Conference on Very Large Databases, 7, Cannes, 1981, pp. 424-434.
- GOK81 Gok, D. "Gettape/Stotape System," LBL Computer Center Publication (machine-readable), WRITEUP subset GSS (revised June, 1981).
- HEAL81 Healey, R. "BYTER and DBYTE," LBL internal design document, May, 1978.
- McCA81 McCarthy, J., et. al. "The SEEDIS Project: A Summary Overview," LBL PUB 424, September, 1981.
- RITC74 Ritchie, D. M. and K. Thompson "The UNIX Time-Sharing System," 17 Communications of the ACM 7 (July 1974), pp. 365-375.

S. Bing Yao and Alan R. Hevner
Database Systems Research Laboratory
University of Maryland
College Park, MD 20742
301-454-6258

1. Introduction

In this short paper we describe on-going projects related to physical database design at the Database Systems Research Laboratory, University of Maryland*. The research projects are divided into three areas: storage structure design and optimization, query optimization, and database design tools. In the following sections, we briefly summarize the results obtained thus far in each area and state the future directions for each project.

2. Design of Storage Structures

In previous research, we presented methods to optimally choose and combine structures to form an index ([YAO75b]) and algorithms to design reduced indices using interpolation. In another model, we proposed a new type of index structure that handles overflow using a combination of splitting and chaining methods ([YAO77a]). B-trees and ISAM were shown to be special cases of this model. Index organizations enhanced by parallel processing and batching were reported in ([HWAN77]). In order to minimize access conflicts and maximize concurrency, we designed a locking procedure which requires the minimal amount of locking when searching and updating an index structure ([LEHM81]).

Recent database system implementations (e.g. System R and INGRES) show that it is sometimes desirable to index several attributes at the same time. We analyzed the performance of the classic doubly-chained-tree structure ([KASH77]). A superior structure based on multi-dimensional clustering was proposed in ([LIOU77]). In order to model the many designs in a common framework and develop uniform design methods, a model for multi-attribute indices was presented in ([YAO76c]).

Our present research in index design combines previously developed structures into an efficient file organization. Since most index structures (e.g., B-trees) suffer from the lack of control in physical storage

*Research projects are supported by grants from NSF and DEC.

allocation and structuring, the new file structure is designed to optimize disk access and allocation. The basic design is a VSAM-like structure enhanced with secondary indices, interpolation search, and concurrent access paths. This file structure will be implemented on a PDP 11/44 and will be used as a basic component of an experimental database machine.

2.2 Database Reorganization

The performance of a physical database system will degenerate as a result of insertion, deletion and restructuring operations. In order to restore the efficiency of the system, periodic reorganizations are performed. This is a costly operation and makes the database unavailable during reorganization. While it is possible to determine optimal reorganization points for minimal overall cost ([YAO76a]), it would be better if database structures requiring little or no reorganizations were available. Structures such as B-trees and VSAM appear to have this property, but a closer inspection shows that they in fact perform dynamic reorganization (after each insertion and deletion). In a multi-user system, these reorganizations must be performed concurrently. This necessitates the use of elaborate locking protocols ([BAY77]). Our concurrent B-tree update algorithms ([LEHM81]) require only a minimal number of shared locks for each update while they allow retrieval operations to proceed without any locking. This makes it possible to perform concurrent B-tree reorganization when the database is continuously operational.

We are working to extend the methods used in this approach and apply them to other database structures. The result will be a collection of concurrent reorganization algorithms. Some of the methods will be implemented and tested in the experimental database machine mentioned above.

2.3 Storage Structure Evaluation

Cost equations developed for storage structures have been very complicated (e.g., [YAO77a], [YAO79]). Different assumptions and different levels of detail also make them difficult to be used in a comprehensive design evaluation tool. Our present approach to design evaluation is to consider only a subset of important storage structures. Since each structure will be implemented in our experimental system, the cost equations can be more accurately tuned. The objective is to develop a design subsystem which will automatically select storage structures and dynamically reconfigure the system for optimal performance.

3. Query Optimization

A number of possible processing strategies are available for queries on multiple files. Finding the optimal strategy is difficult since the performance of a strategy is influenced by many parameters. While an

exhaustive search technique ([SEL79]) is too costly for small computers, a heuristic approach ([WONG76]) that ignores certain data characteristics may miss good strategies. In ([YAO78b]), algorithms for two file queries are systematically classified and compared using a query model. In ([YAO79]), algorithms for more than two files are surveyed and classified as special cases of a general model. This systematic approach has discovered new algorithms that have good optimization potential. It also makes possible the analysis and comparison of various processing strategies for multifile queries. The objective of our further research in this area is to develop an optimization algorithm that integrates effective cost analysis methods into an efficient heuristic approach.

Query optimization in a distributed database system is an even more challenging problem. A good processing algorithm should minimize both the local data access time and the amount of network data transmission. We have developed optimal processing algorithms that minimize the total data transmission time and the query response time for a special class of queries ([HEV78]), ([HEV79]), ([HEV80]). A collection of optimization algorithms for star networks are presented in ([KERS82]). We are currently working on an optimization algorithm for network architectures which consist of multiple processors connected by a local bus. The results of this research will be applied in the experimental database machine project.

4. Database Design Tools

An integrated database design approach is shown to consist of several phases: requirement analysis, view modeling, view integration and physical design ([YAC78a]). In order to provide a basis for logical database design, we developed the Functional Data Model and a Transaction Specification Language ([HOUS79]). This model is extremely simple and easy to understand; yet it is shown to be more general than most of the semantic data models ([WADD79]).

A Database Design Tool (DDT) that is based on the Functional Data Model is presently being developed. The system consists of three main parts:

- A. Logical Database Design System. The system contains several components:
 - 1) A design requirement database which contains the data model requirements represented in the Functional Data Model and the processing requirements represented in the Transaction Specification Language.
 - 2) An interactive design integration component. A version of the third normal form synthesis algorithm is implemented. The algorithm is augmented by heuristic rules and reduces

the amount of interaction required from the designer. The result is a machine and DBMS independent logical database design.

B. Physical Database Design System. The components of this system are:

- 1) A design evaluation component. The intermediate design results will be evaluated on three levels:
 - a) Logical design level (output of A above).
 - b) Access path level - the estimation of access path length after data grouping (into logical records or segments) is performed.
 - c) Physical design level - the estimation of the number of page accesses after considering indices, clustering, and other access path implementations permitted by a particular database system.
- 2) An interactive schema designer. Some output of the evaluation system will be used to select physical database structures. Heuristic methods similar to that used in ([SCHK79]) will be employed.

C. Distributed Database Design System. In ([HEV80]) we proposed a design methodology to be used in an interactive design tool for distributed systems. This design tool emphasized inter-nodal performance aspects such as data transmission times and queueing delays. Analytic equations estimated the performance of the system design and allowed the user to interactively make design changes.

We have presently completed a preliminary implementation of the logical design tool (A above). This system (written in PASCAL on CDC 6500) is presently being moved to UNIX and enhanced. Research on the other subsystems is continuing. They will be developed and implemented on the PDP 11/44 using UNIX.

REFERENCES

- BAY77 Bayer, R. and M. Schkolnick "Concurrency of Operations on B-Trees," Acta Informatica, 9, 1977.
- HEV78 Hevner, A. and S. B. Yao "Query Processing on a Distributed Database," Third Berkeley Workshop on Distributed Data Management and Computer Networks, August 29-31, 1978.
- HEV79 Hevner, A. and S. B. Yao "Query Processing in Distributed Database Systems," IEEE Trans. on Software Engineering SE-5, No. 3, May, 1979.
- HEV80 Hevner, A. and G. M. Schneider "An Integrated Design System for Distributed Database Networks," Proc. IEEE Fall COMPCON

80, Washington, D.C., September 1980.

- HOUS79 Housel, B., V. Waddle and S. B. Yao "The Functional Dependency Model for Data Base Design," Proc. Fifth International Conference on Very Large Databases, Rio de Janeiro, Brazil, October 1979.
- HWAN75 Hwang, K. and S. B. Yao "Parallel Processing of Multiway Search Trees," Proc. Computer Graphics, Pattern Recognition and Data Structures, 1975, pp. 170-176.
- HWAN77 Hwang, K. and S. B. Yao "Optimal Batch Searching in Multiprocessor Computer Systems," Journal of ACM 24, 3, July, 1977, pp. 441-454.
- KASH77 Kashyap, R. I., S. Subas and S. B. Yao "Analysis of the Multiple Attribute Tree Database Organization," IEEE Trans. on Software Engineering SE-3, No. 6, November, 1977, pp. 457.
- KERS82 Kerschberg, L., P. D. Ting and S. B. Yao "Query Optimization in Star Computer Networks," Technical Report DB-80-04, Computer Science, Purdue University, (to appear in ACM TODS).
- LEHM81 Lehman, P. L. and S. B. Yao "Efficient Locking for Concurrent Operations on B-trees," ACM Trans. on Database Syst. 6,4, December, 1981.
- LIOU77 Liou, J. H. and S. B. Yao "Multi-Dimensional Clustering for Database Organizations," Info. Syst., 2, 4, 1977, pp. 187-198.
- SAC81 Sacco, G. and S. B. Yao "Query Optimization in Distributed Database Systems," Working Paper MS/S #81-029, College of Business and Management, University of Maryland, 1981.
- SEL79 Selinger, P. G., et al. "Access Path Selection in a Relational Database Management System," Proc. ACM SIGMOD 1978.
- SCHK79 Schkolnick, M. and P. Tiberio "Consideration in Developing a Design Tool for a Relational DBMS," Proc. of COMPSAC 79, 1979.
- WADD79 Waddle, V., Housel B. and S. B. Yao "View Modeling and Integration Using the Functional Dependence Model," Proc. COMPSAC 79, November, 1979, pp. 236-244.
- WONG76 Wong, E. and K. Youseffi "Decomposition - A Strategy for Query Processing," ACM Trans. Database Syst., 1, 3, September, 1976.
- YA075a Yao, S. B. and A. Merten "Selection of File Organizations Through Analytic Modeling," Proc. Conf. on Very Large Databases, September, 1975, pp. 255-267.
- YA075b Yao, S. B. "Tree Structure Construction Using Key Densities,"

Proc. ACM 75 National Conf., October, 1975, pp. 337-340.

