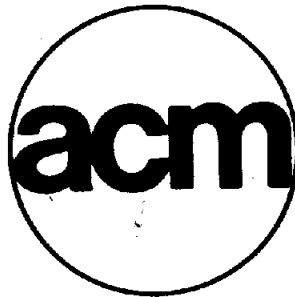




Bulletin of ACM-SIGMOD:
The Special Interest Group on Management of Data
VOLUME 7, NUMBER 3-4 1975

<i>Editor's Note</i>	1
<i>Chairman's Message</i>	2
<i>Implementation of Relational Data Base Management Systems</i>	3
<i>A Practitioner's View of Relational Data Base Theory (T. William Olle)</i>	29
<i>Database Research: Some Comments on Future Directions (W. T. Hardgrave and Ed Sibley)</i>	44
<i>Comments on The Debate: Data-Structure-Set versus Relational (John Clymer)</i>	49
<i>Review: An Introduction to Database Systems (C. J. Date)</i>	53
<i>Bibliography (Claude L. Overstreet)</i>	55
<i>Contents: Proceedings of the Workshop on Data Bases for Interactive Design</i>	60
<i>Full Self-Governing Bylaws of SIGMOD</i>	61
<i>SIGMOD 1975 Business Meeting</i>	65
<i>Washington SIGMOD</i>	68
<i>Announcement: 1976 ACM-SIGMOD Conference</i>	70
<i>Bibliography: Checkpoint, Restart, and Recovery (A.B. Tonik)</i>	72





FDT is a publication of ACM's Special Interest Group on Management of Data. It is published four times a year in New York City.

Technical Papers appearing in this issue are unrefereed working papers.

Authors contributing for publication in FDT are requested to have their copy typed single-spaced using elite (preferably), or pica type, using a well-inked or carbon ribbon.

Leave a 1-inch margin on each side and bottom and 1 1/4-inches on top. However, copy on the first page should begin 3-inches from the top; no title should be typed; it should be written on an accompanying page or in pencil on the back of the page. Please do not paginate the copy; instead, number on the back of each of the pages.

Leave two spaces between paragraphs. All technical drawings should be done in India ink. Include them directly on the page with the copy or else leave appropriate space in the copy and include them separately.

Material submitted for publication in FDT should be sent to the editor:

Randall Rustin
Chase Manhattan Bank, N.A.
1 New York Plaza 21st Floor
New York, NY 10015
(212) 676-3831

Inquiries concerning SIGMOD membership should be sent either to ACM Headquarters or to SIGMOD's Chairman:

Bernard Plagman
83-01 116th Street
Kew Gardens, NY 11418
(212) 847-1157

Additional copies at \$2 each may be obtained from:

ACM Headquarters
1133 Avenue of the Americas
New York, NY 10036
(212) 265-6300

SIGMOD BOARD OF DIRECTORS

Gene Altshuler
Stanford Research Institute
333 Ravenswood Avenue
Menlo Park, California 94025
(415) 326-6200 x3963

Charles Bachman
Honeywell Information Systems
ASTO MS 425C
200 Smith Street
Waltham, Mass. 02154

Susan Brewer (Standards Liaison)
209 West Lamar Road
Phoenix, Arizona 85013
(602) 943-2341 x728

Dr. Edgar F. Codd
IBM Research Laboratory
K51-282
Monterey and Cottle Rds.
San Jose, California 94304
(408) 227-7100

Larry Cohn
IBM Corp.
1501 California Ave.
Palo Alto, California 94304

James P. Fry
Dept. of Industrial & Operation Engineering
231 West Engineering Bldg.
University of Michigan
Ann Arbor, Michigan 48104

Robert Lindner (Secretary-Treasurer)
Data Services J2K
Federal Reserve Bank of N.Y.
33 Liberty St.
New York, N.Y. 10045
(212) 791-5168

Don Moehrke
AO Smith Corp.
Data Systems Division
Milwaukee, Wisconsin 53201
(414) 873-3000

T. William Ollie
27 Blackwood Close
West Byfleet
Surrey, KT14 6PP
England

Eugene Raichelson
MITRE Corporation
Westgate Research Park
McLean, Virginia 22101
(703) 893-3500 x2323

Kendall Wright (Past Chairman)
Dept. J91/036
IBM Corp.
1501 California Ave.
Palo Alto, California 94304

EDITOR'S NOTE: This is the last issue of FDT that will be presented to the readers of SIGMOD under my editorial auspices. The chief Editor for the next two volumes will be:

Douglas S. Kerr (614) 422-1519
Department of Computer and Information Science
2036 Neil Avenue Mall
Ohio State University
Columbus, Ohio 43210

There will also be a European editor:

G. M. Nijssen 02/12.31.50 Telex 25696
Control Data Europe Inc.
Rue De La Loi - 15 - Wetstraat
1040 Brussels - Belgium

Many SIGMOD members are not aware of the fact that the second volume of the 1974 SIGMOD Workshop on Data Description Access and Control, Data Models: Data-Structure-Set versus Relational appeared last fall. Much of the material in this newsletter relates to the material in that debate.

There are two short commentaries on the debate, one by W. T. Hardgrave and Ed Sibley and another by John Clymer that might interest some of the FDT readers who did not attend the Workshop and who have not seen the Proceedings.

The paper by Bill Olle deals with some relationships between the DBTG and Relational Views.

The transcript of the 1975 NCC panel discussion "Implementation of Relational Data Base Management Systems" appears through the courtesy of AFIPS and the session chairman Ted Codd.

I believe it will be of interest to many readers to know that the next issue of FDT, which will appear within a month of this issue, will be "The End User Facility Task Group Progress Report". The price of that issue will be stated on the issue. Unfortunately - I must add in passing - the correct price for the last issue, the "ANSI/X3/SPARC, Study Group on Data Base Management Systems, Interim Report, 75-02-08" is \$7.50.

In concluding this note, and my tenure, I wish to thank all the contributors to this, and other issues of FDT. I consider it more difficult than it might seem to be, (to those who have never done so), to contribute work for publication in the newsletter of one's professional affiliation. My thanks.

Randall Rustin

CHAIRMAN'S MESSAGE

It is not often that I take the liberty of using FDT Newsletter space. However, having served SIGMOD for a two-year term, I would like to say a few non-technical words in departing.

It has been a hectic period, but a personally very rewarding one. Most worthwhile of all to me has been working with the people I have been fortunate enough to have as collaborators. In particular, I want to thank some of those who have helped SIGMOD to be where it is today: Randall Rustin, for his work as editor of FDT and his work as program chairman of the 1974 SIGMOD Workshop and editor of the Proceedings; Jim Fry for his work in the 1974 and 1975 Workshops; Ed Sibley for all he has done, beyond and including the current '76 conference; and last, and certainly not least, SIGMOD's Board of Directors, who served collectively as a staunch panel of advisors at times when I required firm counselling.

I also wish to take this opportunity to urge all of our members to participate and help Jim Fry our new chairman in making SIGMOD the data base organization of the EDP community. My personal best wishes to Jim and his new Administration.

Bernard Plagman

IMPLEMENTATION
OF
RELATIONAL DATA BASE MANAGEMENT SYSTEMS

At the 1975 National Computer Conference held in Anaheim, California (May 19-22) there were several sessions dealing with relational data base management (RDB for short). The first of these sessions consisted of a tutorial on RDB followed by a panel discussion on the major problems in implementing RDB systems. This report is a transcription of the panel discussion, edited in minor details (mostly syntactic) to make it more readable. The original tape recording is available in the form of a cassette (identified as NCC-37) from:

Convention Seminar Cassettes
13356 Sherman Way
North Hollywood, Calif. 91605

The tutorial on relational data base management was presented by C. J. Date, IBM Development Lab., Palo Alto, California. To understand the panel discussion the reader should be acquainted with the material presented by Mr. Date. Much of this material can be found in his recently published book, "An Introduction to Database Systems" (Addison Wesley).

The panel members were:

Morton M. Astrahan, IBM Research Lab., San Jose, California
Jay Goldman, M.I.T. Computing Center, Cambridge, Mass.
Dale Jordan, Tymshare Inc., Cupertino, California
Dennis McLeod, M.I.T. Project MAC, Cambridge, Mass.
Michael Stonebraker, Univ. of Calif., Berkeley, California
Stephen J. P. Todd, IBM Scientific Centre, Peterlee, England
Dionysios Tsichritzis, Univ. of Toronto, Toronto, Canada

A cautionary note is perhaps appropriate here: generally speaking, present implementations of relational data base systems have been tested on data bases of only a few million bytes. It remains to be seen under what circumstances their performance will be competitive with that of non-relational systems on data bases of a few billion bytes.

We are grateful to the American Federation of Information Processing Societies for permission to publish this transcript.

E. F. Codd
Session Chairman

CODD

This panel discussion involves seven people who have actually participated in the implementation of seven distinct relational systems -- either experimental or production systems. There do exist relational implementations other than those represented here, and I apologize to those people for their non-inclusion in the panel. Now, there is a change with respect to the list of participants. Stephen Todd, at my extreme right, replaces Garth Notley. This is not a last minute replacement by the way; Dr. Todd represents the same system that Dr. Notley was due to represent -- namely, the system developed at the IBM Scientific Centre, Peterlee, England.

In this panel discussion we shall have two rounds. In the first round each panel member will be introduced, and he will give the essential, bare facts about his implementation. In the second round there will be a discussion of problems that people have found in implementing these systems and discussion of the performance question: Is there any necessary loss of performance if performance-oriented access paths are known to the system but not to the application programmer?

Prior to this conference the panel members corresponded with each other on these topics. Last night we held a meeting to try to work out how to divide up this rather large topic so as to cover it in the time allowed. Where it is necessary to make comparisons, comparisons will be made with CODASYL DBTG, simply because it is one of the best known proposals and systems.

Let us start then with Dr. Stephen Todd of IBM, Peterlee.

TODD

Our system is called the Peterlee Relational Test Vehicle or PRTV. It has been running for about a year now, and the theoretical size of relations we can deal with is very, very large; I don't know what it is exactly. In the actual running we have been doing, we have had up to 50 or even 100 relations of anything up to 60,000 tuples in a single relation, a data base of about 50 megabytes. The data base came to us from the Greater London Council in the United Kingdom. It isn't a huge data base, of course, but by relational standards as far as we know it is about an order magnitude bigger than any other relational data base. The system is written in PL/1 using a streamlined algebraic relational interface and other special interfaces for application programs to run on. The main point we have been trying to make for the system is that it is not a query system; instead, it is the basis of an application system. It is designed to be extensible. We stress the problems of making relational data bases efficient and finding out and

checking a genuine algebraic relational system rather than an approximate one. We have been looking into the mathematics behind relations.

CODD

The next panel member is Dr. Morton Astrahan of IBM Research, San Jose.

ASTRAHAN

The SEQUEL system is a prototype system implementing a structured English query language about which more will be said in the sessions on relational implementations and query languages [see NCC 75 Proceedings]. It is a single-user system written in PL/1, running in a virtual machine on VM 370. The underlying data base system or access method is XRAM or Extended Relational Access Method developed at the IBM Scientific Center, Cambridge, Massachusetts. The largest data base that we have worked with so far is 100,000 bytes. And, as is typical in such cases, we are now working on an improved system rather than pursuing further development of the initial prototype.

CODD

The next speaker is Prof. Dionysios Tsichritzis, University of Toronto, Toronto, Canada.

TSICHRITZIS

The ZETA system is based on the operating system OS and uses the access method EDAM and the implementation language PL/1. The system can be accessed using three alternative interfaces: one using PL/1 as host language and a preprocessor; another using a query system with an English-like language, very similar to SEQUEL; and third, by interfacing to an artificial intelligence oriented system TORUS which can manipulate natural language.

CODD

The next speaker is Prof. Michael Stonebraker of the University of California at Berkeley.

STONEBRAKER

Our system is called INGRES, Interactive Graphics and REtrieval System. It is constructed on top of the UNIX operating system, which runs on any Digital Equipment Corp. 11/40, 11/45, or 11/70 system. It is implemented in a programming language called C, and makes use of a compiler-compiler called YACC. The two user

languages that we support are 1) a relational calculus oriented language QUEL; and 2) a casual user graphics oriented language called CUPID. It is a multi-user system supporting data types of single and double precision floating point numbers, integers of 1, 2, and 4 bytes, and fixed length character strings up to 256 bytes. It contains about a quarter million bytes of object code, and has taken about 5 man-years to develop. The largest relation possible is 32 megabytes. The system is running or being brought up and tested at five different sites.

CODD

Now we come to Mr. Dale Jordan, Tymshare Inc., Cupertino, California.

JORDAN

The Tymshare system is oriented toward medium-sized commercial applications being used on a remote access basis. It is implemented in the system programming language BLISS for the DEC System 10 with a time-shared, virtual memory operating system. Its language style is about half way between the level zero [e.g., GAMMA ZERO] and the algebraic languages mentioned by Chris Date in his tutorial. It is basically procedural, with facilities for accessing multiple relations simultaneously. It has special language features for report generation and access to external files. It supports data types of decimal numbers up to 20 digits, character strings and dates. The most complex data base tested so far consisted of about 17 relations over 120 domains and a size of about 3 megabytes.

CODD

The next speaker is Mr. Jay Goldman, MIT Computing Center, Cambridge, Massachusetts.

GOLDMAN

The system we have is called RDMS, and it has been running on the MULTICS operating system since late 1970. We are now in the process of transferring much of this work to run under VS on an IBM 168. It is almost entirely written in PL/1, and since MULTICS has a segmented virtual address space in a file system which is mapped into main memory automatically, the concept of access method almost doesn't apply. You don't have to do any I/O to access any of the data in the data base. The interfaces we allow are tuple-at-a-time access through PL/1, relational algebra through PL/1, and also command level relational algebra operations. We have approximately 8 to 10 active applications which involve a dozen different types of users. We have a number of users of the same application. These applications include

the Electrical Engineering Department for maintaining information on 500 or so faculty and staff. At certain points in time, we have had registrar information on-line involving a single relation which is approximately 150,000 words or 0.6 megabytes. That is a single relation in which the individual atomic data elements are represented by a single word. So in terms of character strings, I don't really know how big it would be. The Center for Policy Alternatives at MIT maintains a small document room containing approximately 600 documents and produces a 400-page report listing these documents from their data base.

CODD

The last speaker is Mr. Dennis McLeod, MIT Project MAC, Cambridge, Massachusetts.

