CROSSTALK
Journal of Hewlett-Packard Technical Computer User Groups
July/Aug. 1983

NEW MICROSYSTEM PACKAGING FOR THE A-SERIES

What product offers a choice of HP 1000 A-Series processors, optional integrated 9.4Mb Winchester and 270Kb microfloppy discs, support of the 1206X measurement and control cards, and comes in a box that can be rack-mounted, placed on a table, or on a wheeled stand, beside a desk? The answer is — HP's new microsystems.

With all of the above in one package, you might think that the package must be very big. But, the new microsystem box is only 19" × 7" × 25.5" — over 30% smaller than the 20-slot box.

The new microsystems include an HP 1000 A-Series processor, memory, RTE-A, a terminal interface, and an HP-IB card. The Micro 26 has an A600+ processor, with 512Kb memory. The Micro 27 has an A700 processor, also with 512Kb memory, and the Micro 29 has an A900 processor, with 768Kb memory. For OEMs, the microsystems are available in component configurations (without RTE-A and the interface cards), so they can be tailored to suit any application. The A600+ microsystem has an "execute-only" system which includes the A600+ processor, 512Kb memory, and RTE-A execute only, housed in the microsystem package for 25% less than the 20-slot box. This "execute-only" system is perfect for OEMs integrating the A600+ into their products.

Coupled with RTE-A, the new microsystems offer a powerful, low-cost system for real-time applications. The compact microsystem package fits into environments where space is limited. With built-in flexibility, the microsystems offer you a range of performance in the same physical dimensions. Software developed on one of the microsystems will run on the other two with no changes.

NEW MICROSYSTEM VALUE PACKS OFFER BIG SYSTEM PERFORMANCE AT AN INCOMPARABLE VALUE

The new A-Series Microsystems offer three software value packs to give you big system performance in a small package. Each value pack is available on linus cartridges or microfloppy discs, and includes a choice of BASIC/1000C, Pascal, or FORTRAN 77, with Symbolic Debug; VC+, IMAGE/1000, and DGL Graphics, plus extra memory. The Micro 26 includes an additional 512Kb memory, for a total of 1Mb. The Micro 27 deletes the standard 512Kb and adds a 1Mb card, while the Micro 29 includes an additional 768Kb, for a total of 1.5Mb of memory.

CONTENTS

PUZZLE PLACE

Thank you to those readers who have taken the time to submit solutions for the last two puzzles. I have selected solutions from two readers, and to those whose solutions are not published here — thanks for your inputs and I shall be glad to publish your solutions in the future.

The solution to the first puzzle comes from Dr. P. Havlicek, Dept. of Applied Chemistry, Swinburne Institute of Technology. The second solution comes from John Geremin, Megatronics, N.S.W. Here are their solutions:

Puzzle No. 1

Answers: 4 and 13

Analysis: S knew sum was 17. The possible no. pairs were 2,15; 3,14; 4,13; 5,12; 6,11; 7,10; 8,9.

Since any pair produces a product of at least 3 prime factors, it is obvious P could not know the answer (N.B. P would know the answer if the product was that of 2 primes or of a prime greater than 50 and any other number).

S's statement that P could not know the answer tells P that:

(a) The sum is odd (any even number is the sum of 2 primes).

(b) Therefore the sum — 2 is not a prime, since 2 × prime is a unique solution which would be known to P.

Looking at possible solutions:

```
2 X 15    P=30    =2x15 or 3 x 10 or 6 x 5
                  2 possibilities P can see : 2 x 15 or 6 x 5
                  (3 x 10 gives sum =13 = 11+2)
                  So P could not tell

3 x 14    P=42    =2 x 21, 3 x 14, 6 x 7
                  2 possibilities 2 x 21, 3 x 14

4 x 13    P=52    2 x 26, 4 x 13
                  Single possibility

5 x 12    P=60    2 x 30, 3 x 20, 4 x 15, 5 x 12, 6 x 10
                  2 possibilities 3 x 20, 5 x 12

6 x 11    P=66    2 x 33, 3 x 22, 6 x 11
                  2 possibilities, 2 x 33, 6 x 11

7 x 10    P=70    2 x 35, 5 x 14, 7 x 10
                  2 possibilities 2 x 35, 7 x 10

8 x 9     P=72    2 x 36, 3 x 24, 4 x 18, 6 x 12, 9 x 8
                  2 possibilities 3 x 24, 9 x 8
```

Since P knows the answer, S also now knows the answer.

Computer Solution

a) Generate a list of all possible sums compatible with S's statement.

b) Select a set of primes $p_1, p_2, p_3, \ldots p_n$ ($n \geq 3$) such that their product is factorizable in at least 2 ways and the factors are both less than 100 and one is odd, the other even.

c) Examine all combinations of factors. If the sums of any combination in the set of primes (b) are contained in the list of sums (a) 2 or more times, eliminate the combination.

Also if any sum in list (a) is accessed from two different combinations, eliminate the sum from consideration.

d) The only combination remaining is 2,2,13. This is true even if numbers are in the range up to 200.

Dr. P. Harlicek,
Dept. of Applied Chemistry,
Swinburne Institute of Technology

Puzzle No. 2

Since A is even so are D,B and also G

Therefore MUST BE 2,4,6 and 8 But in what order.

Similarly E is ODD as is K.C,F, (J already)

```
Hence A+G = B        )
      G+G = D        )   if all even
Then  G=4, D=8, B=6, A=2
For
        2 6 5
      ________
EC2  ) F8605
       C44
       2420
       22E2
        K885    So far, so good
        K860
          25
E=12-8-1 = 3
K=1
C=7 & F=9
```

JOHN GEREMIN

And now for this month's puzzle, you should find it a little simpler than the last two — I shall mix up the degree of difficulty! Keep those solutions rolling in.

NEW PUZZLE:
Hercules Poirot, the famous amateur detective, had assembled servants and guests in the drawing room of the manor house. He was about to reveal who was responsible for the murder of the Lord of the manor last week. After summarising the case he concluded:

"Clearly, if the admiral is telling the truth, then so is the butler. However, the butler and the count cannot both be telling the truth. But the Count and the Deacon are not both lying. And if the Deacon is telling the truth then the butler is lying." There are two people whose truthfulness — or lack of it — you can deduce. Who are they?

Philip Greetham

Focus 1000

IMAGE USER'S WORKSHOP

Following our highly successful PASCAL workshop last year, the Victorian HP1000 User's Group is pleased to announce a similar workshop, this time on the ins and outs of HP's IMAGE database products.

Aimed at any and all users of IMAGE (including desktop and HP3000 system users), it is planned to cover the position of, theory of, use and abuse of, and user's experience with, IMAGE databases in all types of applications. As such, I am sure you will derive great benefit from attending.

Whilst further details will be mailed to you shortly, please note the following, particularly the date.

VENUE: HP Melbourne
DATE: Wednesday, 28th September, 1983
TIME: 9.00 a.m. to 4.30 p.m.
COST: $25.00 - $30.00 (estimate only, includes lunch)

Be sure to book early when the entry form arrives!
See you there!

John Gwyther
President, Vic. HP1000 Users Group
Phone (03) 859 9487

INTRODUCTION TO RELATIONAL DATABASE TECHNOLOGY

(Reprinted by permission from Interface 1000)

Bill Hasling
Computer Resources Incorporated
5333 Betsy Ross Drive
Santa Clara, CA 95054

Abstract: The author discusses the pros and cons of hierarchical, network, and relational database structures.

During the last several years, few topics have received as much attention in computer trade publications as relational database theory and technology. But few topics remain as confusing or misunderstood. The mystique surrounding relational technology has led many manufacturers to claim their products to be relational or relational-like. These claims have only added to the confusion.