- YA076a Yao, S. B., K. S. Das and T. J. Teorey "A Dynamic Database Reorganization Algorithm," ACM Trans. on Database Syst., 1,2, June, 1976, pp. 159-174.
- YA076b Yao, S. B. "Modeling and Performance Evaluation of Physical Database Structures," Proc. ACM 76 Annual Conference, October, 1976, pp. 303-309.
- YA076c Yao, S. B. "A Model for Combined Attribute Index Organizations," Proc. Fifth Texas Conference on Computing Systems, October, 1976, pp. 127-130.
- YA077a Yao, S. B. "An Attribute-Based Model for Database Access Cost Analysis," ACM Trans. on Database Syst., 2, 1, March, 1977, pp. 45-67.
- YA077b Yao, S. B. "Approximating Block Accesses in Database Organizations," Comm. ACM, 20, 4, April, 1977, pp. 260-261.
- YA078a Yao, S. B., S. B. Navathe and J. L. Weldon "An Integrated Approach to Logical Database Design," NYU Symposium on Database Design, May 18-19, 1978, pp. 1-14.
- YA078b Yao, S. B. and D. DeJong "Evaluation of Database Access Paths," ACM SIGMOD, June 1-3, 1978.
- YA079 Yao, S. B. "Optimization of Query Evaluation Algorithms," ACM Trans. on Database Syst., 4, 2, June, 1979.
- YA082 Yao, S. B., V. Waddle and B. Housel "View Modeling and Integration Using the Conceptual Data Model," (to appear in IEEE Trans. on Software Engineering.)

SYSTEMS AND TECHNIQUES FOR RESEARCH IN PHYSICAL DATABASE DESIGN
AT THE UNIVERSITY OF MICHIGAN

Toby J. Teorey
Houtan Aghili
Richard Cobb
James P. Fry
Dennis G. Severance
Michael E. Wilens

Information Systems Research Group
Graduate School of Business Administration
The University of Michigan
Ann Arbor, Michigan 48109
313-763-1100
313-763-3580

Physical database design research continues to evolve from simple algorithms for designing record structures and access methods to the more complex interactive systems for integrating the physical structures of large centralized and distributed database systems. A number of recently completed and on-going projects at the Information Systems Research Group of the University of Michigan are concerned with the more advanced techniques of physical design. These efforts include work on a database designer's workbench, a model for physical design called frame memory, and differential files.

1. INFORMATION SYSTEMS RESEARCH GROUP

The Information Systems Research Group (ISRG) at the University of Michigan is a formal association of faculty and students, drawn from the Graduate School of Business Administration, College of Engineering, and Department of Computer and Communication Sciences; and whose goal is to develop new knowledge and extend existing principles for the development of executive information systems. The purpose of these systems is to support the information handling and decision making activities of managers through the use of computers. Research is conducted on all phases of executive information systems including management requirements, system analysis, system design, and implementation as well as operational aspects of internal control and auditing.

ISRG, formerly known as the Database Systems Research Group, began

in 1972 as an outgrowth of the Information System Design and Optimization Systems (ISDOS) Project at the University of Michigan. Beginning with a grant to study the theoretical foundations of stored-data definition, ISRG research evolved to the development of a prototype Stored-Data Definition Language Analyzer and generalized software for data translation. In 1974 the scope of research expanded to develop semi-automatic tools for database design. Several operational software packages have since been completed to evaluate database designs for existing large-scale systems, ranging from index sequential through CODASYL-like network systems. In 1978 the research was expanded to include distributed data systems. Specific research topics have included modeling database systems, distributed database design, and implementation of a distributed system on mini-computers ([AGHI80]).

Sections 2 through 4 describe some of the current research projects specifically related to physical database design.

2. THE DATABASE DESIGNER'S WORKBENCH

Researchers: Toby Teorey, James Fry, Rick Cobb

The Database Designer's Workbench (or "Workbench" for short) is a graphics-oriented decision-support system which assists analysts with the design of a computer database, from the initial specification of the system's requirements through its final physical structure. The Workbench provides designers with a wide variety of design aids, or "tools", which can be used for developing design alternatives and then evaluating their performance. These tools are part of a homogeneous, graphically-oriented environment that allows a designer to use abstract data representations, store incomplete designs, progress smoothly from one design phase to the next, and iterate over previous design stages. The objective of the Workbench is twofold. For the database design practitioner, the Workbench will hopefully improve productivity and the quality of design by making it simpler to explore alternative designs. For the researcher, the Workbench is capable of monitoring an individual designer's use, thus leading to improvements in the database design process.

The Workbench supports the database design methodology described by Teorey and Fry ([TEOR80], [TEOR82]). As shown Figure 1, the methodology is composed of six separate steps, each of which has its own input requirements, design and evaluation techniques, and output. The basic steps for centralized databases are 1-3 and 6; steps 4 and 5 are specific to distributed databases.

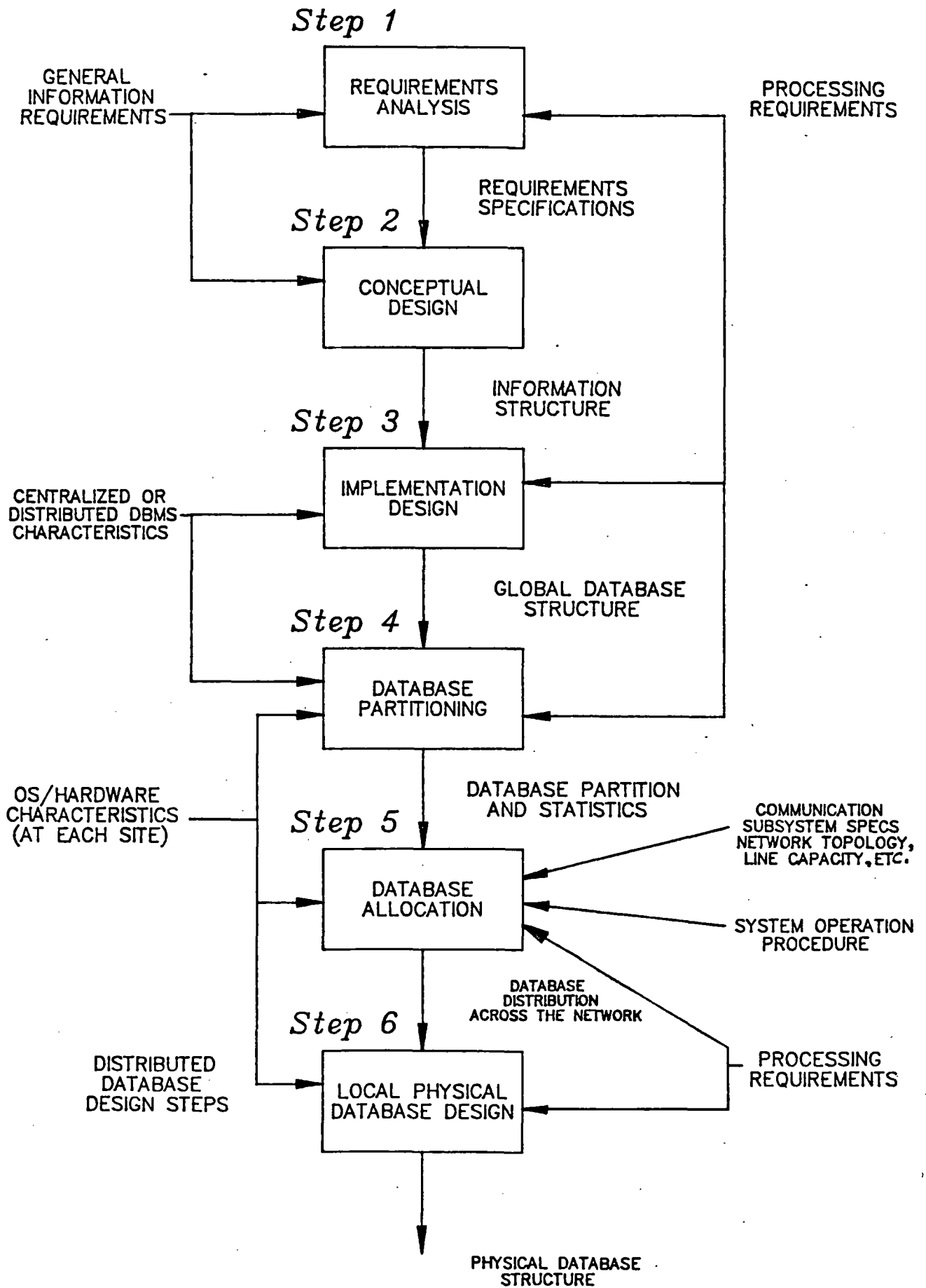


FIGURE 1.. DATABASE DESIGN STEPS

The database design steps are not completely independent of each other. A single pass through them all will not result in an optimal design because there are many interdependencies, some of which result from the biases of the designs themselves and which are inherent in design process. Hence, iteration is a necessary part of the design process. Iteration within a major step also may be necessary to find an optimal solution. Iteration across major steps is required when certain requirements are not met by any of the alternative designs at a particular step, or if new requirements are added that require redesign. As an example, if it is found that candidate physical database structures cannot meet response time criteria, new logical database structures, or even new conceptual information structures may have to be considered. The Workbench simplifies the process of redesign and evaluation.

