Data Base Engineering

Vol. 1 No. 1 **Contents** March 1977

A Quarterly Bulletin published by the IEEE Computer Society Technical Committee on Data Base Engineering

Data Base Engineering Bulletin
Volume 1, No. 1.
March, 1977

A Quarterly Publication of the IEEE Computer Society
Technical Committee on Data Base Engineering

Data Base Engineering Bulletin is a quarterly publication of the IEEE Computer Society Technical Committee on Data Base Engineering. Its scope of interest includes: data structures and models, access strategies, access control techniques, data base architecture, data base machines, intelligent front ends, mass storage for very large data bases, distributed data base problems and techniques, data base software design and implementation, data base utilities, etc.

Contribution to the Bulletin is hereby solicited. News items, letters, technical papers, book reviews, meeting previews and summaries, 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 Data Base Engineering, the IEEE Computer Society, or organizations with which the author may be affiliated.

Membership in Data Base Engineering Technical Committee (at $5 per year) is open to IEEE Computer Society members, student members, and associate members. Nonmembers of IEEE Computer Society may join the Technical Committee (at $10 per year). Library subscriptions are available (at $10 per year).

Research into Database Machines at the University of Utah

Diane C.P. Smith
John Miles Smith

Department of Computer Science
University of Utah

The current concept of a "database machine" is that of a machine in which relatively expensive relation (or file) processing functions such as boolean searching, cross-referencing and updating are consigned to hardware [1, 4, 5, 8]. These functions, oriented towards relation-at-a-time processing, are higher-level than the "GET" and "PUT" primitives of current tuple (or record)-at-a-time processing. The attraction of this is in its promises of a more rapid execution of these functions and a representation of information closer to the way it is visualized by users.* However, the database machine concept can be further extended by considering even higher-level functions, which in addition to searching and updating, maintain consistency within and between relations. Database machine research at the University of Utah is concerned both with the investigation of boolean-search associative devices and with the development of consistency maintaining functions appropriate for hardware (as well as software) implementation.

The investigation of boolean-search associative devices has taken three forms:

1) the development of an architecture for a head-per-track rotating associative device [4],

2) a study of efficient use of the boolean-search primitive to

*This latter feature is important to the production of more reliable database software.

implement additional relation-oriented functions such as sorting [3], and

3) a study of cost-effective ways of providing the boolean-search primitive to large scale database applications [2].

The architecture developed for head-per-track rotating associative devices is called the rotating associative relational store (RARES). It is an associative memory constructed by adding a relatively inexpensive content-addressing mechanism to an existing head-per-track rotating device. It is distinct from other designs in that it utilizes a novel "orthogonal" storage layout. This layout allows a high output rate of selected tuples even when a sort order in a stored relation must be preserved.

The study of the search function as a primitive in defining other database functions concentrated on the sort function. An algorithm called the "bucket sort" was developed to exploit the associative search capability. Compared to the standard sort-merge algorithm, this algorithm requires at most the processing time necessary for the initial run generation and the first pass of the merge operation.

The designs of rotating associative devices are currently based on head-per-track technology. Such associative devices can be used effectively for small scale ($<10^7$ byte) applications. However, the order of magnitude cost differential (for current and predicted for new storage technologies) between head-per-track and movable-head devices dictates that large scale databases be stored on the movable-head devices. To make the boolean search capability available to large scale applications, either a head-per-track associative device can be used in a memory hierarchy as a cache into which data can be paged from conventional movable-head devices, or the search logic can be moved to the heads of the movable-head devices.

The associative cache and the associative movable-head devices can be thought of as two different ways of distributing search logic. In the associative cache approach, the logic is concentrated on a small number of head-per-track devices, one logic unit per track. If the data items to be located are not in the cache, more data must be paged in from conventional movable-head devices. In the associative movable-head device approach, the logic is distributed across a large number of movable-head devices. If the data items to be located are not found on the cylinder being accessed, the head (and logic) are moved. Thus, the relationship between the two approaches is that a page transfer in one is conceptually equivalent to a head movement in the other. However, if both approaches use the same amount of parallelism, the associative cache is less efficient than the associative movable-head devices. The tradeoffs between these two approaches are being investigated further and the design of an associative movable-head device is being developed.

Research at the University of Utah has also concentrated on the development of data accessing primitives that will guarantee consistency within and between relations [6, 7]. This investigation is based on a semantic approach that treats an integrated database as a model of some real-world system. Constraints existing in the real-world system must be represented in the model (database) by the actions of the operators and the interrelational structure of the database. For example, if a relation is to model a set of distinct real-world objects, there must be a unique identifier associated with each tuple in the relation. An insert should maintain the invariance of this condition. Similarly, if the existence of a value for some domain in one relation implies the existence of an object represented by a tuple in another relation, a delete in this second relation must maintain the invariance of this condition. In [6] and [7] a set of invariants is identified that capture constraints true of all real-world systems. The universality of these invariants recommends them for hardware support. However, the method of providing this support remains to be determined.

References

1. Berra, P.B., Some problems in associative processor applications to data base management. Proc. AFIPS 1974 NCC, Vol. 43, AFIPS Press, Montvale, N.J., pp. 1-5.

2. Lin, C.S., Report on Associative Movable-head Devices, in preparation.

3. Lin, C.S., "Sorting with Associative Secondary Storage Devices", submitted for publication.

4. Lin, C.S., D.C.P. Smith and J.M. Smith, "The Design of a Rotating Associative Memory for Relational Database Applications", ACM Trans. Database Systems, Vol. 1, No. 1 (March 1976), pp. 53-65.

5. Ozkarahan, E.A., S.A. Schuster, K.C. Smith, RAP-An associative processor for data base management. Proc. AFIPS 1975 NCC, Vol. 44, AFIPS Press, Montvale, N.J., pp. 379-387.

6. Smith, J.M. and D.C.P. Smith, "Data Base Abstractions: Aggregation", to appear in Comm. ACM, 1977.

7. Smith, J.M. and D.C.P. Smith, "Data Base Abstractions: Aggregation and Generalization", to appear in ACM Trans. Database Systems, 1977.

8. Su, S.Y.W., and G.J. Lipovski, CASSM: A cellular system for very large data bases. Proc. of Very Large Data Base Conf., Sept. 1975, pp. 456-472.