Relational database management systems (DBMS) have been proclaimed to be powerful, mathematically elegant, even futuristic. The major scientific publications about database systems are each year increasingly devoted to discussing the current research problems in the context of the relational data model. Powerful relational database systems are appearing for the IBM environment (System/38, SEQUEL/DS, QUERY BY EXAMPLE) and for the DEC environment (ORACLE and INGRES). Computer Resources Incorporated (CRI) introduced a relational database system for the HP 3000 and rumors persist that Hewlett-Packard (HP) will announce a relational database management system for one of their computers at any time. As relational databases become practical tools instead of research projects it becomes more vital to understand this important subject.

This paper will discuss several common questions concerning relational technology. I'll first define database systems, discuss the three major database models, and then compare these systems to the relational model. A thorough understanding of the differences among the existing database models will help foster an appreciation for the simplicity and flexibility of the relational database design.

What is the Difference between a Data Model and a Database System?

A database management system is a tool for organizing and manipulating data within a computer. Almost all software performs some sort of data management function, but a DBMS provides a structure to the data organization that allows many unrelated users, or programs, to manipulate and use a common collection of data.

A data model is the method used by a DBMS to organize and retrieve the data. There are three basic categories of data models: network, hierarchical and relational. A DBMS may combine features of all three or use only a subset of one or more of the models. This may be a source of much confusion if a company claims that it sells a relational or relational-like DBMS when, actually, it has only a few of the features associated with the relational data model. As described later, each of the models supplies different characteristic benefits and drawbacks to the database management system.

What is the Value of a DBMS?

When does a DBMS provide some value over simply putting data into a disc file and retrieving it? The major benefits of a database system are concurrent access of data by several users, reduced data redundancy, data independence, data security, and database utilities.

A DBMS allows different users to access the same data according to their own needs without requiring their own copy of the data or information database. The DBMS forces a standard on the representation of the data, and acts as a data independent window that can insulate users from other users' needs for the same data. As an analogy, a DBMS is to data what an operating system is to the machine. It allows several users to concurrently access the same resources, checks for inconsistencies, applies security restrictions, maintains integrity, and provides utilities to assist the user in effective use of the DBMS.

What Type of Data Can Be Represented in a DBMS?

This may seem to be a naive question at first. After all, you can put anything into a database, right? Actually, there are many types of data that are difficult to represent even using state-of-the-art DBMS concepts. Currently, database systems store information as entities and relationships between entities. Pictures, abstract ideas or spacial information cannot yet be stored using typical database systems. An entity is an object represented in the database —usually a record of data: a part, supplier, project, person, department, etc.

A relationship is a logical connection between two or more entities; for example, the fact that supplier SMITH supplies part FRAME, or person JOHN belongs to the SALES department. It turns out that this structure is sufficient for most data management problems. Therefore, we will not concern ourselves here about trying to represent abstract data.

Both entities and relationships can be considered facts known by the database system. Entities may contain many associated pieces of information.

A part may have a name, number, description, cost. Entities are straightforward and are represented in all three data models as files. The associated pieces of information of an entity are fields in a file. (For the remainder of this paper, entities and files will be used interchangeably.)

All three data models represent entities in the same fashion. Relationships, on the other hand, are represented differently in each. The different methods force restrictions as to the type of relationships that can be represented, and the way data can be retrieved.

Relationships may be one-to-one, many-to-one or many-to-many. A many-to-one relationship is the most common, such as the relationship between person and department. A single department may contain several people, but a person may belong to only one department. An example of a many-to-many relationship would be the relationship between students and courses. A single student may be enrolled in several courses and a single course usually contains many students. Relationships may also involve more than two entities, as the relationship between parts, suppliers and projects. A single supplier may supply many parts to many projects.

As the relationships among the data increase in number, the database becomes very complex. The method that the DBMS uses to represent the relationships and to manipulate the data affects the productivity of the end user. We will discuss how each of the three data models represent relationships and why some are superior for different kinds of relationships.

What are the Characteristics of Each of the Three Data Models?

To answer this question, we will use a simple database—university enrollment—as an example and represent this database with each data model. We will then discuss the features associated with each model and its advantages and disadvantages. The database contains students, courses, and a relationship between students and courses (which students are taking which courses). In the example, students have a name and an age, and courses have a name and a room number.

Hierachical Data Model

The oldest of the data models is the hierarchical model. Given the problem of representing relationships between entities, the hierarchical approach is the most straightforward. Data is organized as a file box or on a paper listing. The hierarchical model is characterized by a hierarchical ordering of the files. This ordering determines the relationship between the files and the preferred access path for finding the data. The hierarchical diagram (see Figure 1) shows the two different ways of organizing our database: one is grouped by students, the other is grouped by courses. Searching a hierarchical database can only be efficiently performed from top to bottom. In the first case A, it is simple to find out what classes BOB is taking, but hard to find out who is taking PHYSICS. The opposite is true in the organization B; it is simple to find out who is taking ENGLISH, but hard to find out what classes BOB is enrolled in. With a hierarchical data model you cannot have it both ways. You must decide how you want to access your data and pay a large efficiency penalty for asking the wrong questions. The hierarchical model is a simple view of a database that works very well for some database problems, but lacks flexibility.

All three data models represent entities in the same fashion

Data retrieval inflexibility is only one undesirable property of the hierarchical model. It also can be space-inefficient because of data redundancy. In example A, two students are enrolled in PHYSICS. All of the data associated with PHYSICS is duplicated in the BOB record and in the MARY record. Data redundancy is not only inefficient in terms of storage, but also allows the possibility of changing the room for PHYSICS in one record and not in the other. This can cause the database to become inconsistent and outdated.

The data-redundancy problem shown in this example exists because the student/course relationship is a many-to-many relationship. The hierarchical data model does not deal with many-to-many relationships very well; however, it works effectively on many-to-one relationships. Because the many-to-one relationship is the most common, the hierarchical data model is frequently used with success.

BOB 22	PHYSICS	ROOM A
	MATH	ROOM B
	ENGLISH	ROOM C
FRED 21	ENGLISH	ROOM C
	PHILOSOPHY	ROOM D
MARY 27	PHYSICS	ROOM A
	ENGLISH	ROOM C

(A)

PHYSICS	**ROOM A**	BOB 22
		MARY 27
MATH	**ROOM B**	BOB 22
ENGLISH	**ROOM C**	BOB 22
		MARY 27
		FRED 21
PHILOSOPHY	**ROOM D**	FRED 21

(B)

Figure 1: *Example of a hierarchical data model.*

One of the main purposes of a DBMS is to allow many users and applications to view a database in different ways—this requires flexibility. The hierarchical model does not provide this flexibility. For this reason, more flexible data models have evolved. These particular problems inspired the network data model concept.

Network Data Model

The network model offers several improvements over the hierarchical model. The network model solves the data redundancy problem and the problem of representing many-to-many relationships. In the network model, each entity is represented as an independent file and the relationships between files are represented by links. Usually, each link is given a name and provides a connection between two files. In the network diagram (see Figure 2), there are two links: the student/course link, and the course/student link. The links are implemented in a network database as pointers, hash tables, or indexes, but in all cases are specified when the database is designed and therefore cannot be easily changed.

IMAGE is based upon a restricted network model

The standard network model has several good features. First, redundant data is not stored as in the hierarchical model. Second, data can be accessed efficiently in many different ways, limited only by the foresight of the database designer. The complete network model allows links between any files in the database. This allows any type of relationship to be easily represented within the model.

Although the network model provides a vast improvement over the hierarchical model, there are several disadvantages. First, although any type of relationship can be represented, it is very difficult to add, change, or delete relationships from a network database. It is also difficult to add, change, or delete files from a database because of the tight binding between files and relationships. The network model offers the power to represent many different types of relationships, but the tight coupling between relationships and the associated files requires a major operation to make even minor changes to the data structure. For applications that are well understood when the database is designed and do not change over time, the network model is excellent.