McLEOD

The system I am representing is called the Relational Inquiry and Storage System or RISS, which is implemented on a PDP 11/40 running the resource time-sharing system or RSTS 11 -- both the computer and the operating system being commercially available from the Digital Equipment Corporation. It is a multi-user system, implemented in a language called BASIC PLUS and has been running now for about 9 months at Forest Hospital in Illinois. The user interfaces we provide are actually two: we have a non-procedural query-update language for use by non-programming users and by application programmers through calls from the host language programs and a tuple-at-a-time procedural level interface which allows application programmers to optimize programs if necessary. The largest data base we have tested so far is 2 megabytes.

CODD

O.K., having had this introduction to the different systems, we are now going around the panel again to discuss problems that arise in implementing these systems. In connection with the performance question, we have discussed amongst ourselves specific examples. I think it would be impractical to get into those specific examples during the time allowed, but we will try to do the best we can.

TODD

The first problem that faces anyone trying to implement a relational data base system is the choice of access methods. The mere fact that such a system is implemented on a virtual memory doesn't mean that access methods are unnecessary. An access method is a way of getting data. What basic access methods should be used? A lot of people presume that because

a file is a bit like a relation, that relational data bases are implemented on sequential files. There is a tendency for people having a system which is a sequential file system with query capability to call it a relational system. Both of these assertions are incorrect. The sequential file system with query is not necessarily relational. A relational system can work on almost any known access method. Ours works on sorted, hierarchical, indexed sequential files with secondary indices -- which is about half of the known access methods or half those I know. I have also seen systems that work very nicely on rings and hashing, inverted file techniques, and so on. Those are common techniques. I have also seen systems working on slightly less standard techniques, employing various tree mechanisms for example. So all of these can support a relational interface, and an important point is that the relational interface stays the same. So, even if within a given implementation you change the access method completely, the programs will still run. I think the commonest access method of all is a set of three processes in a multi-processor environment: you have a collator, a sorter, and a tabulator, and you call it unit record equipment. [Editor's note: It is not clear what relational interfaces for application programs and terminal activities can be supported using this "access method" unless we stretch our usual notions of support and interface.]

I think that we will know when we have made data bases work really well -- it will be when a perfect programmer produces the same program as our perfect relational data base system. And looking under the covers at the implementation, which as we said the programmer does not look at, if we took a snapshot of the machine running a relational data base, when we couldn't tell whether a hand-tailored program (written by a skilled programmer) or a general purpose relational system was being executed, then we shall have a good relational data base system. The question is which is going to be more perfect? Can we have a more perfect programmer or can we have a more perfect system? Is the programmer going to react to change more quickly or is the system? At the moment, I think, the answer is "Yes, the programmer tends to be slightly better than the system; but the system is more responsive". And we are learning more and more about making systems efficient.

ASTRAHAN

A relational access method provides performance oriented access paths (amongst other things); for example, indexes and links, where indexes are inverted attributes and links are direct pointers from a tuple in one relation to another tuple in the same or some other relation. In a relational system, the specifications for these access paths are known to the system and maintained by the system, not by the user. So there is a

cost involved for maintenance, and that is where we have two challenges: first, which access paths should we specify for the system to maintain, and second under what circumstances should we use them.

Now, the maintenance cost must be balanced against the savings involved in using the paths. One possibility is to have the user specify to the system which paths to maintain, assuming that he can predict the characteristics of his transactions and determine which of the attributes (data items) he is going to be asking questions about, and predict how much updating he is going to do. Another possibility that has been considered is having the system do the specification based upon the statistics gathered by the system on the actual use of the data base.

As for when to use the paths, the problem of finding the relevant path for some ad hoc transaction is now inside the relational system -- it is a system problem rather than a user problem. The first part -- finding those paths which are known to the system and are relevant to the transaction -- is pretty easy, because the path specifications are known to the system (since it is maintaining them). The system can look up this information in the catalog and find which paths are relevant to the characteristics of the transaction. The more complicated challenge, which we call optimization, involves calculating the relative costs of using the various paths as compared with using the most basic access path -- that is, a complete scan of all the tuples in the pertinent relations. This requires such things as knowing the size of each stored relation, how many pages it may occupy, what is the selectivity of a given index (i.e., to what degree does it partition a relation), and what are the characteristics of the predicate tree in the query language statement. It is possible to have a combination of ANDs and ORs that requires a complete scan of a relation, even though you could have used an existing index to advantage for some of the predicates. You may even want to look, in some cases, at the physical clustering rules that were used in storing a relation. Access via an index supporting a predicate is sometimes less efficient than a complete scan of the relation along a path based on the clustering, since the latter guarantees touching each page only once. In SEQUEL, indexes are the principal performance oriented access paths. As an example of their use, suppose we want to execute a join of relation R with relation S, and there is no index. In this case, the system creates an index dynamically, so as to avoid scanning relation R once for every tuple in relation S (or vice versa). And since a dynamically created index is not specified in the catalog, the system will not maintain it, but will instead throw it away. We are still working on the challenge of trying to decide when to keep such an index permanently.

TSICHRITZIS

One of the severe problems that one encounters when designing relational data base management systems is this maintenance of access paths that Morton Astrahan mentioned before. All performance oriented access paths in relational systems are formed according to well defined properties. For instance, an access path can support a join (of the kind Chris Date was discussing in his tutorial) by linking the appropriate tuples of the two relations whose join is required.

Instead of pre-defined access paths, of course, you can "cop out" and reconstruct them on the fly every time you need them. But that is hardly the thing to do, especially when you have light update activity on the relations involved, and you need the access paths very often for retrieval. So you would like to keep them around.

However, if you do keep them around, you need to keep them consistent with their definition. If a pre-defined access path represents, for instance, a join, then you have to make sure that its structure is kept consistent with the definition of the join. Now there are lots of problems when you have insertions, deletions, and updates on the relations involved in making sure that the intermediate structures you keep are consistent. The most severe problem of all is the fact that, if you make an update, it may not only affect the connection of certain tuples with some other tuples in a negative manner (namely, connections may have to be dropped), but also some new connections may have to be introduced. To resolve this, you may have to either scan one relation completely or else use some indexes for the relations and attributes involved. There are many algorithms that you can use. A very simple one, for instance, is to completely reconstruct the access path on the fly whenever there is a change in the relations involved. A more involved method entails correcting the access path when the system has time (a slack period in the system), or when a transaction needs the access path. A third method is to modify the access path and keep it consistent every time an update, insertion or deletion occurs. There are many trade-offs here and the trade-offs are similar to those you have when you keep indexes around. Namely, there is a cost to keeping indexes around, and there is a cost if you don't have them when they are needed (the reconstruction cost or the cost of exhaustive searching).

Now, non-relational systems do not have this problem, and the reason they do not have it is very simple: they just pass it on to the user. That is, the user has the responsibility of direct manipulation of these access paths. He sees the access paths, he knows what they mean, and he directly creates

connections or deletes connections. This means that the consistency of these access paths is his problem. The system doesn't worry at all about it, and if the user makes a mistake with respect to the intended meaning of an access path, that's his problem.

STCNEBRAKER

The first three speakers have given you a flavor of what goes on inside an implementation of a relational data base system. Now, in CODASYL's DBTG proposals the access paths are specifically visible to the user, and the question I have been asked to address is this: Is there any necessary loss of performance when those access paths which are visible in DBTG and other lower level systems are not visible to the user in relational systems? The consensus of the panel concerning the answer to this question is that in some applications there may be a loss of performance, and it is due at least to the following two reasons.

First, a user who has a lower level language controls a lot more of what is actually going on. As a result, he may know something which he can use to his advantage, something which he cannot know in a higher level language because it is hidden from him. This is one general thing that may go wrong with high level language systems. Another thing that may go wrong is that the user simply cannot express the things that he knows in the higher level language because of its structure, and I'd like to give you two very general examples of these problems. The first problem concerns things that are hidden from him. It may very well be that a user can express the same interaction in one of the higher level relational languages in several different ways that are equivalent with respect to user intent, even though they are semantically very different as far as the system is concerned. And it may very well be that these various statements of his problem run drastically differently in speed. Lower level systems give the user the problem of figuring out which of these different statements is going to run faster because he sees performance oriented access paths. In higher level languages, however, he usually does not see these features and has no way of figuring out which of the various ways that he could state his problem will be more efficiently executed.

The second general problem is that of being unable to express certain performance oriented concepts in higher level languages. Chris Date said earlier in his tutorial that relations are not necessarily sorted in any given order. Now, DBTG sets can be sorted in a given order and that sorting order is visible to the user. Ordering properties of underlying data may well be used by a skilled programmer to advantage, whereas in relational systems physical ordering properties are specifically hidden

from him. It is nearly impossible for one to use such features to advantage. Also, there is a concept in DBTG of a data base key, which is, in essence, the storage address of a record. It can be transmitted to the user by the system. He can later return this key back to the system and say "get me that record". Most relational systems do not give out to users the storage addresses of any records, because such addresses are considered to be the system's sole responsibility -- a data independence concern.

These things are simply not expressible in the higher languages that the relational advocates are talking about. As a result, the panel believes that there is a class of interactions where relational systems will run slower. However, there is also a very large class of interactions where a good relational system cleverly optimized will run no slower and perhaps considerably faster than a DBTG implementation. And the primary reason for that is the greatly increased flexibility (as the previous three speakers have discussed) of the access facilities under the covers. Consequently, whether or not there is a performance loss or performance gain is very application dependent. Certainly you will not get any of us to conjecture absolute numbers, but I think the proof of the pudding is going to be in the running of benchmarks on the relational systems and in the comparison of them with lower level language systems.

JORDAN

I am going to address the problem of storage requirements in a relational system. The reason this comes up is people take a look at the tables that Chris Date showed on the screen earlier and say "My God, I have been undone -you are duplicating those key values all over the place". Well, the answers we have for that can be addressed in three or four points.

First of all, we assume that data is in the data base because it is going to be used. There are lots of little tricks that you can play in systems like DBTG, where you neglect to store a data value and imply a relationship by a link. In a relational system all these things become explicit because it assumes that someone, somewhere, is going to do something explicitly with that data value and it must be made visible.

The second thing is data base design. The process of transforming a relational data base into third normal form automatically yields the property of storing one fact in one place. This means that minimal redundancy at the fact level is guaranteed [further compression by means of encoding algorithms is normally feasible]. This doesn't yet address the problem of key duplication. Keys are the primary vehicles for selecting between the relations. As far as keys go, as the

other speakers have mentioned, the access paths and storage mechanisms lying underneath the relational interfaces are no different from those used in current systems. If you wish to apply hierarchical record organization where you factor out the key values and store them only once, that is perfectly fine. The point is that this sort of thing is not visible to the user, and he is insulated from changes in the underlying structures.

The third concern is this link question. If you substitute a link for a key, which is shorter? It is not clear that you win either way in general. [Editor's note: in general, the replacement of a key by a pointer on a one-for-one basis results in a loss of associative capability].

Finally, the relational systems are natural vehicles for using standard kinds of storage compression techniques, because all these things can be selected by the data base administrator and are totally invisible to the user. In summary, there are only a few cases where it appears that relational systems would be inferior to a conventional kind of approach -- and I think most of them are controllable. The emphasis in relational systems on logical and coherent design is probably going to leave these systems with smaller storage requirements than those of more conventional systems.

GOLDMAN

Another consideration for data processing people in comparing data base systems would be what programming languages support this particular data base organization. I think it should be apparent just from the membership of this panel that I am not going to list just one language. In fact, I would like to argue that the nature of the definition of relational data systems makes them highly language independent. From Chris Date's description of the relational model, there has been no assumption and no mention of any operating system dependent feature or any procedural language dependent feature. He doesn't specifically mention the formats of any data structures, the use of pointers or linked lists or anything like that. While such structures may be used to implement access paths, they are not in the relational model. What the relational model deals with are logical entities: relations which are sets of tuples. The operations on these logical entities are procedures (very similar to subroutines). Accordingly, each operation (and only a small set of them is needed) can easily be incorporated in any procedural language. Also, implementations of relational data base systems are structured. At one level, you may have a natural language interface for the naive or casual user; at another level you may have a relational algebra interface; and at an even lower level, you may have tuple-at-a-time retrieval from relations. Since these structured systems are implemented

in higher level languages, the interfaces for high level languages are already present in the implementations.

Since the definitions of these operations on relations are independent of procedural languages and operating systems, programs written using these operations are likely to be more system independent and language independent than the languages they are actually written in. PL/1 is supposed to be a system independent language, but anyone who has had to transfer PL/1 from one machine to another (such as from a 36-bit machine to a 32-bit machine) notices some slight problems. Right now, as you can notice from this panel, there is a reasonable amount of research going on in this area, developing new ideas and new systems. The process of embedding a given relational data sublanguage in a particular procedural language using a preprocessor or something like that is not an important research goal, because the features of a relational system make that a very simple task [providing the underlying operating system can dynamically allocate and free large amounts of space].

CODD

One consequence of Jay Goldman's discussion is that you can hook up COBOL to a relational system.

McLEOD

As Michael Stonebraker and some of the other panel members have indicated, there are probably some cases in which a specifically tailored DBTG type system can outperform the more generalized and higher level relational systems, at least with respect to run time efficiency. In a somewhat analogous manner, it is sometimes possible to improve the run time performance of a program written in a high level programming language by translating it by hand into assembly language.

However, the issue is clearly deeper than it seems on the surface, and there are other important factors involved. That is to say, we may pay the price of some run time efficiency, but gain other types of advantages. The relational model of course is based on a uniform, simple, but general, data model, and this makes the relational systems easier to use for both programmers and non-programmers. The relational approach is much more oriented towards direct non-programmer use than a DBTG-type system, as well as toward the application programmers who need not concern themselves with irrelevant, low level detail. I think that it is interesting to observe the analogy here with the top-down design and levels of abstraction stressed by the structured programming advocates. In a DBTG system users are constantly plagued with the horrendous task of dealing with the low level details of the data base, and significantly, the

complexity of a DBTG-type approach may become prohibitive, especially for very large data bases. The cost of developing, understanding, maintaining, and modifying a relational data base system may well be significantly less than that for a DBTG-type system. It is interesting to note that these exact concerns - cost of development, understandability, maintainability and modifiability of the data bases -- are precisely those issues that come up when one considers the trade-offs of using a high level versus a low level programming language (such as assembly language).

The relational systems also have the advantage of a greater degree of data independence (as has been indicated by other speakers). Sublanguages for dealing with relational systems are potentially more non-procedural, as I mentioned earlier. The emphasis in the relational approach is on the use of aggregates; that is, mathematical sets of objects rather than single objects, very much in the style suggested by some of those people concerned with very high level or non-procedural programming languages.