Cost Analysis of Data Distribution*

by

Geneva G. Belford
and
John D. Day

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Introduction

The advantages of distributing a data base in a network environment have been the subject of much discussion. There has been a scarcity, however, of research attempting to quantify these advantages or to investigate the various tradeoffs and to determine just how great the advantages are. Because of this lack of quantitative data, we recently undertook a preliminary study of the costs of data distribution. Our goal was to gain some understanding of where the major costs are incurred and under what circumstances distributing a file system is worthwhile. Space limitations preclude our giving more than an overview of the work here. Readers interested in the details may consult our report [1].

The Model

For many of the cost-related questions that arise in the development of a distributed data base system (such as those concerned with the costs of queries, updates, backup, recovery, etc.), the system can at first be viewed as a storage hierarchy. That is, to a local

*This work was performed as part of Contract DCA100-75-C-0021 with the Command and Control Technical Center - WWMCCS ADP Directorate of the Defense Communications Agency.

process or user submitting a query to a remote site, storage devices at that site appear as further levels of the hierarchy. From this point of view the network is another channel with some special cost considerations. We have constructed and analyzed such a simple, storage-hierarchy model of distributed data processing. This approach allowed us to investigate the tradeoffs offered by various strategies without becoming involved in extraneous issues, such as which remote site in the network is the best location for the data. In future refinements of the model, we plan to include effects of processing data at the remote site in order to take advantage of cheaper computation or possible parallelism.

Modeling storage hierarchies. Well before networks existed, the question arose as to where one should place a given file in a storage hierarchy - i.e., a set of memory devices of varying accessibility (core, disk, tape, etc.) connected to a single computer. A particularly comprehensive cost model [2] for this problem has recently been developed by Lum and co-workers at IBM Research (San Jose). This model differentiates between random and sequential forms of data access and includes considerations of staging, channel costs, CPU overhead, etc. Because of its completeness, we considered this model an appropriate one for extension to the network case.

Lum's model primarily addresses the problem of "data staging" or "data migration". In other words, when a file or data set is not being used (i.e., is inactive) it is stored on one device (usually a relatively slow, inexpensive one). Then, when the data set is to be used, it is moved to a faster, more expensive device so that the program will waste fewer resources waiting for data. The basic question addressed is: given the accessing characteristics (number of reads and writes, proportion of time the file is in use, etc.), where in a given hierarchy should the data set be stored when it is inactive and where should it be stored when it is active?

Lum develops an objective function which gives the cost of accessing a data set which is stored on one device when inactive and another (possibly the same device) when active. In his model the entire data set is moved from the inactive device to the active one. (We relaxed this requirement in our model.)

The selection algorithm is then quite straightforward. The objective function is evaluated for each pair of devices in the hierarchy. The lowest cost then indicates on which pair of devices the data should be located.

Lum and his co-workers make several simplifying assumptions, most of which can be relaxed at the cost of a more complex cost function. They assume that, for data sets, system paging activity will not significantly affect cost. However, it would probably be necessary to relax this constraint if one wished to consider costs incurred by program activity. They further assume that transfers are direct rather than through core and that there are no flow control problems (i.e., a fast device can always accept data from a slow device). They also assume that transfers are not constrained by the capacity of the device the data set is being moved to. Fortunately, these last two assumptions can both be dropped at the cost of a more complex cost formula. When a network is added to the hierarchy, flow control cannot be ignored.

There are, however, more troublesome deficiencies in the IBM group's approach. First, they implicitly assume a very low rate of data access. Costs which may in fact grow very rapidly with increased load are assumed to be proportional to the number of accesses or to the amount of data handled. Second, they include a number of terms which represent lost CPU time induced by delays in accessing devices. This seems to represent an effort to parcel out, in a simple way, the cost of the inevitable CPU idle time among the various processes. At the same time they omit some real CPU costs which are incurred in the data transfer process and which may be significant. In addition, they assume that only CPU idle time adds significant costs and ignore costs due to other idle equipment, such as channels.

Adding a remote site to the hierarchy. In spite of our reservations, we decided to work initially with Lum's model. We began with Lum's cost formula, with its terms for storage, data transfer, and accessing, and added network terms - including costs for data transfer to, from and over the network, as well as protocol costs. In adding these terms, we felt that, if only for consistency, we should follow the spirit of Lum's model. Hence the extended model also has terms involving costs of "lost" CPU idle time.

The cost formula we arrived at has six components:

$$\begin{aligned}\text{COST} = {} & \left\{\begin{matrix}\text{storage}\\ \text{cost}\end{matrix}\right\} + \left\{\begin{matrix}\text{cost to move data between inactive}\\ \text{remote level and highest remove level}\end{matrix}\right\} \\ & + \left\{\begin{matrix}\text{cost to move data between highest}\\ \text{remote level and the network}\end{matrix}\right\} + \left\{\begin{matrix}\text{network}\\ \text{costs}\end{matrix}\right\} \\ & + \left\{\begin{matrix}\text{cost to move data between the}\\ \text{network and the local active level}\end{matrix}\right\} + \left\{\begin{matrix}\text{costs for the process}\\ \text{to access the local}\\ \text{active level}\end{matrix}\right\}\end{aligned}$$

The basic situation to which our formula is designed to apply can be described briefly as follows. A local process, active for a portion of each day, manages the queries to and updates of a data base. The data base is stored permanently on a mass storage device at a remote site on the network. When the local process becomes active, it causes the data base to be moved to a rapid-access device at the local site. At the end of the day's work, the data base is moved back across the network to the permanent storage location.

Notice that moving the data base over the network is a multi-step process. The data must be first staged at the remote site, then moved onto and across the network, and then finally picked up from the network by the local host. In contrast, local data staging from one device to another involves only one such transfer. Essentially, the third, fourth, and fifth terms of the cost formula are missing when the permanent storage device is local.

Cost Analysis

It is clear that, in order for remote storage to be cost effective, there must be sources of savings large enough to counter-balance the three additional cost terms in the network model. Further-more, examination of the cost formula shows that potential savings are limited to the first two terms. That is, either storage or staging (or both) must be cheaper at the remote site than they are locally. This is not an unreasonable requirement. However, the magnitudes of the cost differentials are critical.

To see how large the savings must be, we have carefully examined typical costs incurred in transferring data over a network. The actual network costs have two major components: the setup cost for using the network and the cost of the traffic sent on the network. The former includes protocol negotiation and processing costs. The latter includes the cost of transmitting protocol messages as well as trans-mitting the data base itself. Not surprisingly, the major cost - generally by an order of magnitude or more - is that of actually trans-mitting the data base across the network. (The costs for data transfer between host and network can become comparable when the network bandwidth is very small.)