The network model can be a burden when used for applications that are only partially understood when the database is designed, or for applications with requirements that change with time—such as applications that start small and slowly enlarge.

The second major disadvantage of the network model is that the designer must name the links in the model and normally the names must be specified by the user when querying the database. This makes navigating the database complex and data-dependent. To ask a simple question involving a relationship, "Find all students taking PHYSICS," requires deciding whether to use the student/course relationship or the course/student relationship. No matter which choice is made, the network model requires a procedural description of how to navigate the database. The procedure to find all students taking PHYSICS is:

```
1. Find PHYSICS record in
   the course file.
2. Follow course → student
   link to the student
   record.
3. Print student record.
4. Go to 2 until all
   students are found.
5. End.
```

A similar procedure can be written to answer the same question by following the student/course link. Usually, one of the available choices is more efficient, and it is up to the database user to decide which one.

You will notice that the above procedure looks much like a computer program. This is characteristic of the network model: The link pointers of the network model are crucial to the data represented in the database. Due to the procedural nature of queries using the network model, it is difficult to construct a high-level front-end command interpreter. The network model is more at home with record-level manipulation than those of a higher level (for example, "Give me records with this property..."). The relational model solves these remaining difficulties.

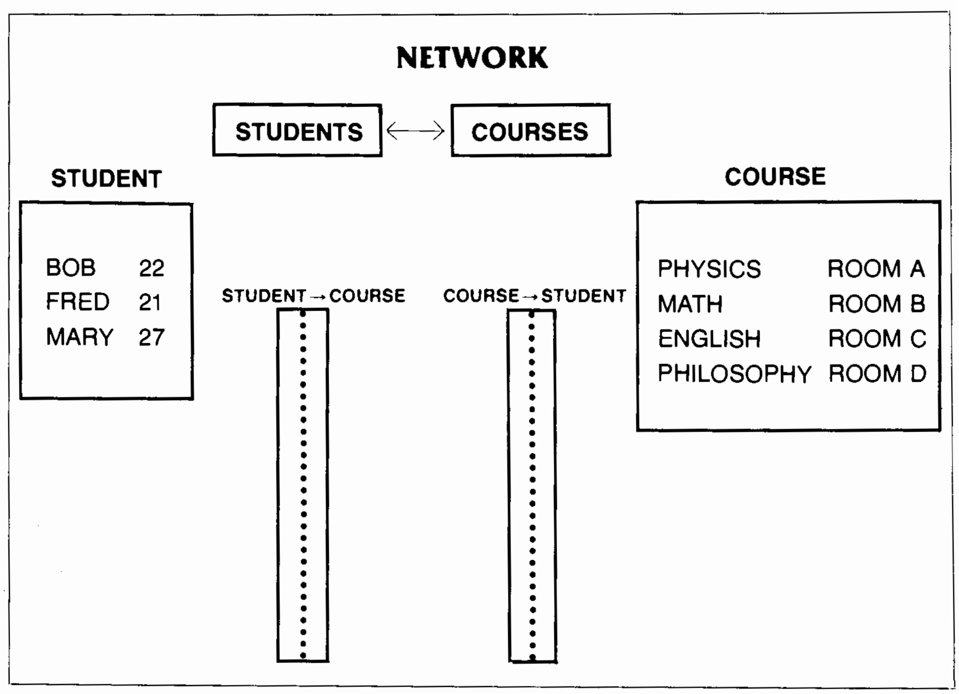

Figure 2: *Example of a network data model.*

Relational Data Model

A true relational DBMS is a system that uses the relational data model for representing relationships and the relational access method for retrieving data.

In a recent letter to Computerworld, E. F. Codd defined what makes a system relational:

> A true relational system requires all information to be represented at the logical level as values in tables. There must be no user-visible navigation links between tables and the system must support (in some form) at least the select, project and join (natural or equijoin) operators of the relational algebra.

More simply, a relational system:

1. Is made up only of flat files, i.e., tables.
2. Allows the user to pick any fields from any file for data manipulation (project operator).
3. Allows the user to select any set of records for manipulation, subject to some condition (select operator).
4. Allows two or more files to be joined into one logical file where fields from records in each file match (join operator).

The relational data model has a strong mathematical basis. Literature on relational models discusses the select, project, and join operators–mathematical terms used to define the operators of the relational model. The join operator is the key to the relational model. As an example of a join operation, we can join the CLASS file and the COURSE file (see Figure 3) by matching the course-name field. The result would be a logical file containing the following data:

BOB	PHYSICS	ROOM A
BOB	MATH	ROOM B
BOB	ENGLISH	ROOM C
FRED	ENGLISH	ROOM C
FRED	PHILOSOPHY	ROOM D
MARY	PHYSICS	ROOM A
MARY	ENGLISH	ROOM C

Following the join operations, we can print all the classrooms BOB attends by further selecting only the records with the name BOB. By using the select, project and join operators, any arbitrary, complex question can be answered.

The relational model appears to be similar in some ways to the network model, except that relationships, represented in the network model by links, are represented in the relational model as another flat data file.

In our example, the new data file contains a field from each of the entity files (in the class file, student name and course name). Each record in this new file establishes a link between the two entity files. This new file is called a relation.

If we take a second look at Figure 3, we can see that our new relation appears identical to the other entities in the database: They are all simply two-dimensional files with fields. The beauty of the relational model is that relationships are stored simply as another entity in the database. The link between the entities is logical, not physical. This loose coupling between files allows much greater freedom to add, delete and change the database structure. New files can be added and new relationships created without affecting the existing database. Changes can be made to the structure of a file without affecting the rest of the database. The database may be designed slowly, one file at a time, or modified to change with a changing environment.

The relational model provides the same representational power as the network model, but provides additional dynamic control over the database. It is simple to create new relationships or add new files to the database. This increases the flexibility of the system, which in turn makes the relational model more powerful than either of the other two models.

The second major advantage of the relational model is that the user need not specify the most efficient path to find the data. In the network model, to find all the students taking physics we need to specify that we want to use the student/course link or the course/student link to join the files. In the relational model, we specify that the class file is to be joined with the student and course files and leave the job of deciding how best to perform the join to the DBMS. This capability allows the user to issue high level queries. For example, the high level command to find all students taking physics would be:

```
SELECT STUDENT WHERE CNAME="PHYSICS" AND
                     CLASS.SNAME=STUDENT.SNAME AND
                     CLASS.CNAME=COURSE.CNAME
```

The construct CLASS.SNAME refers to the SNAME field in the CLASS file. The two conditions

CLASS.SNAME = STUDENT.SNAME
and
CLASS.CNAME = COURSE.CNAME

are called join conditions and specify how the files should be logically linked. The join conditions can be made invisible to the end user or manually applied as in this example. This high-level query is nonprocedural. In relational terminology the query creates a set, a view, or a virtual file. This virtual file is itself a relation, illustrating again the

RELATIONAL

STUDENT

BOB	22
FRED	21
MARY	27

CLASS

BOB	PHYSICS
BOB	MATH
BOB	ENGLISH
FRED	ENGLISH
FRED	PHILOSOPHY
MARY	PHYSICS
MARY	ENGLISH

COURSE

PHYSICS	ROOM A
MATH	ROOM B
ENGLISH	ROOM C
PHILOSOPHY	ROOM D

Figure 3: *Example of a relational data model.*

symmetry of the relational model. We can perform nearly every operation to a virtual file as we can to a real file.

Because data is stored in familiar, two-dimensional tables, users can create their own files and manipulate data through simple, interactive queries using an English-like, nonprocedural command language. Because of their simple design, relational databases are easy to modify and can grow as the application grows. The relational model is understandable even to the casual computer user.