The Workbench provides several advantages over both manual methods of database design and presently existing automated tools. It enables researchers and database administrators to design, experiment with, and use new tools of their own. It encourages this by relieving the designers of some of the details involved in developing user interfaces, input/output of design parameters, and storage specifications. The graphic interface allows the designers to see the various structures they are considering for their database throughout the entire design process. Since database structures are easily modified graphically, designers are encouraged to produce and consider more candidate structures. Also, when the designer is progressing from one design phase to the next and is faced with a set of possible structures to choose from, the system's graphical nature can significantly aid in the heuristics of selection.

A demonstration Workbench has been implemented (in FORTRAN IV) on the Michigan Terminal System (MTS) using an APPLE II microcomputer as its user interface. This demonstration model has been useful both for demonstrating the Workbench concept and for research on tool building, but it is not robust enough for all the analytical tools.

The prototype Workbench currently being implemented on Multics also uses an APPLE II as its graphics input/output device and command level processor; Multics contains the algorithms, the tools, and the internal diagram handler ([BELE81], [COBB81]). The command level processing portion of the software is written in PASCAL, while the algorithms, tools, and diagram handler have been implemented in MACLISP. Normal "command line" interaction is possible through any Multics terminal.

Different tools have been proposed and are currently under development for both the demonstration and the Multics prototype Workbenches. These include an access method evaluator, a secondary index selector, and a clustering methodology for hierarchical and network databases ([SCHK78], [TEOR82]).

3. FRAME MEMORY SYSTEM

Researchers: Dennis Severance, Mike Wilens

The process of physical database design is driven primarily by the need to satisfy user retrieval requirements with data access paths of reasonable speed. Procedures previously developed to assist in this process focus upon modeling a wide range of fundamentally different data access paths and selecting an efficient combination of these paths based upon estimated storage, retrieval, and maintenance costs. Much of the complexity and computational overhead of these models is associated with access path maintenance which occurs as database records are added, deleted, and modified. Difficulties in analyzing database maintenance phenomena (e.g., record shifting, overflow chaining, garbage collection, etc.) have caused a number of researchers to ignore database maintenance costs in initial versions of their design systems.

A generalized model of secondary memory management, called frame memory, has been developed ([MARC81]). The concept evolved from the modeling work as a means of simplifying model cost equations and speeding design evaluation by isolating and consolidating the effects of data maintenance operations at a level below that of the data access paths. Frame memory provides record storage retrieval and update services to access path routines. The speeds and costs of these services can be estimated accurately from frame design parameters and volume and volatility characteristics of the records which are stored. Since these estimates incorporate expected effects of maintenance phenomena, these effects need not be explicitly modeled by performance equations for the data access paths.

Frame implementations are described parametrically. As a result, a variety of alternatives can be generated quickly and evaluated automatically by a design system to select an efficient implementation for a given problem. Although frame memories are not in use commercially, frame parameters do translate approximately into traditional database design variables, such as record blocking and file loading factors. And while frame memory was devised initially as an abstraction to simplify a set of database design procedures, the existence of these procedures now makes the frame memory architecture a natural OS/DBMS interface for future systems.

A design methodology based upon frame memory exists to aid system analysts in the design of physical databases ([CARL80]). It was developed at the University of Minnesota and is overviewed briefly here. The design approach is built upon an analytic model composed of (1) parametric descriptions for components of a generalized database organization, (2) cost equations which evaluate a proposed database design, (3) an analyst interface which accepts arbitrary database designs for evaluation, and

(4) search procedures which automatically generate and compare thousands of alternative organizations.

Ideally, the system will determine and output an optimal combination of (1) frame implementations, (2) record segmentation, and (3) file structure and search mechanism parameters. The design of each of these components is affected by characteristics of the others so that in order to guarantee the selection of a globally optimal design, all combinations of model parameters must be evaluated. Unfortunately, the number of possibilities is enormous and no computationally tractable method of searching the entire solution space is known. As a result, a heuristic based on an optimization procedure was used. While no heuristic design procedure can produce an optimal solution in every case, the designs generated by this procedure are structurally simple and intuitively reasonable. Their performance typically compares favorably with designs constructed by experienced analysts, and since the model's designs are selected in minutes rather than days, at a minimum they provide a starting point for design elaboration and a benchmark against which to compare the performance of proposed alternatives. Alternatives, of course, can be submitted to the system for rapid evaluation.

Ongoing research at ISRG is concerned with investigating novel strategies for improving frame memory performance, improving the existing design procedure, and automating the database implementation process. Currently, the parametric model of frame memory is used for evaluation only. That is, given a problem statement and a relatively small set of heuristically selected frame implementations, alternative database designs are evaluated for each implementation. The number of possible frame implementations is extremely large and a procedure to formalize initial pruning of the solution space is under development. In addition, design problems studied to date have assumed a homogeneous secondary memory environment consisting of only moving head disks. Nevertheless, our model's characterization of secondary memory is sufficiently general to permit consideration of alternative memory technologies without alteration of the performance-measure interface to our design procedures. We plan, for example, to investigate the impact of a bubble memory/disk storage/mass storage memory-hierarchy upon existing database structuring practices. The effect of a frame memory environment upon database backup, checkpoint/restart, and concurrency management procedures is also being studied.

Operational software exists for the frame memory performance model, the automatic file design optimizer, and a prototype frame memory system. The software is written in FORTRAN IV and runs on both IBM 370 and CDC 7600 architectures.

4. DIFFERENTIAL FILE DESIGN

Researchers: Dennis Severance, Houtan Aghili

There are two basic causes of data loss in online systems: (1) partial completion of update operations caused by the program or system failures which render parts of the database inaccurate or inaccessible, and (2) physical destruction of storage media which renders all or part of a database unreadable. For large databases with moderate or extensive update activity, differential files offer an effective strategy for rapid backup and recovery.

A differential file contains all new and modified records which would otherwise have altered another file, called the main file. Since the main file is never changed it can always be recovered quickly from its dump in the event of a loss. Transaction reprocessing is required only in the event of damage to the differential file. Since this file is usually small, it can be backed up quickly and frequently to minimize the reprocessing time. It can also be duplicated at reasonable cost as insurance against physical damage to one of the copies.

While differential files offer a number of other operational advantages ([SEVE76]), our current research at ISRG has focused exclusively upon their value in speeding backup and recovery operations for online databases. Specifically, these operations have been analyzed to establish the frequencies with which backups and mergings should occur for a given main file and differential file. An analytic cost model for this purpose has been developed; it is used to generate a series of tables which enable a designer to quickly determine a near-optimal differential file architecture for a typical operating environment ([AGHI81]).

5. SUMMARY AND FUTURE RESEARCH

Active research is continuing with the Database Designer's Workbench and its associated tools for both conceptual and physical database design. Performance evaluation of distributed database systems will continue as more analytic models are designed, implemented, and tested with performance data from actual system implementations.

In addition to ISRG, other research groups are also active in database research at Michigan. Under the auspices of the Systems Engineering Laboratory in Electrical and Computer Engineering, Irani and others are developing models for data allocation and query optimization in distributed database systems ([KHAB79]). Under the direction of Daniel Teichroew, performance monitoring of database management systems is under experimentation

in the ISDOS project (Industrial and Operations Engineering). Cooperative efforts are maintained among these groups in many areas of database systems research.

REFERENCES

- AGHI80 Aghili, H., D. DeSmith and M. Grocock "A Distributed Database Management System for the IBM Series/1: General Design and Prototype Design," Working Paper 80 DS 8.3, Database Systems Research Group, Graduate School of Business Administration, The University of Michigan, Ann Arbor, 1980.
- AGHI81 Aghili, H. "Differential File Application and Analysis," Ph.D. Thesis, CICE Program, The University of Michigan, Ann Arbor, 1981.
- BELE81 Belew, R. K., R. Cobb, J. P. Fry, and T. J. Teorey "The Database Designer's Workbench: An Overview," Working Paper 81 DE 1.10, Information Systems Research Group, Graduate School of Business Administration, The University of Michigan, Ann Arbor, October 1981.
- CARL80 Carlis, J. V. "An Investigation into the Modeling and Design of Large, Logically Complex, Multi-User Databases," Ph.D. Thesis, University of Minnesota, 1980.
- COBB81 Cobb, R., A. Garcia, L. Hourvitz and D. Meredith "Database Designer's Workbench: Design Specifications," Working Paper 81 DE 1.11, Information Systems Research Group, Graduate School of Business Administration, The University of Michigan, Ann Arbor, October 1981.
- KHAB79 Khabbaz, N. G. "A Combined Communication Network Design and File Allocation for Distribution Databases," Ph.D. Thesis, CICE Program, The University of Michigan, Ann Arbor, 1979.
- MARC81 March, S. T., D. G. Severance and M. E. Wilens "Frame Memory: A Storage Architecture to Support Rapid Design and Implementation of Efficient Databases," ACM Trans. Database Syst. 6,3 (September 1981, pp. 441-463.
- SCHK78 Schkolnick, M. "Physical Database Design Techniques," Proc.

NYU Symposium on Database Design, New York University, N.Y.,
May 18-19, pp. 99-109.

SEVE76 Severance, D. G. and G. M. Lohman "Differential Files: Their
Application to the Maintenance of Large Databases," ACM Trans.
Database Syst. 1,3 (September 1976), pp. 256-267.

TEOR80 Teorey, T. J. and J. P. Fry "The Logical Record Access Approach
to Database Design," ACM Comp. Surveys 12,2 (June 1980), pp.
179-211; 12,4 (December 1980), pp. 465.

TEOR82 Teorey, T. J. and J. P. Fry Design of Database Structures, Pren-
tice-Hall, Englewood Cliffs, N.J., 1982.