At the present time, it is not cheap to transmit a large amount of data across a network. One quoted commercial rate is $1.25 per 1000 125-byte packets. A one-megabyte data base would then cost $10 to ship. If this much data is staged frequently (daily, say) from a remote site, the cost is clearly far more than could possibly be saved because processing and/or storage is cheap at the remote site. (Disk storage of a megabyte for a month is unlikely to cost more than $200 anywhere. Thus, even if remote storage is free, the cost differential will not be enough to offset the network costs. It is also unrealistic to hope for large differentials in processing costs.)

As the amount of data shipped back and forth across the network decreases, the likelihood that a distributed strategy will be cost-effective increases. We have therefore used our cost formula to get a preliminary idea of the value of local data caching. Under this strategy, the local system maintains a partial copy of the data set. The contents of this copy are determined by the results of past accesses or in some cases by some knowledge of what will be needed. When the user requests data, the system first looks to see if the data is local; if so it is fetched from the local storage medium; if not then it must be retrieved from the master copy over the network. This is a sort of "network working set" strategy. Investigating the properties of such a strategy, we found a rather steep rise in cost as the fraction of requests that must use the network increased. Of course, whether or not most requests can be answered locally depends upon the size of the local store and the degree of locality exhibited by the requests. However, if the fraction of remote requests can be kept low, it appears that significant savings can be achieved by the local caching of data. Further study is needed of the locality properties of data base activity, the goal being to determine what the size of the local store must be so that a large fraction of the requests may be satisfied locally.

Conclusions

The main result of our study was that heterogeneity is a necessary requirement for remote storage to be cost effective. This conclusion is intuitively reasonable. Transferring data over the network must cost something - and this additional cost is inevitably incurred if the data is stored at a remote site. In order to offset the network costs, the remote site must be significantly cheaper, in some respect, than the local site.

It should be emphasized that in most situations the cost differential due to heterogeneity must be sizable - not small percentages, but orders of magnitude. As the amount of data transported over the network decreases, the network costs can decrease to the point where smaller cost differentials can make remote storage economical.

There are several ways in which the necessary heterogeneity may be achieved:

1. Excess capacity. That is, some sites may be less heavily loaded either because of usage patterns or because of system differences. Thus, even though "real" system costs are not very different, it may be worthwhile to store the data base at a remote, underutilized host.
2. Inexpensive storage. Special facilities, such as the ARPA Network Data Computer, may be available at one site.
3. Artifically-induced heterogeneity. This may be achieved by arbitrarily setting charging rates at some sites so that they are significantly cheaper than at other sites. (There could be various policy reasons for doing this - including encouraging the use of underutilized sites.)

At this point it is probably a good idea to remind the reader of the limitations of our model. The model only describes data staging and no other aspect of distributed data management. The questions we raised with respect to Lum's model carry over. On the other hand, the questionable terms in the model are usually small enough so that their probable inaccuracies are unlikely to seriously affect the kind of broad conclusions that we want to draw.

In particular the model seemed adequate for an initial study of the key question: Is it ever more economical to store data at a remote site (instead of locally) and bring it over the network when needed? We have used our model to study this question. We believe that the results of the study have validity for real systems. Improvements in the model are not expected to change our conclusions significantly.

References

1. Day, J.D. and Belford, G.G. A Cost Model for Data Distribution. CAC Document Number 179 (1975) Center for Advanced Computation, University of Illinois at Urbana-Champaign, Urbana, Illinois.

2. Lum, V.Y.; Senko, M.E.; Wang, C.P.; and Ling, H. "A Cost Oriented Algorithm for Data Set Allocation in Storage Hierarchies," CACM 18, (1975) pp. 318-322.

WORKSHOP ON OPERATING AND DATA BASE MANAGEMENT SYSTEMS

Norris University Center
Northwestern University
Evanston, Illinois 60201
March 21-22, 1977

Sponsored by IEEE Computer Society
(Technical Committees on Data Base Engineering and Operating Systems)
and Northwestern University

ADVANCED PROGRAM

MONDAY - March 21 — 8:30 a.m.
Registration Material Pick-up

Opening Session — 9:00 a.m.
Chairman: S. S. Yau
Northwestern University

"An Overview of the Critical Issues in the Area of Operating and Data Base Management Systems, " R. Muntz, UCLA

COFFEE BREAK — 9:45 a.m.

Session I — 10:15 a.m.
Data Base/Operating Systems Environment

Chairman: G. J. Popek
UCLA

"Conveyance of Operating and Data Management Systems," W.D. Haseman, Carnegie-Mellon University, Pittsburgh, PA. (2)

"System R and Its Operating System Environment," M.W. Blasgen, IBM Research Lab. San Jose, CA. (4)

"User, Operating System, Database Management System: Global Structure and Functional Dependencies," K.J. McDonell, University of Alberta, Canada (13)

"The Influence of the Operating System in the Design and Implementation of Trigger Subsystem for Database System," K.P. Eswaran, IBM Research Lab., San Jose, CA. (7)

DISCUSSIONS

Session II: — 1:45 p.m.
Security in Database and Operating Systems

Chairman: R. S. Gaines
Rand Corporation, Santa Monica,CA.

"The Relationship Between Operating Systems and Data Base Security: A Survey," E.P. Fernandez and C. Wood, IBM Los Angeles Scientific Center, Los Angeles, CA.

"Similarities and Differences Between System and Data Management Security—A Study Toward a Kernel Design of Database Security Software," D. Downs and G. Popek, University of California at Los Angeles, CA.(3)

"The Requirements for a Secure User Interface to Database and Operating Systems," S.R. Ames, Jr., The MITRE Corporation, Bedford, MA. (10)

"The Use of Reference Monitor Technique for Data Security in Computer Systems," M. Grohn and G. Kirkby, I. P. Sharp Assoc. Ltd., Ottawa, Canada. (1)

DISCUSSIONS

COFFEE BREAK — 3:45 p.m.

Session III — 4:15 p.m.
Performance Evaluation and Design Issues

Chairman: J. Mehl
IBM Research Lab., San Jose, CA.

"Performance Evaluation of Generalized Management Information System on a Virtual Memory Operating and Computer System--A Case Study of GMIS on VM/370," L.E.S. Sarmento and P.S. Chen, M.I.T., Cambridge, MA. (5)

"Database Reorganization Issues Related to Operating and Data Management Systems Interface," G.H. Sockut and R.P. Goldberg, Harvard University, Cambridge, MA. (8)

"An Analysis of Distributed Free Space in an Operating and Data Management Systems Environment," Y.H. Chin, Northwestern University, Evanston, IL. (6)

DISCUSSIONS

WINE AND CHEESE PARTY — 6:00 p.m.
(Room to be Announced)

TUESDAY - March 22

Session IV — 9:00 a.m.
Concurrency, Integrity and Access to Distributed Data

Chairman: P. Denning
Purdue University
West Lafayette, IN.