Where does IMAGE fit as a Data Model?

One of the most common questions from people with an HP computer background is, "Where does IMAGE fit in?" IMAGE is the database system supplied by HP for the HP 1000 and HP 3000 computers. The answer is, IMAGE is based upon a restricted network model. The IMAGE data model most closely resembles the hierarchical model, but it is not strictly hierarchical, so we are forced to regard it as a restricted subset of the network model. The data model used by IMAGE requires the entity files be related in a two level hierarchy. The concept of MASTER-DETAIL files is useful for representing many-to-one relationships, but poor with many-to-many, or relationships involving more than two entities. IMAGE provides the advantages of a good database system: concurrent access to data, data independence, data security, and database utilities. Still, the data model used by IMAGE is very restrictive and can frustrate users trying to represent data that does not fit the model. The relational data model provides many improvements over the IMAGE data model in terms of flexibility and representing relationships.

Are Relational DBMSs Compatible With Other Data Models?

One of the greatest concerns of computer users is compatibility with existing software and data. Users are always concerned about introducing new software that may require changing existing files or programs and retraining personnel in their environment. Since the relational model handles only flat files similar to those used by the network and hierarchical data models, there is no reason that a relational DBMS could not be used with files created by another DBMS using a different data model. The capability to interface to all file types allows organizations to convert gradually to relational technology or to use some of the power of the relational access method with databases from the older data models.

Are Relational DBMSs Efficient?

A common misconception is that relational DBMSs are slow and inefficient. This undeserved reputation has issued from two sources. First, in the early days of relational technology, two major research organizations were attempting to show the feasibility of the relational data model. The two projects, INGRES, at University of California, Berkeley, and SYSTEM R, at IBM, were both research vehicles designed to show that a working database system could be built using the relational model. Both of these research attempts were successful. However, the implementations were not highly efficient because they were designed as research tools, not as production systems. Relational database systems are just as efficient as network systems and can be as efficient as the hierarchical model.

The second reason relational database systems have been perceived as slow is that the flexibility of the relational model allows the user to construct queries that can require extremely difficult database manipulations. These manipulations can be time consuming and, consequently, give the database the appearance of being slow. Yet these same tasks are ones that cannot be done at all with conventional query languages. It is far better that a vital task be done slowly than not done at all.

How Does a Relational DBMS Efficiently Access Data?

A detailed answer to this question is beyond the scope of this paper, but a brief answer may suffice. Many different techniques have been developed to access relational databases efficiently; these techniques ask the DBMS to analyze the query, examine the database structure, and to generate an efficient procedure to access the data. The major reason that relational database systems can be efficient is the intelligence built into the access mechanism.

One or more secondary indexes are usually associated with each file in a relational database system. An index is a mechanism that can be used to find quickly a particular record in a file, given some combination of fields from the file. Common indexing schemes are hashing functions, B-TREE indexes, or sorted keys. The intelligent relational access method uses these indexes to reduce the search for records in the database. The indexes play a role similar to the links in the network model, but the indexes are associated with a file and not a pair of files. The indexes may be created or deleted at any time. They may affect the processing speed, but do not affect the data or relationships among the data.

How Can Relational Technology Help Solve Data Management Problems?

This is the ultimate question because this is what users want most from a database system. The system designer may be impressed with the symmetry, generality, and mathematical beauty of the relational model, but the end user just wants to get his data in and out the way he or she wants it, as fast as possible, with minimal difficulty. The major advantages provided by the relational model are:

1. The power to represent all common relationships as they are seen by the user, without distorting the relationships to fit the data model.

2. The flexibility to change the structure of the database easily and quickly to reflect changing user needs. These changes often have little or no impact on applications that access the data.

3. The ability to access the data efficiently in many different ways without requiring that all possibilities be designed into the database in advance.

4. The ability to query the database in a non-procedural fashion, making it possible to develop very high level query languages that an inexperienced user can use as desired to view the data.

5. The flat nature of relational files, which allows a gradual conversion to relational technology, thus minimizing impact on existing systems.

Relational technology is the direction of the future in database systems. Information about new relational database systems is appearing more often in trade journals and vendor announcements. Relational DBMS systems can save hours of time when making simple changes to databases and can make many applications possible that were too complicated, too time consuming, or impractical to do with previous technology. Relational DBMS's are no longer the wave of the future—they are here now to solve today's data management needs.

Appendix

Articles on Relational Database Systems

Astrahan; Blasgen; Chamberlin; Fswaran; Gray; Griffiths; King; Lorie; McJones; Mehl; Putzolu; Traiger; Wade; Watson. "System R: Relational Approach to Database Management.". *ACM Transactions on Database Systems.* June 1976, pp. 97–137.

Brodie; Schmidt. "Final Report of the ANSI/X3/SPARC DBS-SG Relational Database Task Group." *ACM SIGMOD Record,* July 1982.

Codd. "A Relational Model of Data for Large Shared Data Banks." *CACM,* no. 13,(1970), pp. 377–387.

Codd. "Relational Database: A Practical Foundation for Productivity." *CACM,* vol. 25 no. 2, February 1982, pp. 109–117.

Codd. "Letters," *Computerworld,* June 7, 1982, p. 64.

Kreps; Stonebraker; Wong. "The Design and Implementation of INGRES." *ACM Transactions on Database Systems,* September 1976, pp. 189–222.

Wong; Youssefi. "Decomposition—A Strategy for Query Processing." *ACM Transactions on Database Systems,* no. 1,3, September 1976, pp. 223–241.

Books and Periodical Issues Devoted to Database Systems

ACM Computing Surveys, Vol. 8 no. 1, March 1976.

Date, C.J. *An Introduction to Database Systems.* Addison-Wesley, 1977.

Datamation, September 1981.

Bill Hasling is employed by Computer Resources Incorporated in Santa Clara as a senior programmer/analyst. He is working on development projects for RELATE/3000, the first relational database management system for the HP 3000. In addition to his expertise in database systems, he has held positions with RCA at the David Sarnoff Research Laboratories and at CALMA on advanced computer-aided design systems. Bill holds a B.S. in Mathematics/Computer Science from the University of California at Los Angeles and an M.S. from the University of California at Berkeley.

NEW PRODUCTS

Introducing the HP 82906A printer

The HP 82906A is a new low-cost general purpose printer for personal computers. This dot-matrix impact printer features a maximum print speed of 160 characters per second, and has a long list of features which make it an exceptionally versatile printer. The 82906A fits between the low-priced HP 82905B and the heavy-duty HP 2631B.

A few of the favorite features of the 82906A are:

- High speed (160 cps) for faster throughput
- Friction feed for single sheet paper
- HP standard raster graphics
- Six different character spacings
- Nine different line spacings
- Four levels of boldness
- Superscripts and subscripts
- Proportional spacing
- Automatic underline
- User-definable characters (or 2K print buffer).

The 82906A is available with HP-IB interface only.

The 82906A is recommended for applications which require fewer than 40 pages per day on average. For high volume applications, consider a heavy duty printer from Vancouver Division. If the extra speed and features of the 82906A are not required, the HP 82905B is a more economical alternative. And if typewriter-like quality is essential, choose the HP 2601A or 2602A printers. But if you are looking for a printer which combines high speed, multi-application versatility, and a reasonable price, look into the new HP 82906A.

The new electronic disc capability for the HP-85B

Electronic disc is just what its name implies — using electronic RAM to imitate a disc drive. When the HP-86 and HP-87 were introduced and they had the capability to address all of that RAM, many people asked why we could not do the same thing to the HP-85. We could have, but at the cost of compatibility with currently existing software. But we hit upon the idea of using the extra RAM as a mass storage device — specifically, a disc. In this way, the user can speed up applications considerably, particularly mass storage intensive applications, without serious modifications to the software. And here is why.