The high level relational approach also facilitates a significantly more straightforward expression of the high level integrity and consistency constraints as well as high level specification of protection and concurrency constraints. This is primarily due to the absence in the relational data model of the highly distributed access paths of the DBTG-type approach. In the relational approach the underlying implementation is isolated from the logical data model which results in increased inter-system compatibility and, most significantly, allows the top-down approach to implementation. The fact that the relational approach encourages such a hierarchical implementation is yet another link with structured programming ideas.

In summary we observe that indeed a great deal of commonality exists between the advantages of the relational approach on the one hand and the advantages of both high level and very high level programming languages as well as structured programming techniques (reducing complexity and enforcing programmer discipline) on the other hand. Finally, there is some common ground between the relational and DBTG approaches. Perhaps one way to look at this is to consider the multi-level system, wherein the user sees a high level relational data base interface, unless optimum run-time performance is critical. If it is critical, he dives into a low level interface where he can presumably manipulate the access paths. This is somewhat analogous to the user of a high level, general purpose programming language when he resorts to assembly language, only when he finds it necessary to tune a program. Until we have really smart relational systems, it may be necessary to do some

of this kind of optimization, although I believe that in the long run very little of it may be desirable.

CODD

O.K, we can now turn to questions from the floor. I would like to permit questions not only to the panel but to Chris Date, because we have more time than I expected. So let me recognize George Dodd of General Motors first.

DODD

One thing that the speakers did not mention but which I think is very valuable in the experimental data base system RDMS which we have implemented in the General Motors Research Laboratories is the flexibility of being able to add new data relations to an already existing data base. We have had one or two instances where we had difficulties in trying to add to an already large IMS system even more data. It just couldn't be handled. The same type of data bases have worked very well when we went to the relational approach, where we could add new data items, new concepts, new relations as management requirements and government requirements made these things necessary. Secondly, we find that quite often in our engineering and analysis work, and particularly in working with rapidly changing government requirements, that the needs for new data and new analysis are coming in very fast -- faster than we can re-define our data bases in conventional systems. Although the cost of a relational system is possibly a little higher, we feel that we will more than save the design cost by being able to answer today's question today.

We have also added another thing to the system which the panel did not allude to very much: that is, an interface with a good graphics package. With this package the user can get free-form plots of data that the relational system is retrieving for him.

I have one question that I would like to ask the panel: that is, how would relational data base systems fit into the idea of handling distributed data bases where you have input from different processors across the country and are trying to distribute the data and answer questions from different sources?

CODD

Any volunteers?

TODD

The National Physical Laboratory of Great Britain is conducting research on that subject -- not because they started with

relational data bases and wanted to see how they could use them in a communications network; instead, they started with data bases spread over various areas and found that the easiest way to network them was to use relations. And I don't know anything about their work beyond the fact that they did it. So I suggest you write to the National Physical Laboratory, Teddington, London, England.

STONEBRAKER

Clearly, running data base systems on distributed data bases isn't very easy. But for sure it is no harder with relational systems than with any other system, and it is probably much easier.

CODD

Bob Balzer of the Information Sciences Institute, Marina Del Rey

BALZER

I have two questions actually. The first concerns not what the panel has talked about, which is basically implementation of relational data bases, but what they didn't address very much, which is the user interface and the actual embedding of the systems in existing languages. Should we infer from this lack of discussion that these are solved problems? Secondly, what is the impact on a relational data base of rules of inference allowing the system to infer data that is not explicitly in the data base.

CODD

Chris, would you like to have a crack at that first question?

DATE

The reason there has been little discussion of embedding relational operators in high level languages is fairly obvious. Most of the work that has been done on relational data base systems is of an experimental and first-time nature. And this is simply not being considered an interesting research problem because it is thought to be readily solvable, in contrast to some other aspects of relational systems. That is not to say that it is a totally trivial task, but I, for one, certainly don't anticipate any large problem in doing it.

TODD

There are certainly some significant problems. We have already produced papers in the Peterlee Scientific Centre on some of these problems. There is no problem in getting a tuple-at-a-time interface. However, if you are to develop a high-level user interface which permits non-relational types of operations up to the same level as the relational operations, then it is a problem -- one in which theory must play a part.

CODD

Who wants to handle the second part of the question -- concerning the addition of inferential capabilities?

Since there are no volunteers, let me tackle that. I assume you are talking about deductive inference. There is work going on in our laboratory that is aimed at integrating inferential procedures with the prototype relational data base management system SEQUEL. The method to be tried entails beating the query against the axioms to the maximum extent before accessing the data base proper. In other words, included in the data base description will be some axioms. And instead of using previous deductive inference techniques which involve beating the axioms against the data base facts right away, this new deductive inference technique, due to Chin Chang, postpones access to the data in the large data base -- for reasons of efficiency, of course.

CODD

The first speaker at the microphone, would you identify yourself please?

DONOVAN

My name is John Donovan, MIT Sloan School. Dr. Codd, we have met through CMS, I believe. Let me say that I am enthusiastic about what I have heard from the panel, but I am equally concerned about some of the research directions and applications emphasis that I have heard from the panel. Let me give you specific examples of what I am concerned about.

But before I do, let me say that I think that the relational approach is perhaps one of the most important technologies, if used correctly, to come through in the past three to four years for addressing today's problems. Today's problems, as Jay Forrester pointed out, are that we are running out of energy and running out of food. We need tools to manage those scarce resources. Now, my observations stem from our experience in using the relational implementation SEQUEL and some of your work

directly to build a relational data base system for an information system for New England energy policy decisions. This is not a toy; this is an operational system that State Energy Offices and the New England Regional Commission are using in New England. It is a large data base; all the real problems are there.

First, in his talk Mr. Date used examples taken from inventory control and parts lists. I am concerned about these types of examples, because they tend to throw the profession off the big payoff areas in relational data base systems. I don't see where we have any comparative advantage building inventory control systems. Where we have comparative advantages are in two types of areas: the first is systems that are well structured, but in which the problem keeps changing, as in energy management. We built a crisis management system in 1974. If we had been committed to an IMS system, we would have been in trouble now because the problem changed several times. We are no longer in a shortage crisis management situation. We are in an economic energy management situation now. The second area is in the formulation of an information system in corporations, where the corporate or operational officers don't know what they want, and their apparent needs are changing. In systems of these kinds, I believe we can develop relational data base systems (with appropriate interfaces) for applications in which other systems cannot compete at all. Systems like IMS were not flexible enough for use in our energy management system.

With the possible importance of the relational approach in mind, there are five extensions that I would like to see in relational technology. The data base system is a foundation; obviously we have got to have some floors. Firstly, we have to see some analytical capabilities built on top of relational data base systems: some modeling systems, some statistical packages, some econometrical packages. They can retrieve the data out of relational data base systems, do the modeling, and put new data back in.

Another extension is a multi-user capability. These modeling systems should all interface with the same data base systems.

In your book, Date (which I think is an excellent book, by the way), you advocate the relational approach, but then you have a chapter on IMS. And you say it has one advantage; it works. That is an advantage! Accordingly, I'd like to see some efforts put into making relational data bases operational. Now I mean operational, not in an experimental sense, but in the sense of a real environment, where integrity is important, where protection is important, and where validation is important.

With less priority I would like to see efforts in performance. Performance is important, but I rate it after reduction of fixed costs in building information systems.

Finally, we need interfaces through a relational system to permit the integration of existing data base systems with existing modeling systems.

CODD

John, I think that your notion of the stability of commercial enterprises is a little optimistic. The stability of them is no greater than the stability of our energy resources and may very well be less.

TSICHRITZIS

It is very, very hard in universities to develop a large scale system. We just don't have the resources. So I think that we are in the business of exporting ideas and hope that some of the people here who have the resources are going to do what you ask.

TODD

Our system certainly doesn't do everything you ask for, but it really is operational and it really does get answers. It is being used by Greater London Council Planners. Of course, it answers questions that couldn't be answered by any other methods.

STONEBRAKER

Our system works, at least according to your definition. In addition, we have a pre-compiler running for language C, which allows one to write with a minimal effort all the modeling and statistical packages that are desired. We found that we obtained about a factor of 50 to 100 code reduction in writing such routines in a relational language suitably enhanced with a host language, rather than writing in straight host language code. I agree with your statement that relational systems are most advantageous when data changes and when flexibility of interactions is desired. Most people seem to agree that that represents a very large class of problems.

CODD

Next is Alan Reiter of the Technion in Haifa, Israel.

PEITER

Like Prof. Donovan, I am very excited about the potential of this new technology. So far, the advantages I have seen -- and the advantages claimed by the panel -- seem to be more from the point of view of the data management system designer. It is a nice uniform way of talking about data, allowing one to compare systems, compare algorithms, and so on. It also makes use of some rather elegant mathematical theory. The question I have is addressed from the user's point of view. All the panel members have claimed that the greater simplicity of the relational approach supposedly gives the user great advantages in describing his application. I would like to know if any members of the panel or anyone else in the audience have actually analyzed thoroughly an entire application and have tried comparing approaches -- say, one implementation in relational data base as compared with DBTG or something similar.

GOLDMAN

This panel discussion was not supposed to deal with the comparison issue. There is, in fact, a later session in this conference which does compare two relational sublanguages, SQUARE and SEQUEL [NCC 75 Proceedings & Cassette NCC-72].

CODD

Jay, I thought you were going to talk about actual experiences and applications.

GOLDMAN

We have, as I said earlier, a large number of applications which are used daily by secretarial people and which have been changed over the past two years to incorporate new ideas that the various managers and non-technical people have needed. And I think that the relational model, since it is so simple, is very easily understood by people. In particular, they can understand how the data can be retrieved and how new data can be added to the system.

TSICHRITZIS

Perhaps we have given Prof. Reiter the wrong impression: that relational systems are good for the designers and of dubious value for the user. The reason we concentrated on the implementation questions is that we get lots of criticism in exactly the opposite way: namely, that relational systems must be very bad for the designer because we don't have them working as regular products yet, but they are very good for the users. There are many examples of comparative studies. Many of us

assign students to write applications -- say small insurance systems and personnel systems and so on -- in different approaches, for example DBTG systems and relational systems, and we try to compare them. And these studies are available.

TODD

We haven't actually run comparative studies of applications on different systems, because when we have done paper studies, we have found that the cost of implementing any non-relational systems is beyond our resource level. So we just implement the applications in relations.

CODD

I can only take two more questions. Would you identify yourself please.

MONAKY

Tom Monaky of Riverside County Data Processing, California. Two very quick questions: One, what about the suitability of an APL type interpreter for relational data base systems? Two, given the electronic revolution that's coming through, one can imagine a content-addressable mass storage device, such as a charged couple device, shift register type of thing, in which your pointers and links would all disappear into the hardware. Any comments on these two?

CODD

On the latter question, I refer you to the session on Data Base Machines [NCC 75 Proceedings & Cassette NCC-44]. Who would like to handle the first question?

STONEBRAKER

I think almost all of the current implementations are at least partially interpreters and do work in a similar fashion to APL systems. So, many people are doing what you suggest.

CODD

I would like to thank Chris Date, the panel members, and the audience for their lively participation.

END

UNDERSTANDING RELATIONS

(Installment #7)

E. F. Codd

Q1 Of all the operators in the algebra of n-ary relations, division seems to be the most difficult to understand. Is it analogous in any way to arithmetic division?

A1 There is a useful analogy between certain relational operations and certain arithmetic operations on integers. This analogy may be helpful in understanding relational division and why its name is appropriate.

In ordinary integer arithmetic, dividing 29 by 8 yields 3 as the quotient and 5 as the remainder. The quotient 3 is the largest integer whose product with the divisor 8 is less than the dividend 29. The remainder is obtained by subtracting the product of the divisor and quotient from the dividend.

If we replace arithmetic addition by union of relations, arithmetic multiplication by cartesian product of relations, and arithmetic less-than by inclusion of relations, we can observe a similar pattern in relational division. As an example, consider a binary relation C defined on supplier numbers S# and part numbers P#, where (s,p) belongs to C if supplier s is capable of supplying part p. Suppose we wish to find out which suppliers are capable of supplying both part #2 and part #3. We would set up a unary relation P on P# with the tabulation:

<u>P (P#)</u>
2
3

and divide C by P. Suppose C happens to have the tabulation:

<u>C (S# P#)</u>
u 1
u 2
u 3
v 2
w 2
w 3

The quotient Q is a unary relation defined on S#. It is the largest such relation whose cartesian product with the divisor P is included in the dividend C. Thus, Q has the tabulation:

Q (S#)

u
w

To compute the remainder R, form the cartesian product Q×P and subtract it (in the set-theoretic sense) from C. The following tabulations are obtained:

<u>Q x P (S# P#)</u>	<u>R (S# P#)</u>
u 2	u 1
u 3	v 2
w 2	
w 3	

Note that the union of Q x P with R yields the dividend C.

- Q2 Little attention has been given to the treatment of null values in relations when retrieval operations are being executed. Is it not necessary to extend both the relational algebra and the relational calculus to accommodate null values?
- A2 Present data base products provide little or no support for null values, even though it is a rare data base that does not have numerous instances of missing values (i.e., values that are simply not known at that particular moment). In using these products, the burden of selecting and manipulating suitable representations for these values falls upon the application programmers.

When one adopts a very high level data sublanguage for retrieval and update, the need for explicit system support for null values becomes more pronounced. For example, one would like to have the equi-join operator treat null values in a uniform way that is independent of the particular attributes or underlying domains involved. The scheme discussed below is applicable in a consistent manner to the algebraic, calculus, and mapping oriented types of relational data sublanguages.

We shall concern ourselves with only one type of null value, that which can be interpreted as "value at present unknown". This type of null appears to be the one most needed in data base use, and we shall denote it by "@". Barring any explicit or implicit integrity constraints to the contrary, any occurrence of the value-unknown type of null can be replaced in an updating operation by a non-null value, and vice versa.

The first question which arises is: what is the truth value of x=y if x or y or both are null? An appropriate result in each of these cases is the unknown truth value, rather than true or false. Accordingly, we adopt a three-valued

logic for use in extracting data from data bases that may contain null values. We use the same symbol "@" to denote the unknown truth value, because truth values can be stored in data bases and we want the treatment of all unknown or null values to be uniform. The three-valued logic is based upon the following truth tables:

AND	F	@	T
F	F	F	F
@	F	@	@
T	F	@	T

OR	F	@	T
F	F	@	T
@	@	@	T
T	T	T	T

NOT(F) = T; NOT(@) = @; NOT(T) = F

The existential and universal quantifiers behave like iterated OR and AND respectively.