"A Deadlock Model for Database and Operating Systems, " S.B. Yao, Purdue University, West Lafayette, IN. (11)

Note: The number in parenthesis appearing at the end of a paper corresponds to the numbered abstract to be published in the March, 1977 issue of the Newsletter of the Technical Committee on Data Base Engineering, which will be distributed at the Workshop.

Session IV cont'd

"Extending the Concept of Abstract Data Types (for the Synchronization of Concurrent Accesses to Shared Resources in Operating Systems) to the Representation of Database Integrity Constraints and Views," H. Weber, IBM Research Lab., San Jose, CA. (9)

"Resolving Some Concurrent Update Problems without Looking," P. Bernstein, C.H. Papadimitriou, and J.B. Rothnie, Harvard Univ., Cambridge, MA. (12)

"Intelligent Coupling of the User to Distributed Database Systems," S.K. Chang and B.H. McCormick, University of Illinois, Chicago, IL. (14)

DISCUSSIONS

COFFEE BREAK 11:00 a.m.

Session V 11:30 a.m.

Special Topics

Chairman: C. R. Carlson
Northwestern University

"On the Retrieval Time and Storage Space of Doubly-Chained Multiple Attribute-Free Data Base Organization," J.M. Chang, S. B. Yao, and K.S. Fu, Purdue Univ., West Lafayette, IN.

"The Design of Cryptography-Based Secure File Systems," E. Gudes, Penn State Univ., College Park, PA.

DISCUSSIONS

Session VI 2:00 p.m.

Round Table Discussions

Chairman: David K. Hsiao
Ohio State University
Columbus, Ohio

Panelists:

B. Berra, Syracuse University
P. Denning, Purdue University
M. Edelberg, Sperry Research Center
J. Fry, University of Michigan
J. Liu, Univ. of Illinois, Urbana
J. Mehl, IBM Research Lab.

WORKSHOP COMMITTEE

General Chairman: Professor Stephen S. Yau
Northwestern University

Workshop Co-Chairmen: Professor David K. Hsiao
Ohio State University

Professor Richard Muntz
UCLA

PROGRAM COMMITTEE

Professor P. Bruce Berra Syracuse University	Professor Mani Chandy University of Texas	Professor Peter Chen Mass. Inst. of Technology
Dr. Murray Edelberg Sperry Research Center	Professor James Fray University of Michigan	Professor R. S. Gaines The Rand Corporation
Professor H. R. Hartson Virginia Polytechnic Inst. & State University	Dr. Frank King IBM Research Laboratory	Dr. E. J. McCauley, III Aeronutronic Ford
Professor Peter Denning Purdue University	Professor G. Popek UCLA	

Treasurer and Local Arrangements: Professor Robert Carlson, Northwestern University

1. THE USE OF REFERENCE MONITOR TECHNIQUE FOR DATA SECURITY IN COMPUTER SYSTEMS

Michael Grohn and Gillian Kirkby
I.P. Sharp Associates Ltd.
Suite 600
265, Carling Avenue
Ottawa, K1S 2E1 CANADA

Data security in computer systems is recognized by both industry and the military as a major problem area. A security breach may result in heavy financial loss or disclosure of national secrets, either of which is deemed high undesirable. Unfortunately, the haphazard dispersion of security mechanisms, in current computer technology, makes it impossible to be certain that a system really is secure.

Our approach in addressing this problem chooses the reference monitor technique, explicitly to centralize all security related functions. The definition involved a three stage process, progressing from abstractions to realizations.

A mathematical model was constructed to embody the salient features of secure data management. This led to the 'Parnas-like' functional specification of an adequate DMS design tool. The third step was to utilize validation techniques to certify that the functional specifications maintained security. These specifications are expected to facilitate actual implementations of a secure DMS.

The relational approach to data management was adopted, because its simplicity and theoretical basis allowed concentration on security considerations. The functional specification activity indicated those attributes required of a host operating system. The net result of our work is the conclusion that implementation of a provably secure data management system is possible.

2. CONVEYANCE OF OPERATING AND DATA MANAGEMENT SYSTEMS

William D. Haseman
Carnegie-Mellon University
Pittsburgh, PA 15213

Recent advances in the area of very large data bases and in the area of non-procedural query languages to access those data bases is beginning to demonstrate that the structure and functions of a data base management system and an operating system are rapidly becoming the same. The objective of this paper is to argue because of those similarities, the future will see operating systems replaced by data base management systems, or vice versa depending on how you view the transition.

The approach which will be used to show this convergence will be the parallel discussion of the GPLAN system and a typical operating system. The GPLAN system includes a large scale data base management system, an English like non-procedural query language, and the capability to use a number of models or application programs stored within its program library. An effort will be made to show how many of the problems encountered in the design of this system are very similar to those problems involved in operating systems, particularly when multiple users are permitted. These include, to name a few, issues related to allocation of scarce resources, concerns about deadlock, security and integrity, task scheduling, memory paging, and control of data transfers.

3. SIMILARITIES AND DIFFERENCES BETWEEN OPERATING SYSTEM AND DATA MANAGEMENT SECURITY - A STUDY TOWARD A KERNAL DESIGN OF DATABASE SECURITY SOFTWARE

Deborah Downs and Gerald J. Popek
Computer Science Department
UCLA
Los Angeles, CA

Data management security has become an important part of data base design. To clarify the issues involved in providing this security, the similarities and differences between operating system and data management security are examined and several proposed methods of achieving security in data bases are discussed. The importance of reliable enforcement by the data base security software is noted and an outline of a kernel design is proposed as a possible solution.

4. SYSTEM R AND ITS OPERATING SYSTEM ENVIRONMENT

Michael W. Blasgen
IBM Research Laboratory
San Jose, CA

The relationship between System R, an experimental data management system, and its operating system is explored. The problems of locking, recovery, and storage management are discussed, and their implementations described. The need for efficient low level primitives from the operating system is emphasized.

5. PERFORMANCE EVALUATION OF GENERALIZED MANAGEMENT INFORMATION SYSTEM ON A VIRTUAL MEMORY OPERATING AND COMPUTER SYSTEM - A CASE STUDY OF GMIS ON VM/370

Luis E. S. Sarmento and Peter P. S. Chen
E53-329, Center for Information Systems Research
Sloan School of Management
M.I.T.
Cambridge, MA 02139

Using the virtual machine capability of VM/370, GMIS (Generalized management Information System) provides an unconventional way to integrate a database system and several modeling programs. This paper analyzes the GMIS system architecture and identifies ways to improve the system performance.