Electronic disc transfer rates:

HP-85	TAPE	EDISC
Allocated Program	660 b/s	46000 b/s
Unallocated Program	660 b/s	17500 b/s
Data (Numeric)	80 b/s	3400 b/s
Data (String, best case)	80 b/s	13000 b/s
Data (String, worst case)	80 b/s	600 b/s

This is a considerable improvement in speed. For example, if the Graphics Presentation Pac were run and it had to CHAIN (as it frequently does), it would chain about 20KB. Off of tape this would take about 20,000/600 = 30 seconds. Off of Electronic disc it would take 20,000/17,500 = 1 second. Quite a difference.

It also makes data logging faster. With a memory of only 32K, the 85 had to write to tape frequently. But with the Electronic Disc, writing to mass storage is so fast that it becomes almost negligible. This can make a big difference in a speed critical application.

How to get the most from EDisc

A careful look at the Transfer Rate chart above will reveal that the EDisc can be very fast, but can also be quite slow in the case of string transfers. This article won't go into the reasons for the vast difference in speed, but will tell how to get the most from Electronic Disc.

The only place there is a real change in speed is in string data transfers. The speed of this type transfer can range from .6Kb/s to 13Kb/s. This is quite a difference. Here are some examples of how certain speeds can be obtained.

If a program is written like this:

```
10 DIM A$[10000]
20 ASSIGN#1 TO "DATA"
30 A$ = <something>
40 PRINT#1;A$
50 ASSIGN#1 TO *
60 END
```

the transfer rate to the EDisc will be about 600b/s. Not very fast. If the program was written like this:

```
10 DIM A$[10000]
20 ASSIGN#1 TO "DATA"
30 A$ = <something>
40 FOR I = 1 TO 10000/252
50 PRINT#1,I;A$[I*252-251,I*252]
60 NEXT I
70 PRINT#1,A$[I*252-251,10000]
80 ASSIGN#1 TO *
90 END
```

the transfer rate would be about 7Kb/s. A significant improvement! But to really get the most out of a string transfer, here is how it is done:

```
10 DIM A$[250],B$[250],C$[250]. . . X$[250]
20 ASSIGN#1 TO "DATA"
30 A$,B$,. . . ,X$=<something>
40 PRINT#1;A$,B$,. . . . ,X$
50 ASSIGN#1 TO *
60 END
```

The transfer rate here is 13Kb/s. Very fast indeed for data of any kind.

That's all there is to it. Electronic disc can be very beneficial to your application, particularly if used to its fullest capabilities.

Introducing Control/1000

Control/1000 refers to both a hardware configuration and a software package designed to bring the power of the HP 1000 A-Series processors to factory and plant automation.

To achieve this, we took the HP 2250 measurement and control capabilities and combined them with the A600 microcomputer, RTE-A, and DS/1000-IV to give our customers true computer processing and networking capability in their process or machine control application. The previous L-Series CPU with its MCL/50 compiler is replaced with the A-Series machine and RTE-A. The HP-IB 2250 link to a host computer is functionally replaced with DS/1000-IV hardware and software. Initially, we will only offer the DS node version of Control/1000 as a supported system (a host is necessary). Announcements of standalone configurations are planned for the future.

High level languages for process control such as FORTRAN 77 or Pascal. Now you can write process control programs in the high level languages of FORTRAN 77 and Pascal and run them on the A600. With Control/1000, we've taken some of the best features from MCL/50, the HP 2250 measurement and control language. We've converted the MCL/50 I/O statements to subroutines capable of being called from high level languages running on the A600. Now, program control, computation, and data manipulation can be performed using the high level language, but measurement and control card I/O calls still retain the user friendliness of MCL/50. For customers needing maximum speed, we've created a mode which allows the user to talk directly to the M&C cards without the overhead of these calls.

All the benefits of the Real-time Executive operating system such as true multi-tasking and high level languages. Up until now a user of the HP 2250 that used an HP 1000 as a host had to deal with MCL/50 and RTE. Now we've eliminated MCL/50 and replaced it with easier-to-use subroutines which users can call from application programs. You need never know that you're using anything other than a high level language. Multiple tasks can be run as background programs in the Control/1000 A600 processor, while at the same time the processor is running the primary automation application program.

A large memory space of 512 kilobytes. This provides a substantial increase in ease of writing application programs once the unit is up and running. You can now write your application programs without worrying about constant segment passing to the HP 2250 because of lack of memory. For those number crunching intensive processes the user no longer has to pass the data to the host — the A600 can do it in real-time, in the HP 2250!

True networking performance. Now process control or machine control computers can benefit from all the features of the DS/1000-IV network. This can be used to pass information to the right people at the right time, including everything from maintenance information to process efficiency data. Imagine what this can do for your productivity.

CP/M-68K is coming

HP Series 200 systems will soon have CP/M. Digital Research (DRI) has re-written CP/M for the MC 68000 microprocessor in the C language to take advantage of the 68000's speed. Included with the CP/M-68K system is a C compiler and a run-time library, a bridge to UNIX.

One caveat: because CP/M-68K is a new version of the operating system, the software currently running under CP/M-80 and -86 will not run on this new system. Applications that are written primarily in a high-level language, such as C or CBASIC, should port to CP/M-68K fairly easily. We are currently working with several CP/M software vendors who are porting their software to CP/M-68K.

As the new system becomes more widely accepted, additional software vendors will probably convert their software to run on CP/M-68K.

Because most Series 200 customers will use CP/M-68K in addition to another operating system, a second note of caution is appropriate. CP/M-68K file formats are not compatible with the LIF format used on the Series 200. (They are compatible with the Series 80 and 100 CP/M data files, however).

If you are using flexible discs, this is not a problem, because you simply reboot off the appropriate disc. If, however, you want to use a hard disc, there is a problem. In general, a disc can be either LIF or CP/M, but not both. A way around this problem is to use a hard disc that is partitioned. Then one or more of the partitions can be CP/M and the others can be LIF.

Partitioned discs are only available in the 5MB version, by ordering no option. They will not be available in 10Mb discs.

A Novel Approach to Budgeting

A new management computer programme is now available to help managers of manufacturing industries to plan profitable operations.

It involves a novel approach of synthesis, instead of the usual analysis, of the operations.

To start with a pre-determined profit on sales is stipulated. The so called "Truth-Table" quickly establishes feasibility and necessary overall strategy required to achieve this goal.

The next stage involves creation of a preferred scenario to realise the strategy of the "Truth-Table". The details of product-mix as well as other key parameters are established for a period of up to 12 months, and then presented on a monthly basis.

Detailed budgeting for a period of up to 12 months follows. Actual performance results are posted and compared with budgeted key parameters.

Any unwelcome trends can be spotted at an early date and alternative corrective action evaluated.

The programme is designed for managers and not for computer operators. It serves as a powerful tool for planning, execution and control of all vital parameters. It is also suitable for critical evaluation of potential acquisitions. The minimum configuration requires Hewlett-Packard HP86A desk-top personal computer with associated Video Monitor, 5 ¼" Floppy Disc Mass-Storage Unit and an additional 32k Memory Module. An additional printer is recommended.

The programme is now available from HP Dealers or contact H. Drillich, 44 Dempster Ave., North Balwyn, 3104. Phone (03) 857 6816.

DESKTOP FORUM

Strings and Things

by Chris Simpson, Simpson Computer Services Pty. Ltd.
63 Hartington Street, Kew, Vic. 3101. Phone (03) 859 6643.

A COMPARISON OF STRING HANDLING FACILITIES ON HP DESKTOP COMPUTERS, PLUS A FEW SNIPPETS (BORROWED FROM A RECENT TALK TO THE MELBOURNE H.P.D.C.U.G.V.).