With regard to set membership \in and set inclusion \subseteq , we assign the truth value @ to the expressions: $a \in S$ and $\{a\} \subseteq S$, whenever S is a non-empty unary relation (even if S does contain a null value). This may seem a bit counter-intuitive at first, but one way to make it seem more acceptable is to think of each occurrence of @ as a placeholder for a possibly distinct value. To be more precise, a truth-valued expression has the value @ if and only if (after replacing any non-individual variables by their defining expressions in terms of individual variables) both of the following conditions hold:

1. Each occurrence of @ in the expression can be replaced by a non-null value (possibly a distinct one for every occurrence) so as to yield the value T for the expression;
2. Each occurrence of @ in the expression can be replaced by a non-null value (possibly a distinct one for every occurrence) so as to yield the value F for the expression.

We shall call this the null substitution principle. The three-valued logic described above is consistent with this principle. The following examples illustrate the application of this principle to set membership and set inclusion. Let \emptyset denote the empty set and R, S, T, U, V denote the following relations:

<u>R</u>	<u>S</u>	<u>T</u>	<u>U</u>	<u>V</u>
@	@	@ 1	x @	x @
1	1	y @	@ 3	y 3
	2			z 1

The following expressions have the value F: $a \in \emptyset$, $S \subseteq R$,

$T \subseteq S, V \subseteq U, U \subseteq R$. The following expressions, on the other hand, have the truth-value \emptyset : $R \subseteq S, T \subseteq U, U \subseteq T, T \subseteq V, U \subseteq V$.

By applying the null substitution principle to inequality testing, we can avoid the arbitrary step of giving \emptyset any place in a numerical or lexicographic ordering. In accordance with this principle, we assign the truth value \emptyset to the expressions: $x \theta y$, where θ is any one of $<, \leq, >, \geq$, whenever x or y is null.

For every positive integer n , the n -tuple consisting of n null values is a legal tuple, but an n -ary relation may contain at most one such tuple. As usual, no relation may contain duplicate tuples (note that some relational systems do not adhere strictly to this rule, and for these systems the scheme described here would need modification). In applying this non-duplication rule, a null value in one tuple is regarded as the same as a null value in another. This identification of one null value with another may appear to be in contradiction with our assignment of truth value to the test $\emptyset = \emptyset$. However, tuple identification for duplicate removal should not be confused with equality testing in the evaluation of retrieval conditions. The consequences for union, intersection, and difference are illustrated below.

R	S	R U S	R ∩ S	R - S
\emptyset \emptyset	\emptyset \emptyset	\emptyset \emptyset	\emptyset \emptyset	
u \emptyset	u \emptyset	u \emptyset	u \emptyset	
u 1	u 1	u 1	u 1	
\emptyset 1		\emptyset 1		\emptyset 1

Now, let us look at the effect of this type of null upon the remaining operators of the relational algebra. Cartesian product remains unaffected. Projection behaves as expected, providing one remembers how the non-duplication rule is applied to tuples with null-valued components. The following examples illustrate projection.

A	R	C	R[B,C]	R[C]
	B		B	C
u	\emptyset	\emptyset	\emptyset \emptyset	\emptyset
v	1	\emptyset	1 \emptyset	
w	\emptyset	1	\emptyset 1	1
x	1	\emptyset		
y	\emptyset	1		

The θ -join operator entails concatenation of pairs of tuples subject to some specified condition θ holding between certain components of these tuples. The evaluation of the condition for any candidate pair of tuples yields the truth value F or \emptyset or T. We retain the join operator

that concatenates only those pairs of tuples for which the condition evaluates to T, and call it a true join. In addition, we introduce a maybe join that concatenates only those pairs of tuples for which the specified condition evaluates to @. The following examples illustrate the true and maybe equi-joins and the true and maybe less-than joins.

<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th colspan="2" style="text-align: center;">R</th></tr> <tr><th style="text-align: center;">A</th><th style="text-align: center;">B</th></tr> <tr><td style="text-align: center;">u</td><td style="text-align: center;">@</td></tr> <tr><td style="text-align: center;">@</td><td style="text-align: center;">2</td></tr> <tr><td style="text-align: center;">w</td><td style="text-align: center;">1</td></tr> </table>	R		A	B	u	@	@	2	w	1	<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th style="text-align: center;">S</th></tr> <tr><th style="text-align: center;">C</th></tr> <tr><td style="text-align: center;">@</td></tr> <tr><td style="text-align: center;">2</td></tr> </table>	S	C	@	2	<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th colspan="3" style="text-align: center;">R[B=C]S</th></tr> <tr><th style="text-align: center;">A</th><th style="text-align: center;">B</th><th style="text-align: center;">C</th></tr> <tr><td style="text-align: center;">@</td><td style="text-align: center;">2</td><td style="text-align: center;">2</td></tr> </table>	R[B=C]S			A	B	C	@	2	2	<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th colspan="3" style="text-align: center;">R[B=@C]S</th></tr> <tr><th style="text-align: center;">A</th><th style="text-align: center;">B</th><th style="text-align: center;">C</th></tr> <tr><td style="text-align: center;">u</td><td style="text-align: center;">@</td><td style="text-align: center;">@</td></tr> <tr><td style="text-align: center;">u</td><td style="text-align: center;">@</td><td style="text-align: center;">2</td></tr> <tr><td style="text-align: center;">@</td><td style="text-align: center;">2</td><td style="text-align: center;">@</td></tr> <tr><td style="text-align: center;">w</td><td style="text-align: center;">1</td><td style="text-align: center;">@</td></tr> </table>	R[B=@C]S			A	B	C	u	@	@	u	@	2	@	2	@	w	1	@
R																																												
A	B																																											
u	@																																											
@	2																																											
w	1																																											
S																																												
C																																												
@																																												
2																																												
R[B=C]S																																												
A	B	C																																										
@	2	2																																										
R[B=@C]S																																												
A	B	C																																										
u	@	@																																										
u	@	2																																										
@	2	@																																										
w	1	@																																										

<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th colspan="3" style="text-align: center;">R[B<C]S</th></tr> <tr><th style="text-align: center;">A</th><th style="text-align: center;">B</th><th style="text-align: center;">C</th></tr> <tr><td style="text-align: center;">w</td><td style="text-align: center;">1</td><td style="text-align: center;">2</td></tr> </table>	R[B<C]S			A	B	C	w	1	2	<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th colspan="3" style="text-align: center;">R[B<@C]S</th></tr> <tr><th style="text-align: center;">A</th><th style="text-align: center;">B</th><th style="text-align: center;">C</th></tr> <tr><td colspan="3" style="text-align: center;">same as</td></tr> <tr><td colspan="3" style="text-align: center;">R[B=@C]S</td></tr> </table>	R[B<@C]S			A	B	C	same as			R[B=@C]S		
R[B<C]S																						
A	B	C																				
w	1	2																				
R[B<@C]S																						
A	B	C																				
same as																						
R[B=@C]S																						

If we wish to select only those rows of R that have @ as their B-component, we may form the maybe equi-join of R with a relation T whose only element is a single non-null value (any such value will do, providing it is drawn from the same underlying domain that attribute B is defined on), and then project the result on A,B. In the case above, the reader can verify that the final result is a relation whose only element is the pair (u,@). Treatment of null values by the selection or restriction operators follows the same pattern as the join operators.

Division is treated in a similar manner. The original operator based upon true inclusion (inclusion testing that yields T) is retained and called true division. A new division operator $\div @$ is introduced which entails only maybe inclusion (inclusion testing that yields @), and this is called maybe division. The following examples illustrate the two kinds of division.

<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th colspan="2" style="text-align: center;">R</th></tr> <tr><th style="text-align: center;">A</th><th style="text-align: center;">B</th></tr> <tr><td style="text-align: center;">u</td><td style="text-align: center;">1</td></tr> <tr><td style="text-align: center;">u</td><td style="text-align: center;">2</td></tr> <tr><td style="text-align: center;">u</td><td style="text-align: center;">3</td></tr> <tr><td style="text-align: center;">w</td><td style="text-align: center;">2</td></tr> <tr><td style="text-align: center;">w</td><td style="text-align: center;">@</td></tr> <tr><td style="text-align: center;">x</td><td style="text-align: center;">3</td></tr> </table>	R		A	B	u	1	u	2	u	3	w	2	w	@	x	3	<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th style="text-align: center;">S</th></tr> <tr><th style="text-align: center;">C</th></tr> <tr><td style="text-align: center;">2</td></tr> <tr><td style="text-align: center;">3</td></tr> </table>	S	C	2	3	<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th style="text-align: center;">T</th></tr> <tr><th style="text-align: center;">C</th></tr> <tr><td style="text-align: center;">2</td></tr> <tr><td style="text-align: center;">@</td></tr> </table>	T	C	2	@	<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th colspan="2" style="text-align: center;">R[B÷C]S</th></tr> <tr><th style="text-align: center;">A</th><th style="text-align: center;">C</th></tr> <tr><td style="text-align: center;">u</td><td style="text-align: center;">2</td></tr> </table>	R[B÷C]S		A	C	u	2	<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th colspan="2" style="text-align: center;">R[B÷@C]T</th></tr> <tr><th style="text-align: center;">A</th><th style="text-align: center;">C</th></tr> <tr><td style="text-align: center;">empty</td><td style="text-align: center;">2</td></tr> </table>	R[B÷@C]T		A	C	empty	2
R																																								
A	B																																							
u	1																																							
u	2																																							
u	3																																							
w	2																																							
w	@																																							
x	3																																							
S																																								
C																																								
2																																								
3																																								
T																																								
C																																								
2																																								
@																																								
R[B÷C]S																																								
A	C																																							
u	2																																							
R[B÷@C]T																																								
A	C																																							
empty	2																																							

<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th colspan="2" style="text-align: center;">R[B÷@C]S</th></tr> <tr><th style="text-align: center;">A</th><th style="text-align: center;">C</th></tr> <tr><td style="text-align: center;">w</td><td style="text-align: center;">2</td></tr> </table>	R[B÷@C]S		A	C	w	2	<table style="border-collapse: collapse; margin: 0 auto;"> <tr><th colspan="2" style="text-align: center;">R[B÷@C]T</th></tr> <tr><th style="text-align: center;">A</th><th style="text-align: center;">C</th></tr> <tr><td style="text-align: center;">u</td><td style="text-align: center;">2</td></tr> <tr><td style="text-align: center;">w</td><td style="text-align: center;">2</td></tr> </table>	R[B÷@C]T		A	C	u	2	w	2
R[B÷@C]S															
A	C														
w	2														
R[B÷@C]T															
A	C														
u	2														
w	2														

Let us turn our attention now to calculus-based data sublanguages and, to be specific, consider ALPHA. A very simple extension suffices for handling null values. Two important parts of the GET and DELETE statements are the

target list and qualification expression. The target list specifies the kinds of tuples to be operated upon (retrieved or deleted) and the qualification expression specifies a condition that each such tuple must satisfy. When these parts are separated by a colon only, the operation is performed on precisely those tuples for which the qualification expression evaluates to T (i.e., the usual interpretation applies). When, however, the colon is immediately followed by the symbol "@", the operation is performed on precisely those tuples for which the qualification expression evaluates to @. Note that the MAYBE TOO modifier cited in section 2.3 of the ALPHA report [SIGFIDET Workshop 1971] is exactly equivalent to retrieving or deleting the union of the true set and the maybe set of tuples.

Although language extensions for null value support are conceptually simple, these extensions are not easy to implement efficiently on present hardware. This may explain why so few existing systems provide such support.

Q3 How do arithmetic operations and library functions treat this type of null value?

A3 The arithmetic operations of addition, subtraction, multiplication, and division yield null if any operand is null-valued. Many library functions would similarly yield null if any argument were null-valued. However, for certain functions -- particularly those that map a set of elements into a single number (as many statistical functions do) -- users need the option of having null values ignored. Of course, a count, sum, or average of the non-null elements in a set can always be carried out in two steps: first, retrieve only the non-null elements, and then perform a regular count, sum, or average upon the retrieved set (taking care to preserve associated key values if duplicate non-key values are to participate in the evaluation). These functions tend to be used sufficiently frequently to justify providing the option of collapsing the two steps into one.

In regard to A2 and A3 the author wishes to express his appreciation to J-M. Cadiou, D. D. Chamberlin, and K. P. Eswaran of this laboratory for helpful discussions.

Correspondence should be addressed to:

Dr. E. F. Codd
IBM Research Laboratory K51-282
Monterey & Cottle Roads
San Jose, California 95193

All rights reserved

focuses mainly on definition and enforcement of access constraints. There is also a brief discussion of "scrambling" as a defense against unauthorized access. Chapter 20 discusses "integrity constraints", which Date defines to include edit checks, procedures for control over multiple updates, and backup-and-recovery procedures.

Date's book is a readable introduction to a wide variety of topics concerning database management. However, a reader may have some reservations concerning the book. It has an acknowledged, but nonetheless strong, bias toward the relational approach to database management. Its style is often choppy, due to a plethora of qualifying footnotes on matters such as terminology, software details, etc. It does not deal with certain issues of interest to persons using or preparing to implement database systems; e.g., data dictionary/directory systems, the role and function of the database administrator, and the existence of other DBMS packages featuring different approaches to database management

I have used the book as a text in a course on file and database management at the Graduate School of Business Administration at New York University and plan to do so again.

Jay-Louise Weldon
Assistant Professor of Computer Applications and
Information Systems
New York University School of Business

* * * * *

A feature of Interactive Retrieval Systems.

Thomas H. Martin.
Stanford Univ., Calif. Inst. for Communication Research.
Sep 74; 104p SU-COMM-ICR-74-1
PB-235 952/9WC PC\$5.25/MF\$2.25



The command language features of eleven different on-line information retrieval systems are presented in terms of the functional needs of a searcher sitting at a terminal. Functional areas considered are: becoming familiar with the system, receiving help when in trouble, regulating usage, selecting a data base, formulating simple queries, expressing single concepts, interconnecting concepts, displaying results simply, and controlling the display. Features felt most essential to on-line searching are live help, users' guides, boolean operators, search field control, suffix removal, relational operators, dictionary access, request sets, search review, predefined formats, on-line formatting, and off-line printing.

Compiled by
Claude L. Overstreet
Computer Science Department
Bowling Green State University
Bowling Green, Ohio 43403

Note: Papers with an NTIS report number are available from National Technical Information Service Springfield, Virginia 22151, at the prepaid prices indicated for paper copy (PC) and microfiche (MF).

Stored Data Definition and Translation Approach to the Data Portability Problem.