AN OVERVIEW OF PHYSICAL DATABASE DESIGN
RESEARCH AT THE UNIVERSITY OF MINNESOTA*

S. T. March
Department of Management Sciences
612-373-4363

J. V. Carlis
Computer Science Department
612-376-4592

University of Minnesota
Minneapolis, Minnesota 55455

This paper overviews the ongoing research efforts at the University of Minnesota in physical database design. These efforts are being conducted within the framework of A Database Design Methodology (ADDM) which contains: (1) a multiple level descriptive model for expressing logical database design problems and their physical solutions and (2) a man/machine database design system (DBDS) which inputs a problem description expressed in the logical level model and produces design solutions expressed in the physical level model.

1. INTRODUCTION

The objective of physical database design is to produce data storage structures and accessing mechanisms which effectively and efficiently support the information requirements of some community of users within an organization ([LUM78]). In order to meet this objective, the information requirements must first be unambiguously specified for use in the design process. The design itself must also be unambiguously specified for implementation on the target computer system.

ADDM (A Database Design Methodology) supports a database designer in the task of physical database design. It provides: (1) a multiple level descriptive model for expressing logical database design problems and physical database design solutions and (2) a man/machine database design system (DBDS) for producing efficient database designs. A design produced via the methodology consists of a set of records (types) defining the database schema and sets of algorithms and structures (i.e., file organizations) to store, access and maintain the data. This specification is not dependent upon a particular hardware configuration or database management system (DBMS) but is generic, using commonly available file structures and searching techniques. In order to use the methodology for a particular design problem, however, the types of design alternatives considered by DBDS may be restricted to conform to existing hardware and/or DBMS software configurations.

*This work was supported in part by the David W. Taylor Naval Ship Research and Development Center under Research Contracts N00167-79-C-0140, and N00167-80-C-0061 and N00167-81-C-0104.

2. THE MODELING CONTEXT

The database design methodology is based on a database model (see [CARL81a]) consisting of two levels: logical and physical. The logical level is used to formally express the information requirements of the user community (i.e., the problem) independent of the manner in which they will physically be met. It provides a data model for the data required for physical database design. It thereby guides the collection of user requirements and helps to insure that these requirements are appropriately described. The physical level is used to formally express database designs (i.e., solutions) for implementation. It delimits the set of possible database designs considered by the methodology and thus permits the specification of a "solution space" for a particular design problem.

The logical level of the descriptive model has been adapted from Senko's "infological model" in DIAM II [SENK75]. It contains a simple yet semantically rich conceptual data model (CDM) composed of entities, attributes, relationships, and identifiers. This model is used to describe the logical structure of the database. Logical database volume is described by entity cardinality and attribute cardinality, length and vocabulary size. User retrieval activities are specified as a set of retrieval statements or queries each of which focuses on some of number of entities (termed contexts). At each entity (context) the retrieval is characterized by the entity instances selected, the descriptors projected (attribute and/or relationship descriptors), the output ordering and the relationship which is used to forward the context to the next entity in the retrieval (if any). Update activities are specified by entity insertion and deletion frequencies and attribute and relationship modification frequencies.

The physical level of the descriptive model views a database as a set of interconnected file organizations where a file organization is a set of algorithms and structures used to store, retrieve and maintain a subset of the database (a dataset) residing permanently in secondary memory. A complete physical database design consists of definitions for its datasets (schemas and instances) and a file organization design for each dataset defined.

The generalized model of a file organization (see [MARC78a], [MARC78b]) has three modular components: data records (see [EISN76], [MAXW73], [MARC77], [MARC81b]), access paths (see [SEVE77], [DUHN78]), and maintenance mechanisms (see [MARC81a]), each of which is a parametric model capable of describing a wide range of implementation alternatives which are practical for that component. A complete file organization design is defined by a collection of parameter sets -- one for each modular component.

3. THE DATABASE DESIGN SYSTEM (DBDS)

The database design system (DBDS) of ADDM takes advantage of the

computational power of modern computer systems to augment and support the intuition and skill of a database designer. The DBDS consists of a set of software procedures (approximately 10,000 lines of executable FORTRAN code ([CARL80b], [CARL80c]) and a user interface through which the designer controls the execution of the system. Input to the software is a statement of the information requirements of the user community expressed in the logical level model; output is a complete database design expressed in the physical level model and as a by-product of the design process, a performance estimate for that design.

Figure 1 is a data flow diagram ([DEMA78]) which shows the major activities of the DBDS and delimits the role of the designer and the role of the software in the system. The designer formulates the problem, exercises judgment and constrains the solution space as lead by intuition, experience, and intermediate results produced by the software. The software suggests generic solution alternatives, optimally solves various design subproblems and produces an efficient database design. Optionally, the software may be used to evaluate the performance of specific designs proposed by the designer.

The system includes a number of design heuristics as well as optimization and evaluation algorithms which are incorporated into the four major modules (FORM, CONVERT, DESIGN, and SELECT) shown in the software side of Figure 1. The operation of the design system is briefly described below:

1. FORM potentially useful database schemas. Each schema defines a set of datasets. Optionally the designer may add to or delete from the set of datasets heuristically produced by the software or even completely specify a single set of datasets (see [CARL81d]).
2. CONVERT processing requirements expressed in the logical level model to accessing patterns on the sets of datasets formed in (1). This effectively yields a set of file organization design problems.
3. DESIGN an efficient file organization for each problem from (2). Again, the designer may optionally constrain the solution space or specify partial solutions or even complete solutions for evaluation only (see [MARC78a], [MARC78b]).
4. SELECT a set of file organizations which efficiently meet the user information requirements. This module automatically selects the most efficient set of file organizations from among those produced in (3).

Since there are many qualitative factors not considered by the software which may have considerable impact on overall database performance, the designer may subjectively evaluate the set of designs produced by the system and/or perform sensitivity analysis by varying critical de-

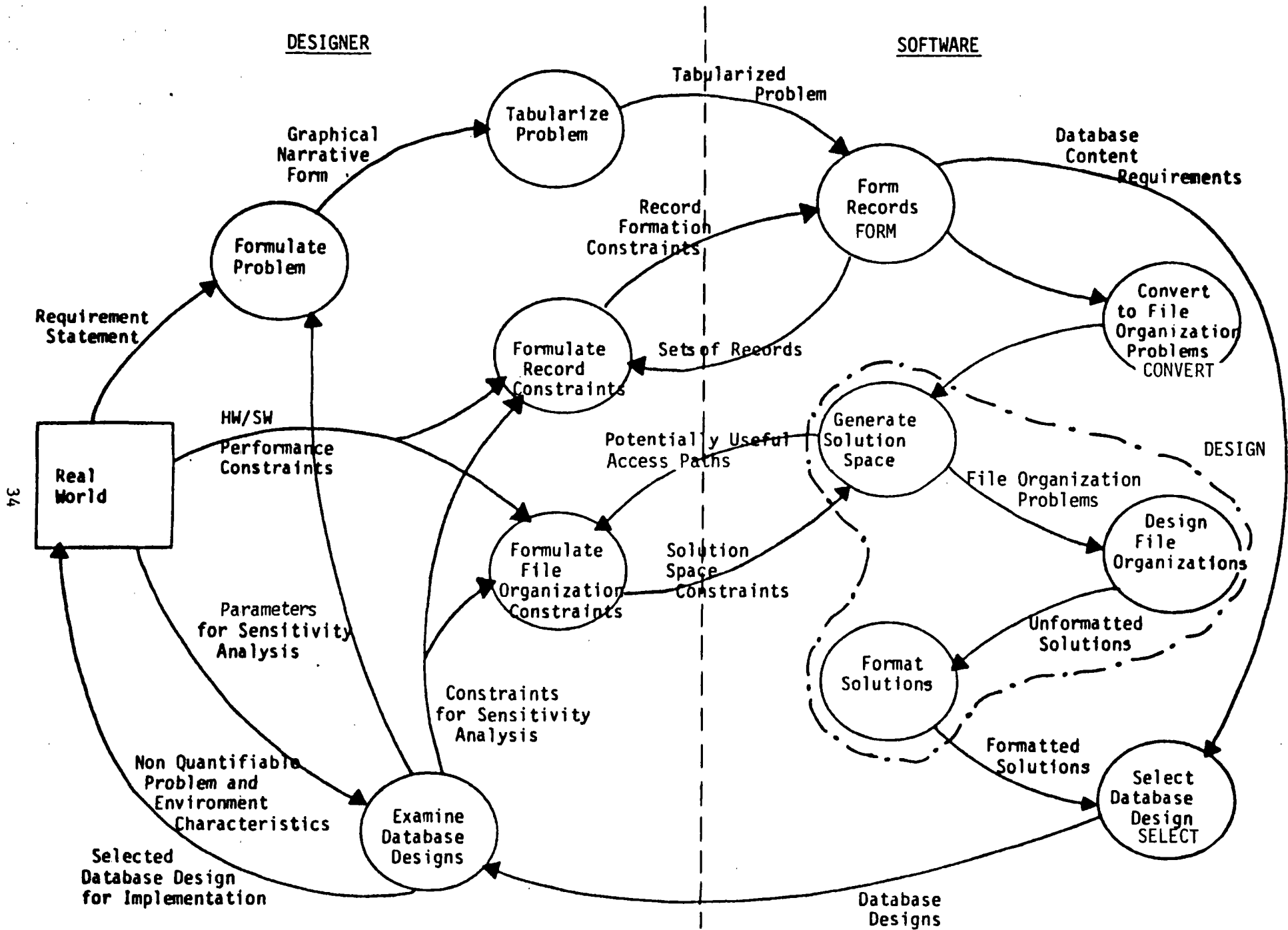


FIGURE 1. DATA FLOW DIAGRAM OF THE ACTIVITIES OF THE DBDS

sign factors and iteratively executing the above modules prior to selecting a design for implementation.