Vivid memories linger of when, wonder of wonders, the 9830 was blessed with a STRINGS ROM! This was in 1974. A new era opened up in which numerical results could be clarified with text. Later, we were able to store text into arrays (with difficulty) when the Advanced Programming ROM finally emerged.

Since then, string handling has become an everyday necessity; taken for granted. HP did not see the light particularly early and left it until 1982 to virtually standardize their string-handling language in "HP BASIC". So we are blessed with several dialects within HP and many more from other manufacturers. Perhaps in the 'others league', MBASIC (Microsoft) is now the most common. HP seemed to think so when the HP125 was introduced.

1. Let's look at some of these Differences in String-handling Facilities. Table 1 attempts to summarise these for several desktop computers. Clearly the 'rogue' in the HP line-up is the Series 80. And as for the HP125, well . . . .?
2. Some Series 80 Differences were described in Crosstalk, August '82 edition. Ron Davis gave us a clear warning about precisions and VAL$. The null string 'problem' was also talked about. As shown in Table 1, filling a string with blanks differs from other desktops in how it may be achieved. The difference lies with Series 80 assuming "the whole string" for both the A$ and A$ [1, last-dim'd-char.no.] representations, whereas for the 9845, the latter representation is classified as a SUBSTRING even though in fact it IS the whole string.

   For example, if A$ is dimensioned to 40 chars, then:-

   A$[1,40] = " " gives rise to 40 spaces (9845) or null string (Series 80)
   or
   A$[1,39] = " " gives rise to 39 spaces (9845 and Series 80),
   or
   A$[2,40] = " " gives rise to an error (9845) or 40 spaces (Series 80).

   Another major difference between Series 80 and 9845 ("HP Basic") is with **STRING COMPARISONS.** Both systems use the ASCII code order to determine smallness or greatness of characters. Both systems compare character by character to give a dictionary style order, EXCEPT that with Series 80, string length over-rides lexical order.

   For example,
   "B" > "AB" is true for 9845 (lexical) but FALSE for Series 80 (length) but "AC" > "AB" is true for both, as the strings are the same length.

   A couple of rules of thumb could be used to surmount these 'problems' (as seen by "HP BASIC" users when switching to a Series 80 machine):

   (a) Always dimension a string to (at least) ONE MORE character in length than will ever be used.

   (b) When comparing strings, always make them EQUAL IN LENGTH by padding the shorter one with trailing spaces first.
3. **A SIMPLE BUBBLE SORT** is often desired for doing a quick job rather than hunting around for more complex, quicker sort programs. Not only can the bubble sort be used to order numbers in an array, but also strings in a string array, or even string records in a 'long string'. One can almost remember this routine by heart:

```
L=N

---FOR P=1 TO N-1              (N is the number of string records
                               (L is the no. of comparisons for the pass
   L=L-1                       (P is the pass number
                               (C is the comparison number
 --FOR C=1 to L

   Put record C into A$

   Put record C+1 into B$

   If A$<=B$ then ->->--       (asceding order) (Use A$>=B$ for descending)
   Put A$ into record C+1  )
   Put B$ into record C    )  swap records around
 --NEXT C  <------------
              skip
---NEXT P
```

4. String arrays are often used for **INDEXES,** e.g.: To find a certain title in the example, one must search from the top down until a match is found, which is time-consuming in BASIC.

Title 1	Pointer 1
Title 2	Pointer 2
⋮	
Title N	Pointer N

   An alternative is to use a LONG STRING as the index, with each title-plus-pointer as a known-length record within the long string. Now to find a certain title, one may use the POS FUNCTION, which is **ultra quick!**

   A second advantage of the long string index is to be gained when inserting a record into the string between two adjacent records. In the array, one would need to 'ripple-down' all the records below the insertion point in order to make room for the new one. The long string, however, requires only to be broken and the new bit tacked in — much quicker!

5. Strings may be used for **COMPACT NUMBER STORAGE,** especially useful if memory is at a premium.

(a) Whole numbers in the range 0 to 255 may be stored in one byte, say the Xth byte of A$, and retrieved, thus:

Storing: A$[X,X] = CHR$(N) (N is the number)
Retrieving: N=NUM (A$[X,X] or N=NUM (A$[X]) will do.

(b) Two bytes offer storage of whole numbers from 0 to 65535:-
Storing: A$[X,XH] = CHR$(N/256 or NDIV256) & CHR$(NMOD256)
Retrieving: N=256*NUM(A$[X]) + NUM (A$[XH])

(c) Integers from 0 to 16,777,215 may be stored in 3 bytes, etc. **Note:** that an IMAGINARY decimal point could be catered for. The three-byte number could represent up to $167,772.15, provided that it is multiplied by 100 before storing (as cents), and divided by 100 upon retrieval to re-create dollars.

The above number systems simply use the mathematical base 256 rather than the more familiar base 10 (decimal). It's efficiency lies in the use of every single bit of memory!

In conclusion, may I invite readers to submit **their** experiences and snippets about strings and things, as we have hardly scratched the surface here!

CHRIS SIMPSON

TABLE 1 : A COMPARISON OF STRING HANDLING FACILITIES
BETWEEN VARIOUS DESKTOP COMPUTERS

COMPUTER/STRING FACILITY	9825 (ETC) HPL	9831 (9830)	9835/45 9826/36 "HP BASIC"	HP86,87 (HP85) DIFFERENT!	HP125 AND MICROSOFT BASIC (TYPICAL)
STRING NAME	One letter e.g. A$	One letter e.g. A$	To 18 char's e.g. Fred$	Multiple char- acters (2 chars maximum)	Two characters
MAX. LENGTH (Bytes)	32,767	32,767 (256)	32,767	65,536 (to full memory)	256
ARRAYS OK? SPECIFY THIS:	Yes A$[X,..]	No,but indirectly with TRANSFER	Yes A$(X)[...]	Yes (no!)	Yes A$(X) No substring
STRING ASSIGNMENT	"AB"→A$	A$="ABC"	A$="ABC"	A$="ABC"	A$="ABC"
SUBSTRING ASSIGNMENT	"AB"→A$[X,Y]	A$[X,Y]="ABC"	A$[X,Y]="ABC"	A$[X,Y]="ABC"	No!!
SUBSTRING EXTRACTION	A$[X,Y]→B$	B$=A$[X,Y]	B$=A$[X,Y]	B$=A$[X,Y]	B$=MID$ (A$,start char, no. of chars.)
CLEARING TO NULL-STRING	""→A$	A$=""	A$=""	A$=""	A$=""
AUTOMATIC SPACE-FILLING (If dimensioned to X chars.)	""→A$[1,X]	A$[1,X]=""	A$[1,X]=""	Not like 9845etc. A$(1X]="" gives null string A$[1,X-1]=""gives X-1 spaces	A$=SPACE$(X)
DIMENSIONING : Simple:	dimA$[123]	DIMA$[123]	DIMA$[123]	DIMA$[123]	Auto dim'g to 256 chars.
Array:	dimB$[3,256]	No!	DIMB$(3)[256]	DIMB$(3)[256] Except HP85:No!	DIMB$(3) length 256 ea.
Indirect:	dimA$[L]	No!	No!	No!	No!
LEN FUNCTION	Yes	Yes	Yes	Yes	Yes
POS FUNCTION	Yes	Yes	Yes	Yes	No! 'STRING' function for some.
CONCATENATION (Tacking on)	A$&X$&"ABC"	Indirectly : A$[len(A$)+1)=X$	---&---	---$---	---+---
STRING DIGITS TO A NO.	val(A$)[..]→N	N=VAL(A$[..])	N=VAL(A$[..])	N=VAL(A$[..])	N=VAL(A$[..])
NUMBER INTO STRING	str(N)→A$	(AP2 ROM)STRING'.	A$=VAL$(N)	A$=VAL$(N)	A$=STR$(N)
STRING CHAR. TO ASCII CODE	num(A$[..])→C	(AP2 ROM) via TRANSFER	C=NUM(A$[..])	C=NUM(A$[..])	C=ASC(A$) takes first char.of A$
ASCII CODE TO STRING CHAR.	char(c)→A$[..]	(AP2 ROM) via TRANSFER	A$[..]=CHR$(C)	A$[..]=CHR$(C)	A$=CHR$(C)
TO UPPER CASE	cap (A$)	No!	UPC$ (A$)	UPC$ (A$)	No?
TO LOWER CASE	No!	No!	LWC$ (A$)	No!	No?
REV$/TRIM$/RPTS$	No!	No!	Yes	Only with binaries or ROMS	No?