James P. Fry.
Michigan Univ Ann Arbor Dept of Industrial and Operations Engineering Feb 74, 25p
AD/A-003 736/6WC PCS3.25/MF\$2.25

The data portability problem is the inability to move data from one system, which has inherently different characteristics in either architecture or software, to another. This problem is costing industry and government millions of dollars each year. One approach to this problem is termed data translation: the process whereby data on one computer is changed into a form that is processable on the same computer with a different software system, or possibly a different computing system. The technical approach to data translation embodies the development of two declarative languages which describe stored-data and the transformations among stored-data and the implementation of a data translator which processes the stored-data. A Stored-Data Definition Language (SDDL) is being developed to declare the logical data structure and the physical encoding of its data instances for both the source and target files. A Translation Definition Language (TDL) is used to define how the target data instances are to be derived from the source data instances.

A Model for a Generalized Data Access Method.

Randall L. Frank, and Koichi Yamaguchi.
Michigan Univ Ann Arbor Dept of Industrial and Operations Engineering Apr 74, 10p AFOSR-TR-74-1852 Availability: Pub. in Proceedings of the 1974 National Computer Conference and Exposition, 6-10 May 74, Chicago, Ill. p45-52 1974.
AD/A-003 972/7WC PCS3.25/MF\$2.25

The proliferation of the methods used in modern operating systems to access data is apparent. In the operating system OS/360 alone there exist multiple ways of accessing sequential data, indexed sequential data and directly accessible data. If one adds to this the variations of the above used by various systems that run under the control of OS/360, such as IBM's data base management system IMS/360, there exists almost a countless number of ways to access and store data within a computer system. This proliferation of data access methods has left persons involved in the design, implementation or evaluation of new or existing access methods with an almost hopeless task. In order to better understand the nature of data access methods, a model has been developed in whose terms existing and proposed data access methods can be stated. This paper will discuss the components of such a generalized data access method and give examples of its use in modeling existing data access methods. Parts of the model to be presented are at a further state of development than others, and therefore, at times a formal discussion of parts of the model may be replaced by a more informal functional description. (Author)

Experiences in Designing a Data Retrieval Capability for a Large Data Base.

L. H. Heiser.
Rand Corp Santa Monica Calif Apr 74, 14p P-5113
AD-786 712/0WC PCS3.25/MF\$2.25

DATAR is a package of subroutines that provide data retrieval capabilities for a large weather data base. The design of this package was a valuable learning experience for the author. Questions of portability had to be resolved since the package must run on several different computers. Questions of interface arose since DATAR would be imbedded in other programs. Also, it was desirable to have changes to the subroutines be transparent to the user. And the list goes on. It is the author's intent to discuss the design of DATAR and to share the learning experiences. (Author)

A Data Description Language Approach to File Translation.

Alan G. Merten, and James P. Fry.
Michigan Univ Ann Arbor Dept of Industrial and Operations Engineering Mar 74, 19p ISDOS Data trans-working paper-304, ISDOS-Working paper-93, AFOSR-TR-75-0038
AD/A-003 715/0WC PCS3.25/MF\$2.25

Data Translation is the process whereby information (data) stored by one computer on files in some particular structure may be transformed so that they could be read by another computer (possibly manufactured by another supplier, and hence, normally incompatible) according to some other file structure. The overall research goal is to develop a generalized methodology for data translation.

On the Implementation of a Physical Data Model for Translation.

James P. Fry.
Michigan Univ Ann Arbor Dept of Industrial and Operations Engineering May 74, 25p Data trans-working paper-6.05, AFOSR-TR-75-0036
AD/A-003 737/4WC PCS3.25/MF\$2.25

The central thesis of this paper is the development of a physical model for stored-data. To this end, the paper reports on the implementation of a physical data model for the University of Michigan Data Translator. First, the evolution of data models for translation is traced and the relevant literature is reviewed. In Section 3 the model is described using a top-down methodology. The basic design considerations are reviewed and the implementation details presented. This section also describes the utilization of the model in the translation process. Section 4 concludes the paper by presenting some observations on the implementation of physical data models for data translation.

Understanding Data Structures.

Rob Gerritsen.
Carnegie-Mellon Univ Pittsburgh Pa Dept of Computer Science Feb 75, 230p AFOSR-TR-75-0545
AD-A008 937/5WC PCS7.50/MF\$2.25

Data management programmers are finding their jobs are getting tougher because of the gradual replacement of sequential data bases by network data bases. In addition, there is a new job called 'Data Administrator' for handling the data structure problems associated with network data bases. The goal of this thesis is reduction of these data management tasks by developing and applying a practical theory of data structure. To insure the practical flavor of this research, the Data Base Task Group (DBTG) report has been selected as the specification of the data management system in which the applications function.

A Low Cost Minicomputer Data Base.

Frederick A. Skove.

Naval Academy Annapolis Md Environmental Protection
Research and Development Team 28 May 74. 30p USNA-
EPRD-5

AD/A-000 948/0WC PCS3.75/MF\$2.25

The report is written in an attempt to assess the electronic computer capability needed for a Navy-wide Environmental Protection Data Base (NEPDB) which will store data pertaining to ships, aircraft, and shore activities. The Base may vary from 0 to 100% centralization and from 0 to 100% automation of numerical and textual data handling. It is the intention of this report to show that with the use of minicomputers, large data processing equipment may not be necessary. The seemingly obvious concept of one large data bank, with one large computer centrally located may not be the way to maximize output while keeping costs at a minimum.

Optimal Program and Data Locations in Computer Networks.

Howard Lee Morgan, and Katriel Dan Levin.

Wharton School of Finance and Commerce Philadelphia Pa
Dept of Decision Sciences (Management) 1974, 23p 74-10-
01

AD/A-001 008/2WC PCS3.25/MF\$2.25

An optimization procedure for the allocation of program and data files in a computer network is presented. This algorithm takes into account the dependencies between files and programs such as occur in real heterogeneous computer networks. Insights into whether or not to convert programs from one computer to another can also be gained from the model. A search procedure for the file location problem is described, along with an example and a possible application of the model.

Organizing Distributed Data Bases in Computer Networks.

Katriel Dan Levin.

Wharton School of Finance and Commerce Philadelphia Pa
Dept of Decision Sciences (Management) Sep 74, 22p 74-
09-01

AD/A-001 009/0WC PCS7.50/MF\$2.25

This research addressed the file location problem for both program and data sharing. In particular, dependencies between programs and data files have been considered, as well as their impact on the optimal distribution of files in the network. Having reviewed existing file location models, a distinction between data sharing and program and data sharing was established. Subsequently, the problems of creating and operating distributed data bases were considered with brief exemplary solutions for these problems being offered. A three dimensional partitioning of the file location problem was employed as the framework for the main body of this research.

Dynamic File Assignment in a Computer Network Part II:**Random Rates of Demand.**

Adrian Segall.

Massachusetts Inst of Tech Cambridge Electronic Systems
Lab Feb 75, 30p ESL-P-590, AFOSR-TR-75-0559

AD-A008 645/4WC PCS3.75/MF\$2.25

The problem of dynamic file allocation in a computer network is treated in the case when the time-pattern of the file demand rates is not perfectly known in advance to the designer. Instead, only a prior distribution and a statistical behavior are assumed, and the rates have to be estimated on-line from the incoming requests. It is shown that these estimates are sufficient statistics for the optimal control. The equations giving the control laws as a function of the current estimates and the present location of the file are obtained.

DMS Evaluation Methodology.

George S. Pan, and Wayne L. Schoeffel.

Systems Architects Inc Randolph Mass Aug 74. 208p
RADC-TR-74-198

AD-787 858/0WC PCS7.25/MF\$2.25

The report was initiated to investigate the feasibility of automating the data management system (DMS) selection process. The selection of a DMS for a particular set of user requirements is often hindered by the lack of a systematic approach to the specification of design constraints. This report details an approach to gathering user requirement in a coherent fashion through the completion of prearranged forms by the user or analyst. Concurrently, the characterization of a set of candidate DMS is made by indicating on a DMS attribute tree whether particular features exist. Finally, the selection of a DMS is made through the use of mapping tables which correlate particular user requirements as indicated on requirements forms to particular DMS attributes as indicated on the attribute tree. If no match is found, a group of usage equations may be invoked to provide a mathematical estimation of the relative need and importance of the particular quantity. (Author)

A Transposition Algorithm for Digital Data Compression Keys.

Fred N. Berra.

Air Force Flight Test Center Edwards AFB Calif Sep 74,
21p AFFTC-TD-74-2

AD/A-006 798/3WC PCS3.25/MF\$2.25

A key transposition algorithm, a procedure by which computer words are transformed into entities that are used to store and retrieve table information with great efficiency, is useful in many areas of computer information retrieval. A specific key transposition algorithm is presented which applies to a set of digital data compression key integers over the range of $1 < \text{or} = k < \text{or} = n$. This set is non-continuous and non-uniform, but has definable subsets (ranging over $k \text{ sub } i < \text{or} = k < \text{or} = k \text{ sub } j$) which are sequentially uniform. The algorithm operates from densely stored tables and performs most entries to obtain table information with a divide and add operation. A minimum controlled scan is used to retrieve the information only when a transition occurs between one subrange and another. The number of scanned key integers is usually very small. Some timing comparisons with a logarithmic search are presented showing from 30 to 40 percent improvements depending on the digital data compression key structure defined.

A Model for Data Secure Systems. Part II.

D. K. Hsiao, and E. J. McCauley, III.

Ohio State Univ Columbus Computer and Information
Science Research Center Oct 74, 46p OSU-CISRC-TR-74-7

AD/A-006 276/0WC PCS3.75/MF\$2.25

The major portion of this report is devoted to the development and discussion of a Structural Model. The authors expand upon the conceptual model introduced in Part I to include a discussion of the overall data base structure and algorithms for its maintenance and use. The atom concept is improved to reflect more accurately the nature of security specifications. The concept is unified with the concept of storage cell in the model for the study of protection completeness and access and storage efficiency. The model shows that proper utilization of the atom and cell concepts can result in a data secure system with effective protection and efficient access. Furthermore, these concepts can lend themselves toward storage compart-

On-Line Management Information System: Feasibility in an R and D Environment.

Harold F. O'Neil, Jr., Mary E. Walker, George H. Walther, and Wilson A. Judd.

Texas Univ At Austin Computer-Assisted Instruction Lab
Dec 74, 83p AFHRL-TR-74-98

AD-A007 723/0WC PC\$4.75/MF\$2.25

The objectives of this study consisted of three stipulated tasks. These tasks were: (1) to conduct and document a thorough, comprehensive review of existing literature which addresses itself to implementation and evaluation of on-line data management systems; (2a) to analyze methods currently in existence within the Air Force Human Resources Laboratory (AFHRL) for processing management and planning information; and (2b) to analyze the information needs of a designated subset within the AFHRL; (3a) to develop implementation and evaluation strategies; (3b) to demonstrate and evaluate the feasibility of the strategies and techniques developed. The interim report discussed tasks (1) and (2) and provided baseline data for task (3). This final report will address itself to objective (3).

Set Processing in a Network Environment.

W. T. Hardgrave.

Universities Space Research Association, Charlottesville, Va.

Inst. for Computer Applications in Science and Engineering.

31 Mar 75, 54p NASA-CR-142597, ICASE-75-7

N75-21035/1WC PCS4.25/MF\$2.25

A combination of a local network, a mass storage system, and an autonomous set processor serving as a data/storage management machine is described. Its characteristics include: content-accessible data bases usable from all connected devices; efficient storage/access of large data bases; simple and direct programming with data manipulation and storage management handled by the set processor; simple data base design and entry from source representation to set processor representation with no predefinition necessary; capability available for user sort/order specification; significant reduction in tape/disk pack storage and mounts; flexible environment that allows upgrading hardware/software configuration without causing major interruptions in service; minimal traffic on data communications network; and improved central memory usage on large processors. (Author)

Languages for Specifying Protection Requirements in Data Base Systems. Part I.

H. R. Hartson, and D. K. Hsiao.

Ohio State Univ Columbus Computer and Information

Science Research Center Jan 75, 68p OSU-CISRC-TR-74-

10

AD/A-006 280/2WC PC\$4.25/MF\$2.25

This report develops a macro-oriented model of access control to accommodate constructs of protection languages at many levels of sophistication. The concept of ownership is explicitly represented in the model and is expanded to include sub-ownership and conditional ownership. The basic sets of the set-theoretic model are presented and the set of system states is derived from the set of all values of the resources. Restrictions on resource values define subsets of states, which are described by expressions called conditions. The concept of a five-dimensional security space is used to visualize how authorization specifications and access requests are manipulated by the authorization and enforcement processes. Several examples are presented to illustrate the relationships among the various parts of the security space.

Evaluation and Optimization of File Organizations through Analytic Modeling.

Shi Bing Yao.

Michigan Univ Ann Arbor Dept of Industrial and Operations Engineering 1974, 220p AFOSR-TR-75-0275 0

AD/A-005 721/6WC PC\$7.25/MF\$2.25

The research concerns the selection of appropriate file structures and access methods for the construction of large data bases within information systems. A file design system is developed here that can be used by a file designer to select good file organizations from a large number of alternatives. This design system takes its design requirement parameters from the data collection, user activities, and machine characteristics. The output of the design system is a suggested file structure, and the details of the actual file structure can be determined by simulation or other techniques.

Natural Language Based Information Retrieval.

R. P. Gabriel, and D. L. Waltz.

Illinois Univ Urbana Coordinated Science Lab 1974, 11p

AD/A-001 131/2WC PC\$3.25/MF\$2.25

A program which accepts natural language queries can allow a nontechnical user to easily obtain information from a large non-uniform data base. To date most natural language systems have used parsers whose operation bears dubious resemblance to human language understanding and furthermore, these systems have proved to be difficult to extend to new data bases. This paper discusses the design of a new natural language data base interrogation system which should overcome these difficulties.

Accessing Technical Data Bases Using STDS: A Collection of Scenarios.

W. T. Hardgrave.

Universities Space Research Association, Charlottesville, Va.

Inst. for Computer Applications in Science and Engineering.

16 Apr 75, 85p NASA-CR-142599, ICASE-75-8

N75-21042/7WC PCS4.75/MF\$2.25

A line by line description is given of sessions using the set-theoretic data system (STDS) to interact with technical data bases. The data bases contain data from actual applications at NASA Langley Research Center. The report is meant to be a tutorial document that accompanies set processing in a network environment. (Author)

A FORTRAN System for Flexible Creation and Accessing of Data Bases.

Leon Osterweil, Lori Clarke, and David W. Smith.