4. CURRENT RESEARCH ACTIVITIES

This section briefly describes some current research efforts within the framework of ADDM aimed at improving the methodology and expanding its scope of applicability with respect to physical database design. Research on conceptual data modeling, also within the framework of ADDM, is underway (see [MARC81c]).

4.1 Procedures to Aid in the Generation of Potentially Useful Database Schemas

FORM relies on a set of structure-based heuristics and constraints imposed by the designer to generate a set of potentially useful database schemas. For even medium sized problems it is essential that the designer judiciously constrain the generation of schemas or the problem becomes computationally intractable. To aid in this process procedures are being developed to assess the impacts of database activities as well as data structure through the use of "similarity" measures and clustering analysis.

4.2 Expansion of the Types of Partial Solutions Which May be Specified by the Designer

It is likely that the human designer can make some storage decisions for a given problem which may include design components not currently considered by the DBDS software. New DBDS features will allow the designer to: 1) cluster attributes; 2) absorb a subset of an entity's descriptors to form repeating groups representing only part of an entity's description; and 3) factor the problem into smaller problems with explicitly defined symbolic pointers crossing subproblem boundaries. A second set of features will allow the designer to: 1) partition entity instances into different datasets; 2) copy entity instances so that data is stored redundantly in different datasets; and 3) merge instances of different entities into the same dataset. Together these features will enable the DBDS to treat a wider variety of practical solutions.

4.3 Expansion of Access Paths

Access paths included in the DBDS are being expanded in two areas: 1) clustered data record retrieval and 2) directories to manage inverted and multilist file structures. Because of the physical characteristics of secondary storage devices, it is typically more efficient to access data records which are physically grouped or clustered together than to access data records which are physically distributed on the storage device. Commonly used access methods such as ISAM and VSAM recognize this efficiency and physically store data records according to the value of the key data item(s). Directories for inverted and multilist file structures can provide efficient support for a variety of retrievals and

they are commonly used in practice.

4.4 Generation of an Integrated Access Path Solution Space

In order for the DBDS software to select an efficient set of access paths for a given dataset and user activity, a space of potentially useful access paths must first be produced. The current procedure for producing this space heuristically generates potentially useful access paths individually for each retrieval activity which must be supported. Potential savings which might be realized by using a single access path to support multiple retrievals must be recognized by the designer who may then modify the solution space accordingly. While this may be a reasonable approach for simple database problems, the sheer volume of information which must be examined for realistic problems makes this approach infeasible. Additional file generation procedures which recognize the potential for sharing data access paths are in development. In addition, the designer will be able to direct the DBDS in the generation of access paths, e.g., to conform to generic access paths available in a particular DBMS.

4.5 Analysis of Data Maintenance Mechanisms

One of the modular components of the generalized file organization model used in the methodology is maintenance mechanisms. These are algorithms and structures which are used to manage secondary memory space for data update operations. A parametric model of secondary memory management schemes has been developed (see [MARC81a]) and incorporated into the methodology; however, the designer is responsible for selecting a set of memory management alternatives for consideration. The number of possible alternatives is enormous, putting a large burden on the designer and making complete enumeration computationally infeasible. Therefore, procedures are being developed to aid the designer in selecting a small set of alternatives for evaluation.

4.6 Algorithmic Improvements

Several of the DBDS modules contain optimization procedures for various design subproblems. Efforts are underway to improve the algorithmic efficiency of these procedures and where possible to develop new procedures which will integrate multiple subproblems and thus more accurately model database operations. In particular, the tasks of record formation and access path selection are independently optimized; however, they are clearly interdependent (see [MARC81b]). Procedures are in development to integrate these tasks.

REFERENCES

- CARL80a Carlis, J.V. An Investigation into the Modeling and Design of Large, Logically Complex Multiuser Databases, Ph.D. Dissertation, University of Minnesota, December, 1980.

- CARL80b Carlis, J.V. and S.T. March "A Computerized Database Design System Version 5: Users Manual," University of Minnesota, MISRC Technical Report TR-81-02, November, 1980.
- CARL80c Carlis, J.V. and S.T. March "A Computerized Database Design System Version 5: System Documentation," University of Minnesota, MISRC Technical Report TR-81-03, November, 1980.
- CARL81a Carlis, J.V. and S.T. March "A Multiple Level Descriptive Model for Expressing Logical Database Problems and Their Physical Solutions," University of Minnesota, MISRC Working Paper WP-81-10, March, 1981.
- CARL81b Carlis, J.V. and S.T. March "A Computer Aided Physical Database Design Methodology," University of Minnesota, MISRC Working Paper WP-82-10, November, 1981.
- CARL81c Carlis, J.V., G.W. Dickson and S.T. March "Physical Database Design: A DSS Approach," Proceedings of the Second Annual Conference on Information Systems, Boston, MA, December, 1981.
- CARL81d Carlis, J.V., and S.T. March "A Database Design Methodology: Forming Records," University of Minnesota, MISRC Working Paper WP-81-11, October, 1981.
- DeMA78 DeMarco, T. Structured Analysis and System Specification, Yourdon Press, 1978.
- DUHN78 Duhne, R.A. and D.G. Severance "Selection of an Efficient Combination of Data Files for a Multiuser Database," AFIPS Conference Proceedings 1978 National Computer Conference, Anaheim, CA (June 1978).
- EISN76 Eisner, M.J. and D.G. Severance. "Mathematical Techniques for Efficient Record Segmentation in Large Shared Databases," Journal of the ACM, 23, 4 (October 1976), pp. 619-635.
- LUM78 Lum, V. et. al. 1978 New Orleans Data Base Design Workshop, IBM Tech Report No. RJ2554, IBM, San Jose, 1978.
- MARC77 March, S.T. and D.G. Severance "The Determination of Efficient Record Segmentations and Blocking Factors for Shared Data Files," ACM Trans. Database Syst. 2, 3 (September 1977), 279-296.
- MARC78a March, S.T. Models of Storage Structures and the Design of Database Records Based Upon a User Characterization, Ph.D. Dissertation, Cornell University (1978).
- MARC78b March, S.T. and D.G. Severance "A Mathematical Modeling Approach to the Automatic Selection of Database Designs," Proceedings ACM SIGMOD, 1978, Austin, Texas, pp. 52-65.
- MARC81a March, S.T., D.G. Severance, and Wilens, M. "Frame Memory: A

Storage Architecture to Support Rapid Design and Implementation of Efficient Database," ACM Trans on Database Syst. 5, 3, September 1981.

- MARC81b March, S.T. "Techniques for Structuring Database Records," University of Minnesota, MISRC Working Paper WP-82-09, November, 1981 (submitted to ACM Computing Surveys).
- MARC81c March, S.T., and J.V. Carlis, "An Overview of Physical Database Design Research at the University of Minnesota," Internal Report, 1981.
- MAXW73 Maxwell, W.L. and D.G. Severance "Comparison of Alternatives for the Representation of Data Item Values in an Information System," Proceedings of the Wharton Conference on Research on Computers in Organizations, University of Pennsylvania (October 1973), 121-136.
- SENK75 Senko, M. E. "Specification of Stored Data Structures and Desired Output Results in DIAM II with FORAL," Proceedings of the International Conference on Very Large Data Bases, Framingham, MA, 1975, 557-587.
- SEVE77 Severance, D.G. and J.V. Carlis "A Practical Guide to the Selection of Record Access Paths," ACM Computing Surveys (December 1977), 259-272.

PHYSICAL DATABASE RESEARCH AT STANFORD

Gio Wiederhold	Daniel Sagalowicz
S. Jerrold Kaplan	SRI International
Stanford University	415-859-4840
415-497-0685	

1. INTRODUCTION

The Knowledge Based Management Systems (KBMS) Project at Stanford (supported by the Defense Advanced Research Projects Agency under contracts MDA903-77-C-0322 and N39-80-G-0132) addresses the problems of intelligent processing in large databases. The project is being directed by Stanford with cooperation from SRI International. Several of the topics under investigation are concerned with physical databases, and are detailed below. In addition, the project is supporting work on topics in logical data modeling, intelligent query processing, database machines ([SHAW80]), binding in information processing ([WIED81]), distributed databases ([CERI81], [MYNO81]), and natural language access to databases.

2. THE OPTIMAL DESIGN OF PHYSICAL DATABASES

A theoretical approach to the optimal design of large multifile physical databases has been devised. The design algorithm is based on the theory that given a set of join methods that satisfy a certain property called separability, the problem of optimal assignment of access structures to the whole database can be reduced to the subproblem of optimizing individual relations independent of one another ([WHAN81a]), ([WHAN81b]). Coupling factors are defined to represent all the interactions among the relations. This approach not only reduces the complexity of the problem significantly, but also provides a better understanding of underlying mechanisms.

This methodology for the design of the physical databases is also extended to include the join methods which are not in the separable set (we call these nonseparable join methods). This is a heuristic extension of the physical database design methodology based on the theory of separability. The basic design is formulated first, assuming that there are only separable join methods, and then is extended to include nonseparable join methods using heuristics. Work is in progress to extend the results to network (etc.) databases ([WHAN82]).

An improved formula for estimating the number of block accesses was developed during this effort. This formula is used in database systems when records are selected randomly and accessed in the order of their record identifiers (TID). This formula is computationally much more efficient than Yao's exact formula and improves upon Cardenas' formula in terms of accuracy.