Selective Keyboard Lockout on the Series 200

The Series 200 provides a way to selectively lockout individual keys using a combination of the ON KBD and OUTPUT to the keyboard features in the BASIC language.

First, the ON KBD ALL command traps all keystrokes. That is, any time the user presses a key, an interrupt will be generated to the running program. The interrupt service routine (ISR) can then decide what to do with the key that was pressed. In many applications, the programmer may choose to ignore all keystrokes so that the keyboard is effectively "locked out".

However, in some applications, it may be desirable to only lock out certain keys, while letting the rest be serviced normally by the operating system. In this case, the ISR can look to see which key(s) were pressed, and if they aren't the "forbidden" keys, pass them on to the operating system using an OUTPUT to select code 2, which is the keyboard.

Suppose for example that the programmer wants to guarantee that the user doesn't PAUSE or STOP the program. Using the technique outlined above, he may selectively lockout the PAUSE and STOP keys like so:

```
10    ON KBD ALL GOSUB Kserv
20    GOTO 20
30 Kserv:      !
40    Key$=KBD$
50    IF Key$=CHR$(255)&"P" OR Key$=CHR$(255)&"!" THEN RETURN
 )    OUTPUT 2 USING "#,K";Key$
 0    RETURN
80    END
```

The above program will run forever (unless you do a RESET) since the PAUSE and STOP keys are disabled, but all the other keys are active — the soft keys, EXECUTE (2 + 2, SIN(45), and other live keyboard operations), clear screen, print all and everything work as usual. Naturally, any other keys on the keyboard can be locked out as well. All you have to do is change the keycodes the program looks for.

H.P.D.C.U.G.V.

PRESIDENT'S REPORT FOR YEAR ENDING 28/2/83

At the end of our first year I am happy to report that our group is in a strong position, but before going into details it seems appropriate to review our beginnings.

On 15th February, 1982, a meeting was convened at Blackburn by Robert Dey of H.P. to consider the possibility of forming a Desktop Computer User's group. The interest seemed to be there so an inaugural meeting was scheduled for 19th April, and invitations issued. The response was overwhelming, necessitating some hasty re-arrangements of accommodation on the day. A committee was elected and set the task of preparing a constitution.

At the next meeting on 25th May, 1982, the constitution was adopted and the group effectively established.

Since then we have had five group meetings and six committee meetings. The committee organised a program of interesting items for the group meetings and we experimented with the meeting format. Average meeting attendance was 40 — 50, which is excellent considering that some of our membership (currently 67) is not even located in Melbourne.

Under Ron Davis' care our financial dealings were put into first class order and since our expenditures have been minimal our financial position is very healthy.

Ian McWilliam has our membership listing and mailing list very well organised on his 9825 so that he can get a mailing out in very short order. This system was the subject of a short talk at a recent meeting.

Under Tony Steven's care our library of some 200 items, mainly manuals and other H.P. literature was catalogued on a card system which will shortly be transferred to a computer data base so that listings can be run easily. We hope that this will help members make better use of our library in the future. Tony also has organised and maintains the forms we all filled out on joining the group which describes members hardware and software.

One of our meetings was held in the excellent conference facilities at C.S.I.R.O. in Syndal. We hope to utilize this venue regularly in the forthcoming year and are very appreciative of C.S.I.R.O.'s generosity.

Since we began there have been five issues of the magazine "CROSSTALK" in all of which we were well represented. Recently we took over the responsibility for mailing issues to our members from H.P. and are now considering making substantial contributions to its operating costs.

One of the main objectives of our group is to encourage and foster discussion and interchange of information amongst members. Your committee has worked hard on this task and a lot of progress has been made but members still sometimes find it difficult to locate fellow members who have experienced similar problems to their own. An improvement is needed in this area.

In summary I am happy to report that our membership is healthy and growing and we are in a sound financial position. Our library and journal arrangements are in good shape and a program of regular meetings has been established.

I would particularly like to thank the members of the committee for their efforts throughout the year.

Finally I extend to all members the challenge to come forward with short talks and with articles for CROSSTALK. The more you give, the more you will gain.

B. T. O'Shannassy,
President
13/4/83

H.P.D.C.U.G.V. COMMITTEE MEMBERS

— May 1983

PRESIDENT:
B. T. O'Shannassy
Arlec Pty. Ltd.
30 Lexton Road,
BOX HILL, 3128.
Bus. 840 1222. A.H. 878 0730

H.P. REPRESENTATIVE:
Mr P. Greetham,
Hewlett-Packard Australia Ltd.,
31-41 Joseph Street,
BLACKBURN, 3130.
Bus. 877 7777

MEMBERSHIP:
Mr I. McWilliam,
Department of Chemistry,
Swinburne Institute of Technology,
Post Office Box 218,
HAWTHORN, 3122.
Bus. 819 8911. A.H. 818 2786

TREASURER:
Mr P. Hendy,
I.G. Hendy Manufacturing,
36 Alfred Street,
BLACKBURN.
Bus. 878 3333

Mr C. Simpson,
Simpson Computer Services Pty. Ltd.,
63 Hartington Street,
KEW, 3101.
Bus. 859 6643

Mr M. Daly,
73 Husband Road,
FOREST HILLS.
Bus. 609 1591. A.H. 878 7772

LIBRARIAN:
Mr A. Stevens,
Telecom Research,
770 Blackburn Road,
CLAYTON.
Bus. 541 6532. A.H. 232 4022

SYDNEY DESKTOP COMPUTER USER GROUP

A meeting of HP Desktop Computer users, chaired by Rex Jooste of HP, was held on Thursday, 23rd June, 1983, at the office of HP in North Ryde, N.S.W. The following office bearers were elected:

Sharon Beder, Noyes Bros., Chairperson.
Bob Harris, C.S.I.R.O., Secretary.
Steve Mitchell, Abbott A/Asia, Treasurer.
Dick Lovegrove, Kingdom, Librarian.

John Knaggs gave an interesting demo of the Context MBA software package on a 9816.

The next meeting will be held at 3.00 pm. on Wednesday, 20th July at the same venue.

All enquiries should be directed to Bob Harris, (02) 543 3460.

Focus 1000

UPGRADES TO RTE-A AND VC+: AN OPPORTUNITY TO GROW WITH HP

All A-Series owners will be able to upgrade their computers to take advantage of the additional power in the RTE-A operating system and the powerful program development features offered by VC+.

Customers who have subscribed to software services to stay abreast of software developments (either CSS or SSS) will receive an automatic, free-of-charge upgrade from RTE-A.1 to RTE-A in the B.83 revision. This upgrade will include firmware and installation, in addition to a copy of the latest software, (this does not include the VC+ package which may be purchased as a separate product).

Customers who do not subscribe to software services will still be able to upgrade their equipment to run RTE-A or VC+. In this note we will discuss the advantages this upgrade program offers to present A-Series customers and how their current applications will be affected.