Colorado Univ., Boulder. Dept. of Computer Science. Aug 74, 110p CU-CS-052-74

PB-236 341/4WC PC\$5.25/MF\$2.25

This paper discusses the implementation of a data base management system designed to enable easy, flexible, data access from FORTRAN programs. The system was created in response to the need for a library of routines for accessing an archival data base whose structure is only tentative and is expected to change frequently and perhaps radically. A FORTRAN library interface between user programs and data bases is described. A methodology for conveniently restructuring data bases without having to change the FORTRAN programs accessing the data bases is presented. This paper also describes a system for restoring and then accessing a data base which has been saved in an obsolete format without having to alter any of the data accessing programs.

Secure Data Management System Preliminary Mathematical Model.

Marvin Schaefer.
System Development Corp Santa Monica Calif Feb 75, 43p
RADC-TR-74-352
AD-A007 784/2WC PC\$3.75/MF\$2.25

In this report a model of a Data Access Monitor (DAM) is developed and certified for operation in an environment in which there are multiple classification levels on subjects and objects. The model is expressed in the program language PASCAL. The MULTICS hierarchical directory structure is modeled and related to the design fundamentals of a relational data management system.

Natural Language Access to a Large Data Base.

David L. Waltz.
Illinois Univ At Urbana-Champaign Coordinated Science Lab
Apr 75, 34p
AD-A013 578/0WC PC\$3.75/MF\$2.25

The report describes the first year's accomplishments toward a natural language system which answers questions about a data base of naval aircraft maintenance and flight data. The system is designed to: Allow a user to ask questions in natural English; provide answers to questions requiring averaging, statistical analysis, comparison of set of data, and other complex functions as well as answers to simpler questions about specific data base records; provide aid in evaluating and predicting the causes of failures and of the need for excessive amount of maintenance work.

An Annotated Bibliography to Network Data Management and Related Literature.

Peter A. Alsberg, Geneva G. Belford, Deborah S. Brown, Steve R. Bunch, and John D. Day.
Illinois Univ At Urbana-Champaign Center for Advanced Computation 1 Apr 75, 283p UIUC-CAC-DN-75-149
AD-A014 232/3WC PC\$8.75/MF\$2.25

Over 400 documents related to network data management and resource sharing are annotated. The documents cover topics in data management, computer networks, operating system concepts, communications, resource allocation, measurement and analysis, front ends, security and application support.

INGRES - A Relational Data Base System.

G. D. Held, M. R. Stonebraker, and E. Wong.
California Univ Berkeley 1975, 10p ARO-11895.4-EL
Availability: Pub. in AFIPS - Conference Proceedings, v44 p409-416 1975.
AD-A013 151/6WC PC\$3.25/MF\$2.25

INGRES (Interactive Graphics and Retrieval System) is a relational data base and graphics system which is being implemented on a PDP-11/40 based hardware configuration at Berkeley. INGRES runs as a normal user job on top of the UNIX operating system. The only significant modification to UNIX that INGRES requires is a substantial increase in the maximum file size allowed. The implementation of INGRES is primarily programmed in 'C', a high level language in which UNIX itself is written. Parsing is done with the assistance of YACC, a compiler-compiler available on UNIX. This paper describes most of the principal components of INGRES. These include: the query language QUEL, an algorithm for processing interactions based on the principle of 'decomposition,' and an approach to access control, views, and integrity preservation via query modification.

Optimal Data Base Schema Design.

Michael F. Mitoma.
Michigan Univ Ann Arbor Systems Engineering Lab Aug 75,
354p RADC-TR-75-175
AD-A016 431/9WC PC\$10.50/MF\$2.25

This report describes a methodology that will automate and optimize the logical structure of a data base for the application that it supports. The methodology consists of three mathematical models and an automated design procedure. A case study analysis is presented and a number of optimal data bases are generated and discussed.

Data Models for Secondary Storage Representations.

Allen Reiter.
Wisconsin Univ Madison Mathematics Research Center Jul 75, 64p MSR-TSR-1554
AD-A016 347/7WC PC\$4.50/MF\$2.25

Performance modelling of data base management systems, and in particular data base models capable of describing representation features in secondary storage, are discussed. Data structures relevant for secondary storage representations are defined, and a partial characterization of several actual systems is given. The accent is on the use of these structures in the context of a simulation model.

File Organizations with Consecutive Retrieval and Related Properties.

Anthony N. Patrinos, and S. Louis Hakimi.
Northwestern Univ Evanston Ill Dept of Electrical Engineering 1975, 24p AFOSR-TR-75-1422
AD-A015 422/9WC PC\$3.25/MF\$2.25

File organization basically involves the introduction of a structure among records (in a file) to simplify the retrieval of records with common attributes. The authors study the consecutive retrieval property in files and describe an alternative algorithm to determine whether or not a file has consecutive retrieval property. Other file structures with interesting graph theoretic properties which are generalizations of the consecutive retrieval property are introduced. Some related research problems are described.

Evaluation and Selection of File Organizations through Analytic Modeling.

S. B. Yao, and A. G. Merten.
Michigan Univ Ann Arbor Graduate School of Business Administration 1975, 24p AFOSR-TR-75-1294
AD-A015 934/3WC PC\$3.25/MF\$2.25

In this paper, a single model and cost function is developed to characterize most of the file structures for a design problem is automated. A file design system is developed that can be used by a file designer to select good file organizations from a large number of alternatives. This generalized model makes explicit the principles underlying data base construction. The computer program which implements the model uses analytic optimization techniques to select file organizations.

Feature Analysis of CODASYL Data Base Management Systems.

Tom Warren.

Department of Defense Washington D C 25 Jun 75, 114p
AD-A014 972/4WC PC\$5.25/MF\$2.25

The CODASYL specifications, it is generally assumed, represent currently the most viable approach towards developing commonality among data base management systems. Though it is still a long way from becoming a standard, implementations based on the CODASYL specifications may significantly diminish the problems associated with data base sharing and portability of programs between heterogeneous computer systems, problems which are becoming more prevalent with the advent of computer networking. This report will look in detail at four systems of the CODASYL family. Their features will be outlined to allow a judgement on the commonality that exists with the CODASYL approach. The four systems are DBMS-10, available from Digital Equipment Corporation on the PDP-10; IDMS, marketed by the Cullinane Corporation and running primarily on IBM equipment; DMS 1100, a Univac product on the 1100 series; and EDMS by Xerox, implemented on its Sigma and 560 series.

A CODASYL-type DBMS System in SIMULA.

Kalle Makila.

Research Inst. of National Defence, Stockholm (Sweden).
Aug 75, 35p FOA-C-10038-M3(E5)
PB-246 999/7WC PC\$4.00/MF\$2.25

A CODASYL type Data Base Management System (DBMS) has been written entirely in SIMULA. The system is called SIMDBM. SIMDBM is fully usable in itself for data bases of moderate size. It is especially suitable for education about Data Base Management, because of its simplicity and clarity. The system is also an investigation into how an interface can be arranged between the SIMULA programming language and a DBMS system. There is a need both for a general purpose interface, where the SIMULA program without change can handle any data base, and a special purpose interface generated for a specific application, in which data base RECORDS are directly mapped onto corresponding SIMULA CLASS objects, with one CLASS attribute for each FIELD of the data base RECORD.

Concept of a Management Information System for TESPO.

B. G. VanBlaricum, and June G. Brenton.

Dikewood Corp Albuquerque N Mex Oct 75, 45p DC-TR-1242-1, AFSWC-TR-75-5
AD-A016 452/5WC PC\$3.75/MF\$2.25

In a recent survey of available Management Information Systems, three systems that can most nearly fulfill the requirements of the Test and Evaluation System Program Office (TESPO) were selected for detailed investigation. The results of the survey and study are given in this report. The conclusion is drawn that the TESPO should select one of the three for implementation to assist with management of the Near-Term COR development and that the operating MIS be improved as experience is gained through operation and methods of improvement are further explored.

Ideas About Management of Lisp Data Bases.

Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab May 75, 38p AI-M-332
AD-A013 312/4WC PC\$3.75/MF\$2.25

The paper advocates the need for systems which support maintenance of LISP-type data bases, the describes an experimental system of this kind, called DABA. In this system, a description of the data base's structure is kept in the data base itself. A number of utility programs use the description for operations on the data base. The description must minimally include syntactic information reminiscent of data structure declarations in more conventional programming languages, and can be extended by the user. Two reasons for such systems are seen: (1) As A.I. programs develop from toy domains using toy data bases, to more realistic exercises, the management of the knowledge base becomes non-trivial and requires program support. (2) A powerful way to organize LISP programs is to make them data-driven, whereby pieces of program are distributed throughout a data base. A data base management system facilitates the use of this programming style.

Users Guide for Information Retrieval Using APL.

A. Shapiro.

National Aeronautics and Space Administration. Goddard Space Flight Center, Greenbelt, Md. Aug 74, 19p NASA-TM-X-72578, NSSDC-74-11
N75-30934/4WC PC\$3.25/MF\$2.25

A computer program called IRA (information retrieval using APL) is described. This program accesses a data base containing information relating to space science experiments and to the phenomena being measured. IRA is a tool for retrieving information in response to requests made by the scientific community. IRA is executed from a remote terminal (to the IBM 360/95) and results are achieved in seconds. The data base, the general search requirement, and the system's operation characteristics are described. (Author)

Data Security Implications of an Extended Subschema Concept.

Frank A. Manola, and Stanley H. Wilson.

Naval Research Lab Washington D C 15 Jul 75, 19p NRL-7905
AD-A013 248/0WC PC\$3.25/MF\$2.25

Modern data base system architectures, such as those based on the CODASYL Data Base Task Group (DBTG) specifications, stress the idea that users of the system should interact with a logical description of that portion of the data base with which they are concerned, called a subschema, which is derived from a logical description of the data base as a whole, called a schema. One of the benefits of this architecture is its ability to provide enhanced data security, since the mapping from schema to subschema may conceal information from the user. Use of this architecture to enhance data security was studied with respect to schema, subschema, and design and implementation of application programs. Instances of such use include both that in the existing DBTG specifications and some proposed extensions to that architecture.

Proceedings of the Workshop on Data Bases for Interactive Design,
 Waterloo, Canada - Sept 15-16, 1975 (Sponsored by SIGOA, SIGMOD &
 SIGGRAPH.)

Significant Developments in Data Base Management Systems, James P. Fry, Univ. of Michigan, Ann Arbor, Mi.	1
Evolving Concepts of Graphics Data Bases in Design-Aids Systems, C.W. Rosenthal, L. Rosler, Bell Laboratories, Murray Hill, N.J.	13
Enhancements to the DBTG Model for Computer-Aided Ship Design, A.E. Bandurski, D.K. Jefferson, Naval Ship R & D Center, Bethesda, Md.	17
An environment for the Interactive Evaluation of Scientific Data and its Application in Computer Aided Design, M. Bergen, P. Erbe, P. Pistor, U. Schauer, G. Walah, IBM Heidelberg Scientific Center, Germany	26
Personalized Management and Graphical Display of Data: An Extensible System Approach, L. Borman, W.D. Dominick, R. Hay, Jr., P. Kron, B. Mittman, Northwestern Univ., Evanston, Ill,	36
Database Features for a Design Information System, C.M. Eastman, A. Baer, Carnegie- Mellon Univ., Pittsburgh, Penn.	45
Data Structures for Interaction, D.F. Barnard, Edinburgh CAAD Studies, Edinburgh, England	54
An Approach to Implementing a Geo-Data System, A. Go, M. Stonebraker, C. Williams, Univ. of California, Berkeley, Ca.	67
Data Structures in Computer Graphics, R. Williams, G.M. Giddings, W.D. Little, W.G. Moorhead, D.L. Weller, IBM Research Laboratories, San Jose, Ca.	78
Data Base Structures - Present and Future, W.C. McGee, IBM Corporation, Palo Alto, Ca.	83
A Data Base Design for Digital Design Automation, M.M. Matelan, R.J. Smith, II, Lawrence Livermore Laboratory, Livermore, Ca.	85
Integration of Data-Base Management and Project Control for Engineering Design, S.J. Fenves, Carnegie-Mellon Univ., Pittsburgh, Penn.	93
VDAM - A Virtual Data Access Manager for Computer Aided Design, D.R. Warn, General Motors Corporation	104
A Fundamental Approach to Data Base Implementation for Design Automation, G.S. Melanson, S.A. Spurlin, Harris Corporation	112
A Data Base for Editing Printed Circuit Artwork in an Interactive CRT Environment, R.D. Wrigley, Bell Telephone Laboratories, Holmdel, N.J.	120
A Computer System for Numerical Geometry Design, G.J. Silverman, K.L. Johnson, IBM Scientific Center, Los Angeles, Ca.	130
Use of a Generalized Data Base Management System in a Design Automation System, D.P. Yelton, Digital Equipment Corporation	139
Individual Data Bases for Circuit Design, A.J. Korenjak, A.H. Teger, RCA Laboratories, Princeton, N.J.	140
DPLS - Database, Dynamic Program Control & Open-Ended POL Support, Masaaki Tsubaki, Chiyoda Chemical Engineering & Construction Co., Ltd. Yokohama, Japan.....	146
Computer Assisted Cartography and Geographic Data Bases J.G. Linders, University of Waterloo, Waterloo, Ontario, Canada.....	161
Author Index.....	169
Subject Index.....	170

ACM Order Department (\$10/Copy)
 P.O. Box 12105 Church Street Station
 New York, N.Y. 10249

Full Self-Government Bylaws of the
Special Interest Group on Management of Data (SIGMOD) of the ACM

Article I - NAME

This organization shall be called the Special Interest Group on MANAGEMENT OF DATA of the Association for Computing Machinery, hereinafter referred to as SIGMOD.

Article II - PURPOSE

1. This Special Interest Group is organized and will be operated exclusively for educational and scientific purposes in the subject area of describing information stored in and used by computing machinery and relating these descriptions to access methods and in furtherance thereof.
2. The organization will promote the interest of professionals by:
 - a. Affording opportunity for discussion of problems of common interest.
 - b. Encouraging presentation of papers of special interest to this Group at National and Regional Meetings of the ACM and at other special meetings organized by this Group.
 - c. Providing guidance to the ACM Council on matters of importance to this Group.
 - d. Publishing a bulletin containing information of interest to this Group.
 - e. Other appropriate means. No addition may contradict the main purpose as stated in Article II - 1 above.

Article III - MEMBERSHIP

Any member of the ACM in good standing may become a member of SIGMOD. Non-ACM members may belong to SIGMOD provided their major professional allegiance is in a field other than information processing or computing. No person who is not a member, associate member or student member in good standing of the Association may hold office in SIGMOD. Any non-ACM member may vote within SIGMOD and must pay required dues and assessments.

Article IV - OFFICERS

1. The Officers of the group shall be:
 - a. Chairman
 - b. Vice-Chairman
 - c. Secretary-Treasurer
2. Each Officer shall normally serve for a 2-year term beginning July 1 of odd-numbered years.