3. FILE ACCESS SYSTEM

A file access system that uses symbolic keys to access variable length records based in PASCAL and supporting several host languages, including PASCAL, INTERLISP, and FORTRAN has been developed and is now being tested. The services that this system, named FLASH, expects from the underlying operating system are limited to directory management for named segments of secondary storage, and access to fixed size blocks or pages of these segments. It is specifically designed to provide strong and symmetric support facilities for databases, so that powerful database systems can become easier to implement than they are when using conventional files; files are designed with only programmer's needs in mind. The underlying structure uses B+ trees for storage of both primary and secondary keys. This system will be used to study various dynamic storage and retrieval strategies. The experience of implementing FLASH is already being used to better define an Input-Output package for the ADA language ([ALLC80]).

REFERENCES

- ALLC80 Allchin, J., A. Keller, and G. Wiederhold "FLASH: A Language-Independent Portable File Access System"; Proceedings of ACM SIGMOD Conference, May 1980, pp. 151-156.
- CERI81 Ceri, S., S. Navathe, G. Wiederhold "Optimal Design of Distributed Databases", to appear as a Computer Science Dept. Technical Report, Stanford University, November, 1981.
- MINO81 Minoura, T. and G. Wiederhold "Resilient Extended True-Copy Token Schema for a Distributed Database System", Symposium on Reliability in Distributed Software and Database Systems, IEEE, July 1981, Pittsburgh, Pa., pp. 1-12.
- SHAW80 Shaw, D. "Knowledge-Based Retrieval on a Relational Database Machine", PhD Dissertation, Stanford University, Computer Science Department report CS-80-823, September, 1980.

- WHAN81a Whang, K., G. Wiederhold and D. Sagalowicz "Separability: An Approach to Physical Database Design"; Proceedings of the Seventh International Conference on Very Large Data Bases, Cannes, France, September 1981, pp. 320-332.
- WHAN81b Whang, K., G. Wiederhold and D. Sagalowicz "Separability as a Physical Database Design Methodology"; Computer Science Laboratory report CSL TR 222, Stanford University, November, 1981.
- WHAN82 Whang, K., G. Wiederhold and D. Sagalowicz "Separability as a Tool for Partitioning the Physical Design of Network Databases", 1982.
- WIED81 Wiederhold, G. "Binding in Information Processing", Stanford University Computer Science report 81-851, May, 1981.

DATABASE DESIGN RESEARCH AT THE UNIVERSITY OF TORONTO

Stavros Christodoulakis
Anil Garg
C. C. Gotlieb
Geovane C. Magalhaes

Computer Systems Research Group
University of Toronto
Toronto, Ontario, Canada
416-978-4105
416-978-6025

A variety of research activities in physical databases is underway at the University of Toronto. Each of these activities is detailed in the following sections.

1. Improving the Performance of Database Systems

G. Magalhaes has taken an experimental approach to analyzing the performance of database systems [MAG82]. A methodology for collecting data on the behavior of large operational DBMSs was developed. It calls for the reproduction of the environment and activities, as observed in an actual on-line DBMS in a controlled (test) environment. Behavioral data is collected during the reproduction.

The methodology has been applied to an operational system that used a commercial DBMS. Five databases, having a combined size of about 200 megabytes, were chosen for study. The workload for a week period, was logged and reprocessed in the test environment. The resulting data contains information on: database definition and contents, transaction start time in the actual system, resources consumed per database processing module (CPU time, database and temporary storage I/O), system messages, selectivities (number of times that data item values are printed or number of records to be updated) and the identification of all database blocks in the order they were referenced (called reference strings). This set of observations, called the observation set, is believed to be the first detailed account of a large operational DBMS reported in the open literature.

The data in the observation set was analyzed in order to characterize the workload, its impact on the DBMS processing modules, and its reference strings. The reference strings were analyzed with respect to locality, sequentiality, and buffer management. Special transactions were analyzed in order to find ways to improve their performance.

The data gathered proved useful for obtaining parameters which characterized each database. These parameters are useful for modeling and theoretical studies, for improving the performance of a particular DBMS, and for making general suggestions about improving database performance. A principle conclusion is that the identified measurement tools and techniques should be embedded in a DBMS, because the effort needed to incorporate the tools is not excessive, the overhead for having these techniques capable of being switched on and off is not large, and the benefits gained with this mode of operation are far greater than the costs for implementing it.

2. Design of Database Performance Predictors

Database performance predictors are software tools that aid the selection, management, and refinement of database management software. K.C. Sevcik is examining several problems related to database performance predictors. A general multilevel analytic model has been formulated and important database design parameters and their interrelationships have been analyzed ([SEVC 81]). The lowest level of the model is based on queuing network models. Analytic techniques for transforming higher level parameters into queuing network parameters have been derived. The outputs of the queuing network model are device utilizations, transaction throughputs, and response times. The application of these techniques in the design of system specific performance predictors (like System 2000) have been demonstrated (see [CASA81], [BELL81]).

A next step in this research will be to implement a software package based on these designs which, together with the THEsolver queuing network solution package ([GRAH81]), is intended to provide a full prediction package for System 2000 environments. Other aspects of the design of database performance predictors involving higher levels of the model will also be investigated.

3. Estimation of Selectivities in Databases

Analytic models that have been proposed for the study of many database design and database performance evaluation problems require estimates of the number of records that are qualified by a query, the average number of records qualified by a set of queries, the number of blocks containing records that are qualified by a query, and the average number of blocks that contain records qualified by a set of queries. These estimates are called selectivities. S. Christodoulakis is pursuing research in this area ([CHR181a-d]). In order to estimate selectivities, the database contents and the data placement on devices, as well as the queries issued by the user population have to be modelled.

Common assumptions used for modelling database contents and data place-

ment on devices have been carefully analyzed and the impact of these assumptions in database design have been formally examined. Methods for more accurate modelling of database contents, data placement on devices, and user queries have been proposed. The improved approximations of selectivities take into account nonuniformities and correlations of attribute values as well as nonrandom placement of the qualifying records among the blocks of a file. Future research in this area will be both theoretical and experimental.

4. Multiple Key File Structures

Traditional file structures that provide multikey access to records, such as inverted files, are composed of one or more file structures that individually provide single key access to records. K.C. Sevcik is investigating multiple key file structures that enable records to be accessed efficiently given one or more of their keys. Emphasis is placed on the dynamic properties of these files and the symmetric treatment of the attributes. These file structures should permit record insertions and deletions and support both single value and range queries. One such structure, called the GRID FILE, which has the above properties, has been designed and simulated ([NIEV81]). The heart of the GRID FILE is the grid directory which is designed as a multidimensional array. The grid directory relates logical file partitions (grid blocks) with specific storage locations (buckets). Initial results show that the number of buckets grows in proportion to the number of records, and memory utilization remains around 70%.

5. Centralized and Distributed Concurrency Control

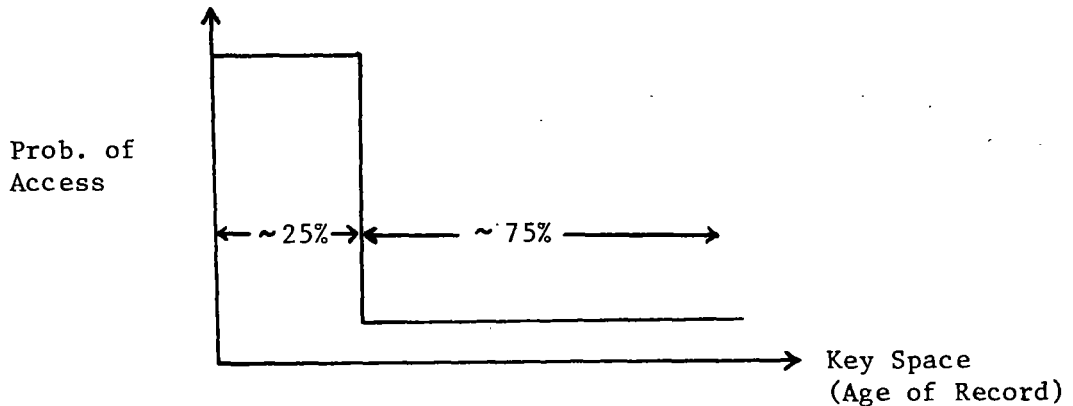
B. Galler is studying concurrency control mechanisms (CCMs) and their performance ([GALL82]). For the case of centralized databases, performance of CCMs are assessed by analytic modeling. Equations are derived to estimate the waiting times at different points in the system in a way such that the sum of the waiting times equals the total waiting time encountered by a job. The set of equations can be shown to converge. The analytic model produces good results when compared with a simulation of the same system.

For the case of distributed databases, a framework has been developed in which all known distributed database CCMs can be cast in order to compare their performance. Owing to this unified approach, certain tradeoffs become apparent and situations under which each CCM performs well or poorly can be determined. In addition, if a system designer indicates the characteristics of the system where a CCM is to be used, a choice among candidate CCMs may be possible.

6. Index Organizations for Skewed Access Patterns

An index organization is being developed by A. Garg to support a file that has the following characteristics:

- number of records in the file is about 6 million;
- the file has a growth rate of the order of 100,000 records per month;
- the deletion of a record from the file is rare;
- extensive modifications to the nonkey information in the record are performed, resulting in this information expanding in size;
- keys as well as associated data have variable lengths;
- the file must permit a browsable access, i.e., random followed by sequential access (forward as well as backward);
- the file has access probability distribution as shown in the figure below:



- the access probability is inversely proportional to the age of the record, i.e., a record enters a file with a high probability of access and slowly (over a period of one year or so) falls into the low access probability region.

These characteristics are specific to the "Author, Title & Subject Index" (ATS) of the University of Toronto Library Automation Systems (UTLAS) which provides browsable access to bibliographic information on author names, publication titles, or subjects as search keys. For an index organization to be of general applicability, the following features are also desirable:

- it should be able to handle any amount of deletions, i.e., perform well for an expanding, shrinking or even a stable (in terms of size) file;
- new, old, or any mix of keys can constitute the two discrete access probability regions.