Both hardware upgrade and software changes are proposed and evaluated. To evaluate whether a hardware upgrade is necessary, a set of measurement experiments are performed on the computer system (an IBM 370/158) to find out the extent of resources utilization. To evaluate the software changes (such as improving the interface between the database system and the O.S.), a simulation model is developed. In the simulation experiments, some "surprising" results are found and analyzed. Finally, recommendations on how to improve the system performance are made.

6. AN ANALYSIS OF DISTRIBUTED FREE SPACE IN AN OPERATING AND DATA MANAGEMENT SYSTEMS ENVIRONMENT

Y. H. Chin
Department of Computer Sciences
Northwestern University
Evanston, IL 60201

Due to various application requirements, many generalized data base management systems (GDBMS) provide not only efficient facilities for the creation and storage of users' data, but also provide efficient system optimization facilities for maintaining and automatically optimizing the performance of a created data base. Technological developments in storage devices have led to larger storage capacity, faster access-time, and cheaper storage cost. Therefore, the conventional response-time criterion for measuring system performance becomes more important than the space utilization (i.e., secondary storage space) criterion.

In order to maintain fast response time, many generalized data base management systems set aside an additional free storage space within secondary storage. This Data Storage Area (DSA) is allocated for the storing of data records. By providing this "distributed" free space within each DSA, subsequent updating operations (e.g., inserting a new record or increasing the length of an existing record) do not require overflow techniques; therefore, from the viewpoint of response time the desired system performance can be maintained. In this paper, the size

of "distributed" free-space in a DSA which should be claimed at load time is represented as a function of two variables, namely: (1) the insertion ratio of the file subsequent to the load time, and (2) the number of records in the data storage area (DSA) at load time. Using these two variables as known parameters, the size of "distributed" free space can be determined and reserved, and we can determine the size of the DSA's. This is used to determine the number of pages required for a file at load time. Based on a probability function, a generalized model which represents the "appropriate" DSA size has been obtained with respect to various Access Methods available today. Although the motivation of the paper is to present a general mathematical model, few tests have been done to study the analytical results.

7. THE INFLUENCE OF THE OPERATING SYSTEM IN THE DESIGN AND IMPLEMENTATION OF TRIGGER SUBSYSTEM FOR DATABASE SYSTEM

Kapali P. Eswaran
IBM Research Laboratory
San Joe, CA 95193

This paper considers the specifications and design of a trigger subsystem in a database management system. The use of triggers as extended assertions, as a means to materialize virtual data objects, and as a tool in enforcing authorization specifications is discussed. The functional requirements of a trigger subsystem and different implementation issues are studied. We also examine the interactions between a trigger subsystem and the rest of the database system, in particular the authorization and locking subsystems. We also point out that to meet the general requirements of a database system, two kinds of triggers are needed. A short discussion on the philosophies of subroutines and triggers in database systems is also included.

8. DATABASE REORGANIZATION ISSUES RELATED TO OPERATING AND DATA MANAGEMENT SYSTEMS INTERFACE

Gary H. Sockut and Robert P. Goldberg
Aiken Computation Laboratory
Harvard University
Cambridge, MA 02138

Reorganizing a data base concurrently with usage is presented as an alternative to the conventional strategy of keeping a data base offline during reorganization. Offline reorganization could require an intolerably long time for a very large data base or for an essential computer utility, which is to be available 24 hours per day. Examples and purposes of reorganization are described. During concurrent reorganization, users have full access (including update) to the entire data base, with only a brief wait if a portion to be accessed is being reorganized. Thus locking must be of small granularity, or reorganization must release locks frequently. Data independence is required; a level of the data base is invariant while lower levels are reorganized. Requirements for con-

current reorganization include correct operation, synchronization for consistency, deadlock prevention or recovery, journalling and recovery, and efficiency. Several alternatives to full concurrent reorganization are described. Issues which should be investigated include a conceptual model, inter-level mappings, algorithms, principles, performance models, and semantics. Several of the issues that are discussed (synchronization, deadlocks, process dispatching priorities, journalling, file structure optimization, and performance models) relate to the interface between operating and data base management systems.

9. EXTENDING THE CONCEPT OF ABSTRACT DATA TYPES (FOR THE SYNCHRONIZATION OF CONCURRENT ACCESSES TO SHARED RESOURCES IN OPERATING SYSTEMS) TO THE REPRESENTATION OF DATABASE INTEGRITY CONSTRAINTS AND VIEWS

Herbert Weber
IBM Research Laboratory
San Jose, CA 95193

An abstract data type consists of a collection of data and a set of procedures associated with the type. Only those procedures which are local to the type have access to the data of the type. A program may operate on those data through the invocation of the procedures of this type.

Abstract data types have been introduced to model and implement the synchronization of accesses to shared resources in Operating Systems. One frequently used notation of an abstract data type for this purpose is called the monitor concept. A monitor provides the primitive operations "wait" and "signal" which share a variable denoting the characteristics of a resource in a computer system. With these operations one may delay or resume programs in order to organize a conflict free use of the resource.

One may also consider data items in a data base as shared among various concurrent transactions against the data base. The monitor concept may then serve as a means to organize a conflict free interrogation and manipulation of the data base.

The incapsulation of the data in the data base by all the operations applicable to them is beneficial for other purposes as well:

(1) The creation of a data base requires the definition of data structures, integrity assertions, and different user views. Current data base concepts provide therefore three basically different notations to describe the content of the data base. It is shown here to be possible that an extended abstract data type concept may serve as a uniform notation.

(2) The definition of data structures, integrity constraints, and views usually results in a very complex structure of the data base. The structure of programs which interrograte and manipulate a complex data base are then complex themselves, and therefore difficult to design, implement, and maintain. The abstract data type concept offers means for a structured and modular design and helps therefore to simplify these tasks.

For this purpose we allow the definition of abstract data types which characterize data entities and logical dependencies among data entities as well. In order to model arbitrary logical dependencies we propose the following extension of the concept: Abstract data types may be represented in terms of other more primitive abstract data types. The resulting hierarchical composition schema for abstract data types will be called D-graph.

We show how one may represent data structures, integrity constraints and views by means of D-graphs. We demonstrate also how users may manipulate the data base via various user views and provide therefore a schema for the proper communication of different users by means of data shared among different views.