Article V - DUTIES OF OFFICERS

1. The duties of the Chairman shall include presiding at all meetings, appointing all standing and ad hoc committees, and such other duties as normally fall to this office.
2. The duties of the Vice-Chairman shall be to plan the program for all meetings and to assume any duties delegated by the Chairman.
3. The duties of the Secretary-Treasurer shall be to keep minutes of business meetings, maintain the roster of members, prepare budgets, report the Group's finances annually as required by the Treasurer of ACM, file such other reports as are required by ACM, and maintain a knowledge of the financial situation of the Group, sending official notification (two copies) to the Chairman of the Committee on Special Interest Groups and Committees of changes in the appointed officers of the Group. The responsibility for maintaining the roster of members may be delegated to ACM Headquarters.

Article VI - ELECTION OF OFFICERS

1. The Chairman shall appoint an election committee by November 30 in each even-numbered year. This committee will nominate for each elective office at least two candidates who consent to serve. All candidates must be members of the Group and voting members of ACM. Candidates may also be nominated by petition of at least ten full members of the Group. The election committee shall certify that each candidate meets ACM requirements for the office of which he is nominated.
2. By February 28 of each odd-numbered year, the vote shall be initiated by mail ballot. The balloting shall be conducted by ACM Headquarters according to the procedures of the Association for balloting. An alternative procedure may be used if prior permission is granted by the SIG/SIC Board.

3. Of the ballots returned by the specified date, a plurality of votes for each office shall determine the winner. In the event of a tie vote, the office will be filled by appointment by the Executive Committee of SIGMOD.

Article VII - MEETINGS

1. An annual business meeting of the Group will be held, normally in conjunction with the annual National Conference of the ACM.
2. Additional meetings of National or Regional character will be convened by the Chairman as he sees fit in order to serve the needs of the Group.
3. The Group may hold meetings only in places that are open to all classes of members of the ACM.

Article VIII - BOARD OF DIRECTORS

1. There shall be a Board of Directors consisting of all Officers, the previous Chairman, and ten other persons who shall be appointed by the Chairman. The editor of the newsletter will be a member of the Board.
2. The members of the Board of Directors shall serve for two-year terms. Five members shall start their terms on June 1 of odd-numbered years, and five on June 1 of even-numbered years.
3. The duties of the Board of Directors will be to advise the Chairman on all matters of interest to the Group, and to approve proposed annual dues as well as Bylaws amendments.

Article IX - DUES

Dues will be assessed annually of each member of the Group. The amount will be determined by the Board of Directors. These dues will be collected by ACM Headquarters and disbursed to the Group as warranted to meet its financial obligations.

Article X - AMENDMENTS

1. a. A resolution of the majority of the Officers of the Group shall be sufficient to cause a Bylaw amendment to be voted on by the Group members.
- b. Any member of the Group may submit an amendment to the Group Chairman. The Group Secretary shall determine whether a majority of the Officers of the Group are in favor of proposing the amendment, and shall announce this determination.

- c. A petition of ten percent (10%) of the Group members shall be sufficient to cause a Bylaw amendment to be voted on by the Group members. The right to petition shall be independent of any decisions taken with respect to the procedures provided in paragraphs (a) and (b) in Article X - 1.
2. The proposed amendment shall be reviewed prior to the distribution referred to in Section 3, below, by the Chairman of the ACM Committee on Special Interest Groups and Committees and by the Chairman of the ACM Constitution and Bylaws Committee.
3. The proposed amendment shall be voted on by the following mail balloting procedure:
 - a. The ballots shall be mailed out first-class mail from and returned to ACM Headquarters, unless the Group Secretary-Treasurer specifies an alternative mailing procedure. The ballot shall include: (1) a copy of the proposed amendment including a specification of the date on which it will become effective and (2) a copy of the Article(s) in the existing Bylaws that is (are) being proposed for amendment.
 - b. No ballot received at ACM Headquarters (or at an alternative address - if specified by the Group Secretary-Treasurer) postmarked later than thirty (30) calendar days after the postmark of the last ballot mailed out shall be valid.
4. The amendment shall become effective if and only if the following two (2) conditions are satisfied:
 - a. The number of valid ballots returned is greater than or equal to twenty-five percent (25%) of the total number of Group members in good standing at the time the last ballot mailed out is postmarked.
 - b. A majority of valid ballots returned approve the proposed amendment.

Article XI - DISSOLUTION

In the event of dissolution of the Group, all assets of the Group will be transferred to the ACM.



Association for Computing Machinery

1133 AVENUE OF THE AMERICAS
NEW YORK, N. Y. 10036
(212) 265-6300

SPECIAL INTEREST GROUP ON MANAGEMENT OF DATA - SIGMOD (formerly SIGFIDET)

Reply to:

ACM SIGMOD

ANNUAL BUSINESS MEETING

May 15, 1975

The annual business meeting of SIGMOD was held at the 1975 ACM SIGMOD Workshop, San Jose, California, on Thursday, May 15, 1975, from 5:15 P.M. to 6:30 P.M. Gene Altshuler, acting for the Chairman, presided.

Board of Directors Present:

Gene Altshuler	
Susan Brewer	
Edgar Codd	
Larry Cohn	
James Fry	
Randall Rustin	Newsletter Editor

Board of Directors Absent:

Charles Bachman	
Robert Lindner	Secretary-Treasurer
Don Moehrke	
T. William Olie	
Bernard Plagman	Chairman
Eugene Raichelson	
Kendall Wright	

1. Financial Report

The financial status of the SIG was presented:

a) FY 1975 (June 74 - June 75)

Financial position as of December 31, 1974:

Income	\$6,342.03
Expense	\$2,043.99

Surplus:

As of June 74	\$1,200.18
Income - Expense	<u>\$4,298.04</u>
As of Dec 74	<u>\$5,498.22</u>

Outside Fund:

As of May 75	\$2,818.07
--------------	------------

b) FY 1976 (June 75 - June 76):

The budget for fiscal year 1976 has been submitted to ACM Headquarters and approved. The budget operating level for FY 76 is \$11,000 (Income = Expense). This compares to:

\$6,900 for FY 75

\$5,500 for FY 74

A motion was made, and passed, to start publishing the Annual Financial Statement in the SIGMOD Newsletter.

As a result of the growing amount of spendable surplus funds, a motion was made, and passed, to have the Board of Directors address the issue as to how the surplus funds can best be spent for the benefit of all SIGMOD members. It was suggested that the surplus funds be used to lower some of the annual Workshop costs, such as Workshop fees and proceedings.

2. Membership

As of December 31, 1974, SIGMOD had 1421 members, which represents an increase of 24% in membership for the first half of FY 1975.

3. Bylaws Amendment

The Chairman described the proposed amendment to Article VI of the SIGMOD Bylaws, which would change the SIG from limited self-government (appointment of Officers) to full self-government (election of Officers).

Permission has been granted by ACM Headquarters to allow for a single mailing of the two required ballots (the Bylaws change ballot and the Officer election ballot).

4. Candidates for SIGMOD Office

The nominating committee reported the selection of candidates for the Offices of Chairman, Vice Chairman, and Secretary-Treasurer. These are:

<u>Chairman:</u>	Susan Brewer James Fry
<u>Vice Chairman:</u>	J. Gerry Purdy Diane C. P. Smith
<u>Secretary-Treasurer:</u>	John D. Joyce Stanley Y. W. Su

A motion was made, and passed, thanking the nominating committee for their fine work.

5. Newsletter

a) New Editor - A new newsletter editor will be appointed shortly and all suggestions for nominees for this position should be forwarded to the Chairman, B. Plagman.

G.M. Nijssen has been appointed European editor for the Newsletter.

b) Newsletter Format - Improvements to the newsletter format were discussed. No resolutions concerning this issue were made.


- c) 74 Workshop Proceedings - Volume II of the proceedings for the 74 Workshop are now intact and will be sent to the printers shortly.


6. Additional Agenda Items

- a) Annual Workshop Format - The following ideas were discussed as possible changes to the annual workshop format:
- 1 to 2 days of small working sessions to provide discussion over and above the presentation of papers.
 - invited papers.
 - greater emphasis on tutorials (a greater SIG educational role). In conjunction with this, it was proposed that Regional Workshops, sponsored by the local chapters of SIGMOD, be used to serve the tutorial role.
 - better coordination within sessions (i.e. the first paper an overview or tutorial followed by papers covering specifics.
 - visual standards. Standards for visuals are available from ACM Headquarters for distribution to the Workshop speakers.

As a result of the discussion it was recommended that all comments and ideas be addressed to the Chairman or Newsletter editor for consideration for next years Workshop.

- b) Papers/Proceedings - Some papers presented at the 75 Workshop are not published in the proceedings because they are to be published in the Communications of the ACM. A resolution was made, and passed, to send copies of the Communication of the ACM to those Workshop attendees who are not ACM members. It was noted, that, in the future, pre-prints of papers falling into this category could be made available to Workshop attendees.


Robert J. Lindner
Secretary-Treasurer


Bernard K. Plagman
Chairman

NEW WASHINGTON, D. C. CHAPTER OF SIGMOD HAS FULL YEAR

Beginning with only a Chairman in September 1974, the Washington, D.C. Chapter SIGMOD now has a Vice-Chairman, a Secretary, and Treasurer, and a Steering Committee of seventeen. It also has a mailing list with over 250 names. The Steering Committee has met and agreed upon a statement of purpose and bylaws for the group. An opportunity for formal membership and payment of dues will be extended to all those on the mailing list before the new Chairman takes over September 1, 1975.

The response has been overwhelming to the activities which have been held. During the 1974-75 year, D.C. SIGMOD sponsored five programs for the Washington, D.C. area. In November, we co-sponsored with the University of Maryland Departments of Computer Science and Information Systems Management a talk by Dr. Michael E. Senko of IBM on "DIAM: Data Independent Accessing Model."

Later in November, we held our first all-day workshop on "How to Choose a Data Base Management System." This workshop was conducted by Dr. David Jefferson of the Naval Ship Research and Development Center and included seven speakers in the morning and questions and discussion in the afternoon for an audience of about 90.

In February, Mr. Frank Manola of the Naval Research Laboratory and a member of the CODASYL Data Definition Language Committee presented a one-day course on "Since the CODASYL Data Base Task Group Report--Developments and Implementations." Mr. Manola and other CODASYL members present answered questions on current directions of the various committees. Attendance was about 80.

In May, a workshop was led by Ms. Elizabeth Fong of the National Bureau of Standards on "The Users Speak Out." Users of five different commercial database management systems spoke on their applications and experiences with the packages and entertained questions from an audience of 140.

Finally, the ACM Washington, D.C. Chapter Technical Symposium in June included a session on Data Management which was chaired by the SIGMOD chairman. Four papers were presented, two by D.C. SIGMOD members, and two were published in the Symposium Proceedings. The papers were:

- . "A Data Base Management System Utilizing a Small Back-end Computer" by Evan L. Ivie, Bell Telephone Laboratories, New Brunswick, New Jersey
- . "Designing Data Base Management Systems for the Users" by David K. Jefferson, Naval Ship Research and Development Center, Bethesda, Maryland
- . "A Concept for Validation of Generalized Data Base Software" by Elizabeth Fong, National Bureau of Standards, Gaithersburg, Maryland
- . "The Role of the Data Administrator" by Robert J. Tufts, Air Force Data Services Center, Washington, D.C.

The organization of a local SIGMOD chapter in the Washington, D.C. area has evidently served the needs of many who have responsibility in their organizations for decision-making in the fast-growing area of data management.

Any questions concerning D.C. SIGMOD may be addressed to one of the following:

The organizing chairman:	The 1975-76 Chairman:
Mrs. Ann Ellis Bandurski	Mr. Thomas E. Deeley
Naval Ship Research & Development Center	American Management Systems, Inc.
Code 1835	1515 Wilson Blvd.
Bethesda, Maryland 20084	Arlington, Va. 22209
(202) 227-1533 or 227-1618	(703) 841-6051

INTERNATIONAL SYMPOSIUM
TECHNOLOGY FOR SELECTIVE DISSEMINATION OF INFORMATION
PALAZZO DEI CONGRESSI DI S. MARINO • SEPTEMBER 8 - 10, 1976

CALL FOR PAPERS

The increasing impact of computer stored information in industrial, governmental and educational institutions emphasises the need for improved computer and communications technology for building and accessing computer-processable data bases.

Papers are solicited exploring current research and developments. The following is a list of suggested (but not exclusive) topics to be covered during the Symposium:

Data Networks	Automation in Offices
Transmission Technology	Computer Aided Document Illustration
Information Systems	User Oriented Systems and Interactive Graphics
Data Management: Systems and File Design	Computerized Cable Television
Distributed Data Bases	Technology of Graphics Terminals
Human Factors	Audio Input and Output
Cost Performance Evaluation	Computer Aided Education
Cost Benefit Analysis	Computers in Humanities
Computers and Libraries	User Machine Dialogues in Interactive Problem Solving.

IMPORTANT DATES

January 30, 1976	Draft paper due for review and referee (3 copies) Papers should not exceed 20 double-spaced typed pages in length.
March 30, 1976	Acceptance notification
May 30, 1976	Final version of paper due in camera ready form for publication in the Symposium Proceedings.
September 8 - 10, 1976	Symposium meets in S. Marino at the Palazzo dei Congressi.

Papers or inquiries should be addressed to the Publication or Program Chairman. Reduced group air fare will be arranged by Alitalia for Symposium attendees.