A potential solution for the above problem will most likely fall into any one of the following two major classes of file organization: tree structures and hashing.

Among the well known tree structures, ISAM and B-trees are generally good but they do not efficiently support a skewed access pattern. The same is true of the various B-tree variants appearing in the literature (Prefix B-trees, Digital B-trees, etc.). One of the exceptions to this is a C-tree which is built taking into consideration access probabilities of the keys.

A variety of potentially good hashing techniques has appeared in the literature recently, namely, extendible hashing, spiral hashing, linear hashing with partial expansions, trie hashing, etc. For a hashing technique to be suitable for our purpose, it must be based on an order-preserving hash function to facilitate sequential access.

In either case, a two-level structure appears to be a most appropriate solution given the nature of the access pattern, with automatic (or triggered) migration of records from one level to another as their access probabilities change over time ([GARG82]).

7. File Organizations for Messages

D. Tschritzis and S. Christodoulakis are investigating problems of storage and retrieval in large files of messages. Messages consist of a header and a body. The header contains formatted data representing the most important characteristics of messages, e.g. origin, destination, etc. The body is textual. An organization for these files should be able to deal with a variety of messages in a flexible manner, to provide a simple and uniform interface to the user, and to be efficient for a large volume of messages. In addition, reorganizations should not be frequent (if required at all), and messages should be retrievable on the basis of their content.

To achieve these objectives, a file organization has been specified, analyzed, and implemented. The user of the system specifies a filter which restricts the attention to a manageable subset of messages. Messages within the subset are obtained for a final check. The identification of relevant messages is done by sequentially searching a small auxiliary file. Detailed performance analysis and integration of the facility in a larger system is underway ([TSIC81]).

REFERENCES

- BELL81 Bell, B. "Database System Performance Prediction: The first two Levels of a Multilevel Modelling Framework", M.Sc. Thesis, University of Toronto, 1981.
- CASA81 Casas-Raposo, I. "Analytic Modelling of Database Systems: The Design of a System 2000 Performance Predictor", M.Sc. Thesis, University of Toronto, July 1981.
- CHRI81a Christodoulakis, S. "Estimating Selectivities in Databases", CSRG report #136, University of Toronto, 1981.
- CHRI81b Christodoulakis, S. "A Multivariate Statistical Model for Database Performance Evaluation", Proceedings ORSA/TIMS Symposium on Applied Probability and Computer Science, The Interface, Boca Raton, 1981.
- CHRI81c Christodoulakis, S. "Estimating Block Selectivities", submitted for publication, 1981.
- CHRI81d Christodoulakis, S. "Implication of Certain Assumptions in Database Performance Evaluation", submitted for publication. 1981.
- GALL82 B. Galler "Concurrency Control Performance Issues", Ph.D. Thesis, University of Toronto, 1982 (to appear).
- GARG82 Garg, A. "Efficient Index Organization for Very Large Files with Skewed Access Patterns", (to appear) 1982.
- GRAH81 Graham, G. S. and J. Zahorian "THEsolver user guide", technical note 18, Computer Systems Research Group, University of Toronto, 1981.
- MAG82 Magalhaes, G. "Improving the Performance of Database Systems", Ph.D. Thesis, Department of Computer Science, University of Toronto, 1982.
- NIEV81 Nievergelt, J., H. Hinterberger and K. C. Sevcik "THE GRID FILE: An adaptable, symmetric multi-key file structure", presented at ECI81, Third Conference of European Co-Operation in Informatics, Munich, October 1981.

TSIC81 Tsihritzis, D. and S. Christodoulakis "Message Files", Submitted for publication, 1981.

SEVC81 Sevcik, K. C. "Database Performance Prediction Using an Analytic Model", Proceedings of Very Large Database Conf., 1981, pp. 182-198.

DATABASE STORAGE STRUCTURES RESEARCH AT THE UNIVERSITY OF WATERLOO

Gaston H. Gonnet, Per-Åke Larson,
J. Ian Munro, Doron Rotem,
David J. Taylor, and Frank Wm. Tompa

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
519-885-1211

The principal directions of the data structuring effort at Waterloo are the development of (1) search techniques for database applications, (2) storage methods that are particularly frugal in their memory requirements, (3) representation and manipulation techniques to increase robustness, and (4) tools to design and evaluate composite structures. A central application for these approaches is the design of videotex database systems.

1. Search Techniques

Extensive research effort has been and continues to be directed at investigating alternative representations and algorithms for index structures. Typical of work in this area is that reported in Gonnet and Munro's "Efficient ordering of hash tables" ([GONN79]) and Gonnet, Ziviani, and Wood's "Analysis of 2-3 trees and B-trees" ([GONN81b]).

Wherever feasible, expected behaviour, worst-case behaviour, and expected worst-case behaviour are investigated in order to understand standard algorithms and to develop efficient alternative ones. In general the analysis of structures is carried out along two directions. First simulation is used to test algorithms' practicality as well as to develop hypotheses for anticipated performance. Once such hypotheses are developed, there remains the task of proof or refutation. The subsequent analyses frequently rely heavily on the use of the Maple algebraic manipulation system, developed by Geddes and Gonnet ([GEDD81]).

2. Compact Structures

The term "implicit data structure" has been proposed by Munro to refer to a storage scheme under which the structural information is implicit in the relative order of the data rather than explicit in pointers (e.g., the scheme used for heapsort). Such structures are suitable for placing data on secondary store (where pointer chasing is expensive) as well as for organizing the records within a physical block (where the storage overhead of pointers is detrimental). Munro and Suwanda's "Implicit data structures for fast retrieval and update"

([MUNR80]) provided implicit techniques for efficiently maintaining index structures and for performing multikey searches. More recent work by Munro and Poblete has led to such structures which permit searches and insertions in $O(\log^2 n)$ time with no extra space.

It is well-known that the expected retrieval time for data is strongly dependent on the relative probabilities of requests for elements in a structure, which frequently will not only differ but will also be unknown. Rather than approximating these probabilities by storing the number of accesses to each element, a more space-efficient technique is that of adapting the elements' order incrementally with each access (or set of accesses). Work on these self-organizing structures is exemplified by Allen and Munro's "Self organizing binary search trees" ([ALLE79]), Gonnet, Munro, and Suwanda's "Exegesis of self-organizing linear search" ([GONN81a]), and Matthews, Rotem, and Bretholz's "Self-organizing doubly linked lists" ([MATT80]).

As a third related aspect, Gonnet and Larson are investigating hashing organizations that minimize the number of probes to secondary storage while maintaining in main memory only a few bits per bucket. Three approaches are being examined: What is the best method that uses only one bit per bucket? How many bits are required to guarantee retrieval in one probe? Is there a technique which by suitably ordering the elements prior to insertion requires only one bit per bucket in main memory and guarantees retrieval in one probe?

3. Robust Structures

As organizations become dependent upon computer systems, they become highly sensitive to failures in these systems. One approach to reducing occurrences of failures is to make systems fault tolerant. Thus Munro and Taylor are involved in developing data structures which are tolerant of both hardware and software faults, as exemplified in Taylor, Morgan, and Black's "Redundancy in data structures" ([TAYL80]).

One aspect of the work is the development of general techniques for analyzing the robustness of data structures, for example, calculating upper and lower bounds on the number of errors which can be detected and/or corrected in a structure. Some commonly used data structures, including linear linked lists, binary trees and, most importantly, B-trees have been examined in detail. The study of these structures has included empirical investigations (with pseudo-random errors) as well as theoretical analyses.

4. Composite Structures

The behaviour of many storage structures has been analyzed in isolation, but there has been much less work devoted to studying the interaction of superimposed structures. Tompa has developed a framework for examining data structures at several levels of detail, and this has been

particularly formalized for the representational levels in Gonnet and Tompa's "A constructive approach to describing algorithms and their data structures" ([GONN80]).

Tompa and Ramirez's "An aid for the selection of efficient storage structures" ([TOMP81]) describes several algorithms based on dynamic programming to create a composite structure from a set of compatible representations for the components. Taylor and Black are developing techniques for evaluating the robustness of such a composite representation. The goal of current research is to investigate the practicality of these ideas and to enhance them as necessary to build an integrated facility.

5. Videotex

One emphasis of the data structuring effort is the application of its research to the design of videotex databases. Initial observations are reported in Tompa, Gecsei, and Bochmann's "Data structuring facilities for interactive videotex systems" ([TOMP81]). Contributing to the state-of-the-art in videotex is dependent on direct access to a videotex server. To this end, the CCNG/Telidon host software, running under Unix, was developed by members of the Computer Communications Networks Group at the University of Waterloo.

Gonnet, Manning, and Tompa are conducting research into the design, analysis, simulation, and implementation of mnemonic page labelling schemes, keyword access to Telidon data, data structures for interfacing videotex data to external databases, and memory management schemes for efficient page retrieval. Related research on distributed software is also being applied to videotex problems where appropriate.