10. THE REQUIREMENT FOR A SECURE USER INTERFACE TO DATABASE AND OPERATING SYSTEMS

Stanley R. Ames, Jr.
The MITRE Corporation
Bedford, MA

The requirements for multilevel security have a great impact on the formulation of a user interface, especially in a transaction-oriented data base management system. In this report, we discuss several tradeoffs between the requirements for multilevel security and the requirements for a facile user interface.

We have reached several conclusions that impact the requirements for a usably secure user interface. Among these are the realization that: object size determines the types of features that can easily be supported; a limited type of write down may well be needed to ensure user acceptance; and unless process control is handled by hardware, the security requirements for multiple processes may impact performance to an unacceptable degree. Among the problems that require additional effort are: the best way to securely input commands and identifications of tradeoffs between verifying a portion of code and improved user interfaces.

The first implementation of the message systems will be completed in January. These systems will include secure behavior at the user interface. In addition, each system will provide a detailed design for implementing the system in a verifiably secure fashion. This design will be implemented by July of 77, giving us a conceptually secure system. With these operational systems, we will be in a better position to answer questions regarding the impact security has on the user interface.

11. A DEADLOCK MODEL FOR DATA BASE AND OPERATING SYSTEMS

S. B. Yao
Math Science Building
Purdue University
West Lafayette, IN 47907

The problem of locks and deadlocks in database systems are reviewed and compared. A probabilistic model for deadlocks in database and operating systems is presented and a database deadlock detection algorithm is introduced.

12. RESOLVING SOME CONCURRENT UPDATE PROBLEMS WITHOUT LOCKING

Philip Bernstein, C. H. Papadimitriou, J. B. Rothnie
Aiken Computation Laboratory
Harvard University
Cambridge, MA 02138

When many users retrieve from and update a single data base, their interleaved transactions may result in states of the database that are not reachable by any ordinary (i.e., serial) sequence of transactions. The usual approach to this problem has been to embody a locking mechanism in the database system to prevent arbitrary interleaving of transactions on those database items that are being updated. Unfortunately, locking operations can be quite time consuming and may force some processes to idle for significant time periods until it is safe for them to proceed with their transactions. However, not all update transactions require locking the database items upon which they operate. In this paper we propose a mathematical model for analyzing what kinds of transactions can dispense with locking.

We assume that the set of transactions $T = \{T_1,\ldots,T_n\}$ are defined statically as follows: Each transaction T_i consists of a read operation R_i, which (instantaneously) extracts some information from the database into a local work area, followed later by a write operation W_i, which (instantaneously) inserts, deletes and/or modifies some database items. The subset of the database that R_i reads, denoted $S(R_i)$, and that W_i writes, denoted $S(W_i)$, are also known. (The granularity of the subset is not important for the results that follow.) Since the exact nature of the computation performed by T_i is not easy to determine a priori, we assume that T_i is a set of functions, each mapping the set of possible values of elements of $S(R_i)$ to the set of possible values of a particular element of $S(W_i)$, and open to arbitrary interpretations (in the program schema theoretic sense).

A <u>history</u> H of a set of transactions T is a permutation of the symbols R_j and W_j in which each R_j precedes the corresponding W_j. A <u>serial</u> history is one in which each R_j <u>immediately</u> precedes the corresponding W_j. Intuitively, two histories are <u>equivalent</u> if for any given initial state of the database, the two map the database into the same final state, independent of the exact nature of the computation performed by each transaction.

Depending upon the structure of the sets $S(R_j)$ and $S(W_j)$, there are histories that are not equivalent to any serial history. An example is $H = \langle R_1,R_2,W_2,W_1\rangle$ where $S(W_1) = S(R_2) = \{X\}$ and $S(W_2) = S(R_1) = \{Y\}$. In this case, we say that H is <u>not serially reproducible</u>. We only want to allow consistent database states - ones that result from serially reproducible histories. A set of transactions T is <u>safe</u> if <u>all</u> histories of T are serially reproducible. A history of a safe set of transactions always yields a consistent database state, without synchronizing reads and writes. We now informally outline some preliminary results regarding this model.

A transaction $T_i = [R_i,W_i]$ is live in history H if at least one value of $S(W_i)$ is read by another transaction before it is overwritten.

Theorem 1 Two histories are equivalent if they contain the same set of transactions and each live read, R_i, reads each element of $S(R_i)$ from the same write operations in both histories. □
We say that a digraph $G = (T,E)$ with node set $T = \{T_1,\ldots,T_n\}$ is compatible with a history H iff
For all T_1, T_2 for which R_2 reads some x from $S(W_1)$ in H we have
i. $(T_1,T_2) \in E$
ii. For all $T_3 \in T$ such that $x \in S(W_3)$, exactly one of (T_3,T_1), (T_2,T_3) is in E.

Theorem 2 A history H is serially reproducible iff there is an acyclic digraph G compatible with H. □
However, the necessary and sufficient condition of Theorem 2 does not seem to suggest an efficient test for serial reproducibility:
Theorem 3 Determining whether a given history H is serially reproducible is NP-complete (i.e., probably requires exponential time). □
For a set of transactions T, define a graph $G(T) = (V,E)$ as $V=\{R_j,W_j:\text{all } j\}$ and $E = \{(R_j,W_j):\text{all } j\} \cup \{(R_i,W_j)\colon S(R_i) \cap S(W_j) \neq \emptyset\} \cup \{(W_i,W_j)\colon S(W_i) \cap S(W_j) \neq \emptyset\}$.

Theorem 4 Let T be a set of transactions such that for each $T_i \in T$, $S(W_i)$ has an element not present in any other $S(W_i)$, $T_j \in T$. Then T is not safe iff G(T) has a cycle. Furthermore, the presence of a cycle in G(T) can be determined in linear time. □
Theorem 4 provides a method for determining safety of a certain class of transaction sets. Interesting practical examples of these transaction sets have been isolated. We conjecture that the general problem of determining safety is not computationally feasible.

13. USER, OPERATING SYSTEM, DATABASE MANAGEMENT SYSTEM: GLOBAL STRUCTURE AND FUNCTIONAL DEPENDENCIES

Ken J. McDonell
Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2H1 Canada

The hardware architecture of secondary storage subsystems is rapidly changing. Technological advancements have heralded the introduction of input-output processors with significant autonomous processing potential, and the requirements of non-numeric processing are being recognized with the development of specially tailored storage modules and access techniques. These hardware changes within the input-output subsystem will seriously impact both the internal structure and the external interfaces of the database management system.