SYMPOSIUM COMMITTEE

Symposium Chairman

Prof. Giancarlo Corazza
Università di Bologna
Istituto di Elettronica
Viale Risorgimento 2
40136 Bologna, Italy

Registration Chairman

Prof. Aureliano Casali
Istituto di Cibernetica
Palazzo degli Studi
Repubblica di S. Marino

Dr. Louis Pouzin
Reseau Cyclades
IRIA
78150 Rocquencourt, France

Program Chairman

Dr. Giorgio Valle
Università di Bologna
Istituto di Elettronica
Viale Risorgimento 2
40136 Bologna, Italy

Dr. Lynn Hopewell
Network Analysis Corporation
9105 Westerholme Way
Vienna VA 22180, USA

Dr. James Fry
Business Administration School
The University of Michigan
Ann Arbor Mich 48104, USA

Publication Chairman

Mr. Ira Cotton
National Bureau of Standards
Institute for Computer Science
and Technology
Washington D.C. 20234, USA

Prof. Peter Kirstein
University College London
44 Gordon Square
London WC1H 0PD, England

Prof. Giancarlo Martella
Istituto di Elettrotecnica ed Elettronica
Politecnico di Milano
20100 Milano, Italy

1976 ACM - SIGMOD: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA
June 2-4, Washington, D.C.

Session I - Performance and Physical Design Issues (June 2, 9-12:30)

Chairman: Professor Howard Morgan; University of Pennsylvania

Index Selection in a Self-Adaptive Data Base Management System
M. Hammer, A. Chan; MIT

Application of an Analytical Model to Evaluate Storage Structures
T. J. Teorey, K. S. Das; University of Michigan

Retrieval Using Ordered Lists in Inverted and Multilist Files
J.W. Welch, J.W. Graham; University of Waterloo, Canada

Performance of a Data Base Manager in a Virtual Memory System
S. W. Sherman; Institute for Computer Applications in Science and
Engineering; R. S. Brice; George Washington University

Session II - Recovery, Concurrency, and Protection (June 2, 2-5:30)

Chairman: Professor Michael Stonebraker; University of
California, Berkeley

Data Base Recovery at CMIC
N.J. Giordano; Pennsylvania Central Management Information Center
M.S. Schwartz; Chi Corporation

Differential Files: Their Application to the Maintenance of Large
Data Bases
D.G. Severance; University of Minnesota
G.M. Lohman; Cornell University

Deadlock Detection and Resolution in a CODASYL Based Data Management
System: P.P. Macri; Bell Laboratories

An Authorization Mechanism for a Relational Data Base System
P.P. Griffiths, B.W. Wade; IBM

Session III - Logical Design (June 3, 9-12:30)

Chairman: Professor Terry Hardgrave; University of Maryland and
Institute for Computer Applications in Science and Technology

A Method for Describing the Information Required by the Data Base
Design Process; B. K. Kahn; University of Michigan

Files With Semantics N. Minsky; Rutgers University

A Relational View of the DIAM
L. S. Schneider; Martin Marietta Data Systems Division

A Constructive Definition of Third Normal Form
G. C. H. Sharman; IBM United Kingdom Laboratories, LTD.

Session IV - Languages (June 3, 2-5:30)

Chairman: Dr. Donald Chamberlin; IBM

An Architecture for High Level Language Database Extensions
C. J. Date; IBM

LSL: A Link and Selector Language
D. Tsichritzis; University of Toronto

NUL: A Navigational User's Language for a Network Structured Data Base
C. Deheneffe, H. Hennebert; Institut d'Informatique, Belgium

A Generalized Access Path Model and Its Application to a Relational
Data Base System
C. R. Carlson, R. S. Kaplan; Northwestern University

Session V - Automatic Design and Optimization (June 4, 9-12:30)

Chairman: Professor Alan Merten; University of Michigan

Decomposition - A Strategy for Query Processing
E. Wong, K. Youssefi; University of California, Berkeley

Schema and Occurrence Structure Transformations in Hierarchical Systems
A. G. Dale, N. B. Dale; University of Texas at Austin

Efficient Exploration of Product Spaces
R.M. Pecherer; Washington State University

An Approach to the Creation of Structured Data Processing Systems
L. L. Beck; Southern Methodist University

A Business Application of Data Translation
E. W. Winters, A. F. Dickey; AT&T Long Lines

EVENING SESSIONS: June 2 and 3, 8 - 10:00 p.m

1. Impact of Privacy Legislation on Data Base Systems: Panel
Discussion

2. Seminar on Experimental Systems: Discussion of systems on
display at the conference

3. Birds of a feather (SIGRELAX) refreshments

1. Reports from CODASYL and ANSI Data Base Committees

2. Panel Discussion on Data Base Research Funding

3. Birds of a feather (SIGRELAX).

CHECKPOINT, RESTART, AND RECOVERY:
SELECTED ANNOTATED BIBLIOGRAPHY

A.B. Tonik - Sperry Univac
P.O. Box 500 - Blue Bell, Penna 19422

1. Daley, R. C., Neumann, P. G., "A General-Purpose File System for Secondary Storage", FJCC, 1965.

This paper outlines the file structure that was proposed to be built into the MULTICS system at MIT. Users address files thru a file name directory. This directory has a tree structure. Every several hours or when a user signs off two copies of his files are made on tapes. Once a week the system files are dumped onto two tapes. After a catastrophe the file directory is examined. If it points to files that exist and not to the same place for two files, then O.K. If not, the system files are loaded and files reconstructed from the hourly tapes.

2. Day, William J., "Rio Grande Message Switching/Transportation System", ACM Conference, 1968.

The Denver and Rio Grande Western Railroad installed a system of 60 teletype machines to input data about freight cars and output messages to alert the stations about the disposition of freight cars. All messages are collected in duplicate on two disk units. If one disk is down the messages go to the other. All tables are written every 20 seconds to a snapshot area on disk.

3. Lockemann, Peter C., Knutsen, W. Dale, "Recovery of Disk Contents after System Failure", CACM, Aug. 1968.

Cal Tech has a data acquisition system. They record files on disk in fixed length records. Each record within a file has a forward pointer to the next record. Each file is given a unique number. This number is also recorded in every record of the file. The recovery routine traces every file until the end or a new file number and determines free space.

4. Oppenheimer, G., Clancy, K. P., "Considerations for Software Protection and Recovery from Hardware Failures in a Multiaccess, Multiprogramming, Single Processor System", FJCC, 1968.

This paper is a tutorial on the subject of recovery of information on mass storage. They discuss practically every aspect of preserving information and how to use it in recovery.

5. Fraser, A. G., "Integrity of a Mass Storage Filing System", The Computer Journal, Feb. 1969.

Cambridge University has an interactive system similar to Multics. They have several stoppages per day and 50% of the files are less than 4K characters. Every 20 minutes a check is made for an updated file. It is written on two tapes. When the tapes are full, new ones are started. The beginning of each tape has the file directories and other administration information. Each file

on disk has the tape identification of the latest copy. On the disk whenever the administration information is changed, it is written into an alternate area.

6. Ishizaki, Sumio, "Designing a Large-Scale On-Line Real-Time System", SJCC, 1971.

The Fuji Bank, Ltd. of Japan has a connected system with a large data base in Tokyo and Osaka and 1000 teller terminals in 200 branches. They handle deposits and withdrawals and remittances from one account to another. They have 6M accounts on drums with 650K transactions per day. All transactions are recorded on tape in time sequence. If an account can not be read, it is recovered from tape in 10 to 20 minutes. At night a copy of the drum files is recorded on tape. If a drum breaks, yesterday's file is recovered from tape and brought up to date from the transaction file. Also, during the night, a complete list of updated deposits is compiled and printed (4 hours) and put at each teller window by 9 A.M. in case system breaks down.

7. Isbell, Ira B., "Automated Management Information System for California Department of Motor Vehicles", ACM Conference, 1971.

They have a file of 14.5M drivers and 20M vehicles, with 90K updates per day. Every day a part of the file is dumped on tape, with the entire file being dumped once a month. The dump takes place while the system is on-line. Therefore, every block is time stamped. Every day log tapes are prepared with the after-images of updated blocks with time stamps. Every day an index tape is prepared showing the location on tape of the latest image of each block. An average of about 10 blocks a day are unreadable in the on-line data base. However, the recovery run is only performed about 3 times per week.

8. von Gohren, Gerald L., "Pillsbury Database Recovery", ACM Conference, 1971.

Pillsbury's 4 divisions has a database of 400M characters. Up dating is only done in batch mode. The files are dumped 4 times per day. These are stored for several days; those at the end of the day for 2 weeks, those at end of month for 13 months, those at end of year for 2 years. Per week there are 6 job aborts with each taking 1.5 hours to reprocess; 0.5 erroneous updates from wrong input with each taking 3 hours to correct; and 2.5 system failures with each taking 0.25 hours to reboot. Annual cost of dumps is \$86K and recovery is \$69K.

9. Boyd, Ralph, "Restoral of a Real-Time Operating System", ACM Conference, 1971.

The Ohio Bell System has a data base of 100M characters. They process 9K transactions from 8 A.M. to 11 P.M. Most of these are service orders with about 200 file accesses. They create an audit trail on tape of beginning of a transaction, before images, after images, and end of transaction. At the end of the day they dump

the files on tape. If during the processing of a transaction, an inconsistency is discovered, the transaction is unwound by finding all the before images from the audit trail. There are 250 of these per day and require about 5% of processing time to recover. A system failure requires either short, medium, or long restoral. The short restoral unwinds all incomplete transactions. This takes about 25 minutes and occurs once a week. Medium restoral restores all files to their condition as of a specific time, the trouble is fixed, and the transactions are reprocessed. This takes an hour and is used every two months. Long restoral reloads the data base as of a certain day and then processes transactions from the audit trails. This is used once in 2 years and takes 6 hours.

10. Grafton, William P., "Database Recovery with IMS/360", ACM SIGBDP DATABASE, Spring, 1972.

North American Rockwell uses a database to control their engineering production. They have 200 terminals and process 25K transactions per day. They avoid the monolithic database for easier recovery. They have 50 databases ranging from 300K to 80M bytes with a median size of 3M. Every thing that is changed in core is back-uped by storing in non-volatile storage. A log is kept on tape of every transaction and before and after images of changed records. After 5K entries (about 10 minutes) processing is suspended and the system control tables are recorded on the log tape. Every 5 hours they have to restart. The log tape is backed up to the check point, the system tables are restored, and the transactions on the log are reprocessed. This takes 15 minutes. Once every week or two, a database is dumped on tape. About every 3 or 4 months, a data base has to be recovered. The latest copy of the database is read from tape and the log tapes bring it up to date.

11. Fichten, J. P., "The Weyerhaeuser Information System - A Progress Report", FJCC, 1972.

Weyerhaeuser has installed a Management Information System for the entire company. They can store as many as 32B characters. They have 100 teletypes that handle 900 jobs per hour. When a data-base is processed, after images of pages (2K characters) are recorded on tape. If a page can not be read, it is recovered from this tape. If a program is halted because of a detected malfunction, the altered data base is restored to the condition before the program started. Every 3 minutes, snapshots of the system are recorded on tape. Restart will recover this control structure and reprocess all jobs. Recovery time for a system fault is 12 minutes and down time is less than 1%.

12. Schneiderman, Ben, "Optimum Data Base Reorganization Points", CACM, June 1973.

Assume a file where records can be inserted and deleted. The

insertions are via overflow records. The access time for records in this file will increase with time. A recopy of the file will reduce access time. Here is a formula for the optimum time to reorganize the file based on file size, number of insertions, and number of accesses, and type of file structure. Some of the formulas are approximately:

Time to Reorganize the Database

Increase in Access Time

It is stated that the reorganization could occur when a back-up copy of the database is created.

13. Computerworld, June 27, 1973, Page 6.

One of the papers in the IBM-Telex trail was a user survey to the Management Committee in 1969. About 270 users were using their system over 400 hours per month. They had 150 interrupts per month with 44 hours of lost time and 26 hours of rerun time.

14. Young, John W., "A First Order Approximation to the Optimum Check-Point Interval", CACM, September 1974.

Here is a formula which minimizes the time to create enough information to rerun and the restart time:

Interval between Checkpoints = $\sqrt{2(\text{Time to create})(\text{Interval between Failures})}$

An installation runs 144 hours per week with 2 partitions and 19.5 errors. The time to catalog a data set is 7.5 seconds and two data sets are catalogued at checkpoint time. Therefore, check pointing should be done about every 21 minutes.

15. Rappaport, Robert L., "File Structure Design to Facilitate On-Line Instantaneous Updating", ACM SIGMOD Conference, 1975.

The Department of Transportation of Puerto Rico has a motor vehicle data base. The driver file is 1M records and the vehicle file is 800K records. There are 70 terminals and less than 5K updates per day. The files are reorganized every night. This paper shows how to organize the file and the sequence of operations so that when the system crashes, the file will not have inconsistencies. A modified record is not put back into the original file but into a modified file. The modified record points back to the original. A remodified record points back to the modified. An index for the modified file points to the latest copy of the record. As transactions come in from terminals they are assigned consecutive numbers. As transactions are completed, the transaction number is added to the transaction complete file. In all cases new items are inserted into files before pointers are changed.

16. Sayani, Hasan H., "Restart and Recovery in a Transaction-Oriented Information Processing System", ACM SIGMOD Workshop on Data Description, Access and Control, May 1974.

A transaction oriented system needs to record different types of data and needs to run different procedures to recovery from different types of errors. The author attempts to minimize the cost of recovery by minimizing both the cost of recording the data needed for recovery and the cost of running the restart procedures based on the frequencies of the different errors and the amount of time available on the resources of the system. This is a non-linear programming problem written in FORTRAN. An example is presented for a parts manufacturer. They run every day, 13 hours per day. This time does not include taking dumps, or logging input and output messages, or before and after images, etc. The transactions per day are: 100,000 input messages, 10,000 changes, 1000 output messages. The errors per month are: one loss of data base, 5 losses of main memory, 20 user input mistakes. The optimum solution requires an extra 9½ hours per day to record the data needed for recovery. It is not clear how much of this time can be overlapped with the 13 hours of running time. To recover from a user error, they restore the data base to before the bad transaction by means of before images and reprocess the input messages with suppression of output messages. To recover from a loss of main memory, restart with last memory dump and rerun input messages without affecting data base. To recover a data base, they reload the data base from the last dump. This takes 2 hours. They rerun the input messages.

* * * * *

SECOND INTERNATIONAL CONFERENCE ON
VERY LARGE DATA BASES 

Program Committee Members:

Professor M. Brockhaus
Technical University of Vienna
Austria

Professor C. Delobel
University of Grenoble
France

Dr. G. G. Dodd
General Motor Corp.
Warren, Mich., USA

Dr. K. Eswaran
IBM Research
San Jose, USA

Dr. D. W. Fife
National Bureau of Standards
Washington DC

Dr. E. I. Lowenthal
MRF Systems Corp.
Austin, Texas, USA

Dr. J. S. Knowles
Aberdeen University Computing Centre
GB

Professor T. L. Kunii
University of Tokyo
Japan

Dr. Heinz Schappert
Bayer AG
Laverkusen, W. Germany

Dr. H. A. Schmid
University of Stuttgart
FRG

Professor D. Tsichritzis
University of Toronto
Canada

Professor R. Yeh
University of Texas
Austin, USA

Professor C. A. Zehnder
ETH Zurich
Switzerland

Treasurer:

H. Biller
University of Stuttgart
Stuttgart, W. Germany

Local Organization Chairman:

G. M. Nijssen
Control Data Corp.
Bruxelles, Belgium

Registration Chairperson:

Ms. Maria Briers
Control Data Corp.
Bruxelles, Belgium