REFERENCES

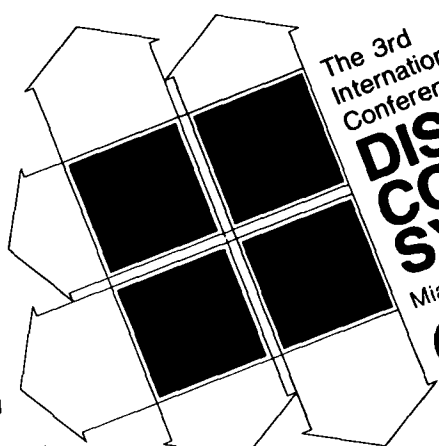
- ALLE79 Allen, B. and J.I. Munro "Self Organizing Binary Search Trees", JACM 25,4, (October 1979). pp. 526-535.
- GEDD81 Geddes, K.O. and G.H. Gonnet ""Maple User's Manual", U. of Waterloo, Department of Computer Science, Tech. Rept. CS-81-25 (July 1981) 40 pp.
- GONN79 Gonnet, G.H. and J.I. Munro "Efficient Ordering of Hash Tables", SIAM J. Comp. 8,3 (August 1979), pp. 463-478.
- GONN80 Gonnet, G.H., and F.W. Tompa "A Constructive Approach to the Design of Algorithms and their data structures", Tech. Rep. CS-80-47, Dept. of Computer Science, University of Waterloo, (November 1980), 15 pp.
- GONN81a Gonnet, G. H., J. I. Munro and H. Suwanda "Exegesis of Self-

- organizing Linear Search", SIAM J. Comp. 10,3 (August 1981), pp. 613-637.
- GONN81b Gonnet, G.H., N. Ziviani and D. Wood "Analysis of 2-3 Trees and B-Trees", CS-81-21, Dept. of Computer Science, University of Waterloo, (June 1981) 30 pp.
- MATT80 Matthews, D., D. Rotem and E. Bretholz "Self Organizing Doubly Linked Lists", Intern. J. Comp. Maths, A, 8 (1980) pp. 99-106.
- MUNR80 Munro, J.I. and H. Suwanda "Implicit Data Structures for Fast Retrieval and Update", JCSS 21,2, (October 1980), pp. 236-250.
- TAYL80 Taylor, D. J., D. E. Morgan and J. P. Black "Redundancy in Data Structures". IEEE Trans, Soft. Engg. 6.6. (November 1980), pp. 585-602.
- TOMP80 Tompa, F.W. and R. J. Rameriz "An Aid for the Selection of Efficient Storage Structures", CS-80-46, Dept. of Comp. Sci., University of Waterloo, (October 1980) 26 pp.
- TOMP81 Tompa, F.W., J. Gecsei, and G. V. Bochmann "Data Structuring Facilities for Interactive Videotex Systems", Computer 14,8 (August 1981) pp. 72-81.

SCOPE

The scope of this conference encompasses the technical aspects of specifying, designing, implementing, and evaluating distributed computing systems. In such systems, there is a multiplicity of interconnected processing resources able to cooperate under system-wide control on a single problem with minimal reliance on centralized procedures, data, or hardware. The location of computing resources may span the spectrum from physical adjacency to geographical dispersion. The topics of interest include the following aspects of distributed computing systems:

- SYSTEM AND HARDWARE ARCHITECTURE, INCLUDING SIMD
- DECENTRALIZED CONTROL, EXECUTIVES, AND OPERATING SYSTEMS
- DISTRIBUTED DATABASES
- LOGICAL AND PHYSICAL INTERCONNECTION NETWORKS
- COMPUTER COMMUNICATION
- SOFTWARE ENGINEERING AND PROGRAMMING LANGUAGES
- SURVIVABILITY, RELIABILITY, AND FAULT TOLERANCE
- SPECIFICATION, VERIFICATION, AND VALIDATION
- DESIGN METHODOLOGIES
- VLSI-BASED SYSTEMS
- ANALYSIS, MODELING, AND MEASUREMENT
- APPLICATIONS, INCLUDING SIMULATION



The 3rd International Conference on
DISTRIBUTED COMPUTING SYSTEMS
Miami/Ft. Lauderdale, Florida • October 18-22, 1982

CALL FOR PAPERS

PAPER SUBMISSION REQUIREMENTS

Five copies of the manuscript should be submitted to the Program Chairman by March 1, 1982. The conference language is English and papers are restricted to a maximum of 20 double-spaced pages including figures. Each copy must contain a 150 word abstract and a title page with complete mailing addresses and phone numbers of all authors. A submission letter must accompany the paper. It should contain a commitment to present the paper at the conference if accepted. In case of multiple authors, an indication of which author is responsible for correspondence and preparing the camera-ready copy for the proceedings must be included. Authors will be notified of acceptance by July 1, 1982 and will be given instructions for final preparation of their papers. Deadline for receiving camera-ready papers is August 1, 1982

TUTORIALS

In addition to papers, proposals for one-day tutorials are solicited in any of the above areas. Such proposals should be submitted to the Tutorial Chairman by Dec. 16, 1981. The authors of accepted proposals will be asked to submit complete tutorial texts within a month.

OUTSTANDING PAPER AWARDS

The two best papers will receive awards. All outstanding papers will be considered for planned special issues of journals.

CONFERENCE LOCATION

Diplomat Hotel in Hollywood, Florida near the international airport at Miami. Beachfront resort with tennis, golf, swimming, shopping, and boating.

Sponsored by
IEEE COMPUTER SOCIETY
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS INC.
In Cooperation with
Information Processing Society of Japan (IPSJ)
Institut National de Recherche d'Informatique et d'Automatique (INRIA)

General Chairman

H. J. Siegel
Purdue University
School of Electrical Engineering
West Lafayette, IN 47907

Program Chairman

Carl G. Davis
Ballistic Missile
Defense Advanced
Technology Center
ATTN: BMDATC-P
P. O. Box 1500
Huntsville, AL 35807

Tutorial Chairman

K. H. Kim
Computer Science Dept.
LIB 630
University of South Florida
Tampa, FL 33620

Exhibits Chairman

Edith W. Martin
Govt. Systems/CDC
Suite 520
5500 Interstate N. Pkwy.
Atlanta, GA 30328

Standing Committee Chairman

Charles R. Vick
Systems Control, Inc.

Awards Chairman

T. Y. Feng
Ohio State University

PROGRAM COMMITTEE

- Bob Arnold** (Honeywell)
- Geneva Belford** (U. Ill.)
- Bill Carroll** (U. Texas)
- Glenn Cox** (General Research Corp.)
- Berry Gilbert** (Mayo Clinic)
- J. C. Huang** (U. of Houston)
- Robert Keller** (U. Utah)
- Annette Kryglet** (Defense Mapping Agency)
- Bill McDonald** (Systems Development Corp.)
- Vic Nelson** (Auburn U.)
- Peter Ng** (U. Missouri)
- Dan Sieworek** (Carnegie-Mellon U.)
- Harold Stone** (U. Mass.)
- Ken Thurber** (Architecture Tech. Corp.)
- Larry Wittle** (SUNY/Buf.)
- Ken Batcher** (Goodyear)
- Bharat Bhargava** (U. Pittsburgh)
- Doug DeGroot** (IBM)
- Clarence Glesse** (Dept. of Army AIRMICS)
- Bob Heath** (U. Kentucky)
- Lana Kartashev** (U. Nebraska)
- Jack Lipowski** (U. Texas)
- Mike Liu** (Ohio State U.)
- John Musa** (Bell Labs)
- Chong Nam** (Systems Control, Inc.)
- C. V. Ramamoorthy** (UC/Berkeley)
- Azriel Rosenfeld** (U. Maryland)
- Steve Smolar** (Schlumberger-Doll Research)
- Joe Urban** (U. Southwestern La.)

International Associate Chairpeople

- Helmut Kerner, Austria
- C. M. Woodside, Canada
- Paul Pearson, England
- Gerard Le Lann, France
- Herbert Weber, Germany
- Mariagiovanna Sami, Italy
- Hideo Aiso, Japan
- J. Wilmink, Netherlands
- R. C. T. Lee, Taiwan
- Leah J. Siegel, U.S.A.

Additional Program Committee Members will be chosen by the International Associate Chairpeople.

If you wish to receive a copy of the Advance Program for the Third International Conference on Distributed Computing Systems, clip and mail this coupon to: Harry Hayman, IEEE Computer Society, 1109 Spring Street, Suite 202, Silver Spring, MD 20910.

NAME _____ EMPLOYER _____
STREET _____
CITY _____ STATE _____ ZIP _____ COUNTRY _____

PHYSICAL DATABASE KEYWORD INDEX

ADDM	31
analysis of algorithms	49,50
analytic models	6,16-18,25,26,28,33,37,43,44
assumptions of analytic models	6,44
atomic configuration/costs	9
backup, checkpoint/restart, recovery	27,28
B-trees	49,50
cache buffers for disks	3
clustering	10,19,25,35
concurrency control mechanisms	16,17,27,44
concurrent reorganization	17
data compression	7,12-15
database design and evaluation packages	9-11,23,26-27,33,42-43
database implementation/software projects	6,12-14,19,26-27,33,40,46,51
database performance	6,18,19,26,33,42-44,51
database performance monitor	42
database workload	9,42
DBDS	31-34
DBDSGN	9-11
DDT	18
decomposition	5
descriptive models of physical databases	6,14,32
differential files	6,28
empirical studies	2,42
estimating number of block accesses	40,43
fault tolerant structures	50
file access patterns (empirical)	2,42
file locks	16,17
file migration	2,14
file systems	2,3
file/database statistics (empirical)	2,10,42
files for storing messages	46
FLASH	40
frame memory	26
GRID FILE	44

hash-based files	49-50
hierarchical databases	25
implicit data structures	49
index selection	6,9-11,25
index structures	16,45,49
inverted files	9-11,35
I/O system optimization/architecture	3
joins	9,18,39
linksets	5-6
linear splitting	6
logarithmic access	14
modeling data and queries	43-44
multifile databases	5,32,39
multikeyed file structures	16,44
multilevel model	43
multilist files	35
network databases	5,25,39
optimization techniques/heuristics	6,10,17,18,27,33,35,36,51
optimizer	9
plausible indexes	10
query optimization	17-18
queueing models	43,44
reference strings	42
relational database system	9
reorganization	6,17
run length encoding	12,14
SEEDIS	12
selectivities	42,43
self-organizing files	6,17,50
separability	39
simple files	5-6
skewed access patterns	45
statistical databases	6,12-15
storage allocation	17
System R	9
System 2000	42,43
transactions	6
transposed files	6,7
Videotex Systems	51
WORKBENCH	23