Assuming these preconditions, a justification will be presented for replacing the current multiple input-output interfaces between the secondary storage devices and the software modules (i.e. user processes, service utilities and operating system routines) with a homogeneous interface to the database management system. In fact, the enforcement of a single, high level input-output interface to secondary storage for all central-processor-resident software appears to result in significant advantages -- specifically in the areas of software size, reliability and maintainability, system structure, security and integrity, and efficient resource utilization.

14. INTELLIGENT COUPLING OF THE USER TO DISTRIBUTED DATABASE SYSTEMS

Shi-Kuo Chang and B. H. McCormick
Medical Information Systems Laboratory
University of Illinois at Chicago Circle
Chicago, IL

Recent advances in communication technology have made the design of sophisticated computer networks feasible. One of the main advantages of a computer network is the sharing and exchange of information and resources among a large number of users. Instead of a centralized database, with computer networking distributed database systems have become a reality.

However, the complexity of a distributed database system makes it difficult for the casual end user to effectively interact with the system. He will have difficulty in locating information, formulate queries, and obtain access permission. In this paper, the concept of an intelligent coupler is introduced. The intelligent coupler serves as the intermediary between the end user and the distributed database system. The intelligent coupler listens to the user's retrieval requests; helps the user formulate his retrieval requests correctly; efficiently translates user's retrieval requests into a network-compatible retrieval command language; and obtains authorization from the system for data retrieval and/or update. In short, the intelligent coupler serves as the user's surrogate in a distributed database system. However from the user's viewpoint, the intelligent coupler also serves as the system's surrogate.

THE RELATIONSHIP BETWEEN OPERATING SYSTEM AND DATABASE SECURITY: A SURVEY

E. B. Fernandez and C. Wood
IBM Corp., Los Angeles Scientific Center
9045 Lincoln Blvd.
Los Angeles, CA 90045

This paper discusses some policies that have been used or proposed for data security in operating and database management systems. Some mechanisms to implement these policies are discussed also. A survey of systems and the approaches to interrelate OS & DBMS is given.

TC ON DATA BASE ENGINEERING

MEMBERSHIP LIST

Russell J. Abbott
Department of Computer Science
California State University
Northridge, CA 91324

Larry V. Allen
MD. C-73
Honeywell Information Systems
P. O. Box 6000
Phoenix, AZ 85005

D. Au
PCS, Inc.
467 Hill 23 Drive
Flint, MI 48504

Charles Bachman
Honeywell Information Systems
200 Smith Street
Waltham, MA 02154

D. Z. Badal
UCLA
Computer Science Department
3436 Boelter Hall
Los Angeles, CA 90024

J. L. Baer
Department of Computer Science
FR - 35
University of Washington
Seattle, WA 98195

A. James Baroody, Jr.
713 Eugenia Avenue
Madison, WI 53705

John L. Berg
National Bureau of Standards
TECH A-265
Washington, DC 20234

P. Bruce Berra
Industrial Engineering and
Operations Research
Syracuse University
441 Link Hall
Syracuse, NY 13210

M. V. Bhat
Engineering Computing
Pratt & Whitney Aircraft
of Canada
P. O. Box 10
Longueuil, Quebec, CANADA

Robert Carlson
Computer Sciences Department
Northwestern University
Evanston, IL 60201

Cheng-Wen Cheng
Room 4L-409
Bell Laboratories
Warrenville Road
Naperville, IL 60540

L. Cheung
Marquette University
1515 W. Wisconsin Avenue
Milwaukee, WI 53233

Y. H. Chin
Institute of Applied Mathematics
National Tsing-Hua University
Hsinchu, Taiwan
REPUBLIC OF CHINA

Yaohan Chu
Department of Computer Science
University of Maryland
College Park, MD 20742

Henry Y. H. Chuang
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260

Billy G. Claybrook
Computer Science Department
VPT and State University
Blacksburg, VA 24061

E. F. Codd
IBM Research Laboratory
Monterey and Cottle Roads
San Jose, CA 95193

John Czelen
3550 Whitehaven Parkway, N.W.
Washington, DC 20007

Lorraine Duvall
IIT Research Institute
Box 1355, Branch P. O.
Rome, NY 13440

Murray Edelberg
Sperry Research Center
100 North Road
Sudbury, MA 01776

Roger W. Elliott
Dept. of Industrial Engineering
Texas A & M University
College Station, TX 77843

Ronald L. Enfield
1727 South Charlotte Avenue
San Gabriel, CA 91776

D. Ferrari
CS Division, EECS Department
University of California
Berkeley, CA 94720

Edward Feustel
Rice University
P. O. Box 1892
Houston, TX 77001

M. Frame
First Data Corporation
2011 Eye Street N.W.
Washinton, DC 20008

K. S. Fu
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

R. Stockton Gaines
Information Science Department
The Rand Corporation
1700 Main Street
Santa Monica, CA 90406

Edward M. Gawlinski
Environmental Protection Agency
26 Federal Plaza, Room 1642
New York, NY 10007

S. P. Ghosh
IBM Research Laboratory
Monterey and Cottle Roads
San Jose, CA 95193

E. Howard Green Jr.
28 Southview Terrace
San Anselmo, CA 94960

J. M. Grochow
C/o A.M.S.
1515 Wilson Blvd.
Arlington, VA 22209

Leo H. Groner
B/330-75 D/78F
IBM E. Fishkill
Hopewell Jct., NY 12533

Kevin Gross
200 Washington Street
Troy, NY 12180

Michael Hammer
Dept. of Electrical Engineering
M.I.T.
Cambridge, MA 02139

Martin Hellman
Dept. of Electrical Engineering
Stanford University
Stanford, CA 94305

Gene F. Hoffnagle
6900 Keats Court
Rockville, MD 20855

William L. Honig
Bell Labs, Room 6B327
Naperville & Warrenville Roads
Naperville, IL 60540

Grace M. Hopper
Dept. of the Navy, NPLS, OP-911F
Pentagon BD 770
Washington, DC 20350

David K. Hsiao
Dept. of Computer and Information
Science
The Ohio State University
2036 Neil Avenue Mall
Columbus, OH 43210

Larry G. Hull
Code 533.1
Goddard Space Flight Center
Greenbelt, MD 20771

Keki Irani
Dept. of Computer and
Electrical Engineering
University of Michigan
Ann Arbor, MI 48104

David K. Jefferson
David W. Taylor Naval Ship
R & D Center
Code 188A
Bethesda, MD 20084

Douglas S. Kerr
Dept. of Computer and Information
Science
The Ohio State University
2036 Neil Avenue Mall
Columbus, OH 43210

Richard B. Kieburtz
Dept. of Computer Science
State University of New York
Stony Brook, NY 11794

T. L. Kunii
Information Sciences Laboratories
University of Tokyo
Tokyo, 113, JAPAN

Yutaka Kuwahara
Hitachi Central Research Lab
2672 Bayshore Frontage Road #703
Mountain View, CA 94043

Larry Kuzma
HQ. AFSC SMR 6
Andrews AFB, MD 20334

Edward Y. S. Lee
Jet Propulsion Laboratory
MS-168-534
4800 Oak Grove Drive
Pasadena, CA 91103

Ester K. C. Lee
Jet Propulsion Laboratory
171-266
4800 Oak Grove Drive
Pasadena, CA 91103

David Lefkowitz
The Moore School of
Electrical Engineering
The University of Pennsylvania
Philadelphia, PA 19104

Philippe G. H. Lehot
Philippe LEHOT Associates
976 Longridge Road
Oakland, CA 97610

Edward Levinson
Room 2A214
Bell Laboratories
Whippany Road
Whippany, NJ 07981

Richard G. Luebke
1841 St. Andrews Plaza
San Jose, CA 95132

Jane W. S. Liu
Department of Computer Science
University of Illinois
Urbana, IL 61801

Vincent Lum
IBM Research Laboratory
Monterey and Cottle Roads
San Jose, CA 95193

Stuart E. Madnick
Sloan School of Management
M.I.T.
Cambridge, MA 02139

Frank Manola
Communications Sciences Division
Navel Research Laboratory
Washington, DC 20375

Giacomo Marini
IBM Scientific Center
Dorsoduro 3228
30123 Venezia, ITALY

E. J. McCauley
Aeronutronic Ford Corp., X40
393 Fabian Way
Palo Alto, CA 94303

John K. McCandliss
12164 Wensley Road
Florissant, MO 63033

Jim Mehl
P. O. Box 632
Los Gatos, CA 95030

J. Misra
Computer Science Department
University of Texas
Austin, TX 78712

L. I. Morganstein
E. I. Dupont Company
Engineering Department
Louviers Building
Wilmington, DE 19898

Peter A. Ng
Computer Science Program
Department of Mathematics
Hunte College of CUNY
695 Park Avenue
New York, NY 10021

Sylvia Osborn
Department of Computer Science
University of Waterloo
Waterloo, Ontario
CANADA N2L 3G1

David Pessel
Electrical Engineering Dept.
University of Rochester
Rochester, NY 14627

Charles T. Pierce
Attn: DRSEL-PA-S
Product Assurance Dir.
Ft. Monmouth, NY 07703

Noah Prywes
122 Moore D2
University of Pennsylvania
Philadelphia, PA 19174

B. H. Sams
RCA Laboratories
Princeton, NY 08540

N. F. Schneidewind
Code 5555
Naval Postgraduate School
Monterey, CA 93940

Harold Schwenk
BGS Systems, Inc.
P. O. Box 128
Lincoln, MA 01773

Michael E. Senko
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

Gerard P. Shabe
C/o CCTC/WAD
11440 Isacc Newton Square
Reston, VA 22090

Michael D. Shealy
1924 Woodmont Drive
Richmond, VA 23235

Martin L. Shooman
Polytechnic Institute of
New York
Long Island Center
Route 110
Farmingdale, NY 11735

A. Shoshani
System Development Corporation
2500 Colorado Avenue
Santa Monica, CA 90406

Diane C. P. Smith
Computer Science Department
University of Utah
Salt Lake City, UT 84112

Norman Sondak
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609

Donald H. Springer
1482 Platt Avenue
Milpitas, CA 95035

Michael Stonebraker
EECS Department
549 Evans Hall
University of California
Berkeley, CA 94720

T. C. Ting
School of Information and
  Computer Science
Georgia Institute of Technology
Atlanta, GA 30332

M. Tsuchiya
Dept. of Electrical Engineering
University of Hawaii at Manoa
2540 Dole Street
Honolulu, HI 96822

Anthony I. Wasserman
26 Malta Drive
San Francisco, CA 94131

Leonard H. Weiner
Computer Science Department
Michigan State University
400 Computer Center
East Lansing, MI 48824

Geo Wiederhold
155 Marine Road
Woodside, CA 94062

August Martin Wildberger
15811 Pinecroft Lane
Bowie, MD 20716

Paul A. Willis
TELEDYNE GEOTECH
314 Montgomery Street
P. O. Box 334
Alexandria, VA 22319

Eugene Wong
Dept of Electrical Engineering
  and Computer Science
University of California
Berkeley, CA 94720

S. B. Yao
Department of Computer Science
Purdue University
West Lafayette, IN 47909

Stephen S. Yau
Computer Science Department
Northwestern University
Evanston, IL 60201

Raymond T. Yeh
Computer Science Department
Painter 3.26
University of Texas at Austin
Austin, TX 78712

A. W. Yonda
12 Sunset Drive
Medway, MA 02053

Carlo A. Zaniolo
Sperry Research Center
100 North Road
Sudbury, MA 01776

EDITOR'S NOTES: This is the first issue of Data Base Engineering Bulletin, a quarterly publication of the IEEE Computer Society Technical Committee on Data Base Engineering. We hope that the Bulletin will serve as an effective communication channel for the expedient dissemination of news and technical ideas in the area of Data Base Engineering. I seek your advice on all editorial aspects of the Bulletin. Your contributions, suggestions, and criticisms will be greatly appreciated.

-- Jane W. S. Liu

MEETINGS OF INTEREST

◆ Third Workshop On Computer Architecture For Non-Numeric Processing
May 17-18, 1977
Syracuse, New York

Sponsors: ACM SIGIR, SIGARCH, and SIGMOD

Conference Chairman:	P. Bruce Berra Syracuse University 441 Link Hall Syracuse, NY 13210 (315) 423-2826
Program Chairman:	Stewart A. Schuster University of Toronto Toronto, Ontario, CANADA (416) 978-6026
Local Arrangement:	M. J. Fairbanks 111A Link Hall Syracuse University Syracuse, NY 13210 (315) 423-3511

◆ Third International Conference On Very Large Data Bases
October 6-8, 1977
Tokyo, Japan

Papers due:	April 15, 1977
Send 5 copies to:	Alan Merten Graduate School of Business Administration University of Michigan Ann Arbor, MI 48109
Requests for information regarding travel support:	Stuart Madnick M.I.T. Sloan School of Management 50 Memorial Drive Cambridge, MA 02139