Reasons to Upgrade

Users who are currently developing application code are likely to benefit the most from an upgrade to RTE-A and VC+. RTE-A provides a friendlier, more flexible and better organised user environment for program development. VC+, an optional O/S enhancement package, adds multiuser development tools. Specifically, users who move to RTE-A will benefit from:

- Increased programmer productivity using the new file structure (with time stamps, longer file names, detailed error messages and a hierarchical file directory).
- Support of a larger number of peripherals (up to 255 LUs). This should particularly benefit instrumentation-intensive applications.

VC+ is recommended for the development of large applications where many users share a development system. VC+ offers:

- Full outspooling and protection among users for a more secure working environment.
- Transparent development of megabyte-sized programs.
- Shared code among users running the same program to save physical memory.

Current applications will be minimally impacted by the changeover to RTE-A.

- All A-Series programs using standard systems services will execute on this new revision of RTE-A.
- Systems configured with only 128Kb memory may, in some cases, require additional memory, due to the increased memory requirements of RTE-A and associated system programs.
- Only the 2326 version of subsystems should be used with RTE-A.

PROGRAM HANDLING.

***Large Programs**

By separating the code from the data, a virtual scheme for programs can be supported allowing support of large programs. Totally transparent to the programmer, a Fortran 77 (*1) program in the neighborhood of 200,000 to 500,000 lines can be compiled, loaded and run without any modifications. BYE-BYE SEGLD!!

***Shared Code**

The separation of code and data also allows the use of **re-entrant and recursive programming** techniques. The operating system will also use this re-entrancy to support the sharing of a code partition by many data partitions. This allows multiple copies of a program to be executed using only one copy of the code, consequently saving significant physical memory. Shared data is still available through shareable EMA areas.

MULTIUSER ENVIRONMENT.

***Log-On and Log-Off**

Log-on and Log-off utilities allow for **two levels of users** in a system with the ability for programs to be executed at log-on to create a particular environment for a user. These utilities allow a user to read and/or write protect individual files or directories of files via the log-on and password, providing much desired protection of files between users for a program development or high-security environment.

***Outspooling**

The outspooling utility permits programmatic and interactive outspooling of files to devices or files. This utility also provides the capability **for I/O redirection,** allowing a user to redirect output from one device to another or to a file. Error logging is also provided allowing system error messages to be echoed to a file as well as to the system console. This provides for analysis of system activity without constant monitoring.

(*1) The FTN77 compiler has been updated for this feature at revision B.83.

***Larger Systems**
RTE-A will allow drivers to be mapped in and out of the system creating more room for system tables. This will allow more programs and devices to be managed by one RTE-A system. The file system will allow larger files and support large discs allowing **over 20 Gigabytes of mass storage.**

***Transportable Program Files**
Programs loaded for one RTE-A system, with a few exceptions (*1), will execute on another RTE-A system. This greatly facilitates the process of updating a system configuration or revision.

***More Devices**
RTE-A provides the capability to access **up to 255 Logical Units** in the system. Coupled with the large system features discussed above, this allows the customer to generate **FOUR times as many I/0 devices** into the system than earlier RTE-A revisions.

(*1) Programs which access system common will normally have to be reloaded.

***DS Transparency**
RTE-A adds the capability to access files located at other RTE-A nodes transparent to the DS subsystem. Merely by specifying the node number in the file name, a local file call becomes a remote file call. This will greatly facilitate the communication between RTE-A nodes both interactively and programmatically.

***Friendly User Interface**
A new user interface, Command Interpreter (CI), has been aimed at creating a friendlier environment for the A-Series user. **On-line help facilities** will make it easy for a user to find out how to use a particular command and what commands are available. **Expanded error messages** make CI easier to understand what error has occurred when using a particular command. **A command stack** presents a more forgiving environment when a user makes a typing error or simply wishes to repeat a command.

OTHER OPERATING SYSTEM ENHANCEMENTS.
***More SAM**
Up to 64 kilobytes of System Available Memory will always be available to the system designer for RTE-A. This is guaranteed by dedicating a set of map registers to SAM. SAM is used by the system for buffering I/0, so increasing the amount available will increase overall system throughput and performance.

***Longer file names**
By allowing 16-character file names, the user can more adequately describe the contents of the file via the name, as well as remember the contents at a later date. The 4-character type extension will be available to differentiate between a source, relocatable, program and any other type which a user defines. For example this will replace the previous cryptic convention of '&', '%', etc. with 'FTN' and 'REL', which will present a more friendly, understandable convention for naming files.

***Sophisticated File Masking**
Many file operations may be specified using the file mask feature, giving access to multiple files with a single command. In addition to two "wildcard" characters, there is a mask qualifier of up to 40 characters which may be appended to a file name and type extension.

***Unpurge**
A purged file can now be reinstated by the UNPURGE command, provided that the disc space it used has not been overwritten by subsequent file operations.

***Incremental Back-Up of Files**
The addition of a back-up bit to the information retained on a file, will allow a user to just back-up those files which have been changed. This will cut down the time spent on the back-up process significantly.

FILE SYSTEM ENHANCEMENTS.
***Hierarchical Directory Format**
Users can have more efficient and more logical disc space utilization because of the hierarchical directories. To logically group a common set of files, a programmer need merely create a new subdirectory for these files. More and more subdirectories can be created on-line expanding the hierarchy vertically and horizontally.

***File and directory ownership and protection**
This feature is provided in conjunction with the log-on facility which is part of the VC+ product. Under VC+, each file and directory has a unique owner, which by default, is the creator, however this ownership may be transferred to other users on the system. The owner of a directory has special privileges regarding the files contained in that directory, such as the capability to specify file protection status. File access will be determined by checking protect bits. Protection will be read, write, both or neither for owner, everyone or neither. The protection bits can be checked or changed either interactively or programmatically.

***Time stamps**
The system will maintain the time of creation, last update and last access for each file. These times can be checked either interactively or programmatically. This will present a tool to the programmer for maintaining revisions of software as well as keeping much-needed disc space free of 'old' programs.

Focus 1000

PROGRAMMING TIP FOR HP 1000

The following utility program is useful for printing files on a line printer without using LI, which prints line numbers or DU, which uses the first character of each line as a control character when dumping to the line printer.
To execute, simply: PT, filename, 6

```
PROGRAM PT(INPUT,OUTPUT);
BEGIN
   WHILE NOT EOF DO BEGIN
      IF EOLN THEN BEGIN
         READLN;
         WRITELN;
      END
      ELSE BEGIN
         OUTPUT^ := INPUT^;
         PUT (OUTPUT);
         GET (INPUT);
      END;
   END;
   PAGE (OUTPUT);
END.
```

COMING EVENTS

18th July: RTE-A/VC+ System Designer's Course, HP Sydney

20th July: Sydney Desktop Computer User's Group Meeting. Bob Harris (02) 543 3468

25th July: RTE-6/VM Upgrade Course, HP Melbourne

25th July: 9845 Operating and Programming Course, HP Sydney

25th July: HP 1000 Distributed Systems/ Networks Course, HP Sydney

27th July: RTE-A.1 for RTE User's, HP Melbourne

8th August: Image — Advanced Workshop, HP Sydney

15th August: HP Basic Programming Course, HP Melbourne

22nd Aug.: Intro to HP 1000 Course, HP Sydney

22nd Aug.: Pascal Course, HP Melbourne

29th August: RTE-A/VC+ System User's Course, HP Melbourne

29th August: RTE Internals and Operating Course, HP Sydney

6th Sept.: H.P. Desktop Computer User's Group Meeting, C.S.I.R.O. Syndal, Vic.

Surplus Manuals

The HP1000 User Group has the following surplus manuals to give away to any bonafide user. Contact the Secretary, Chris Emery on (03) 667 72328.

HP92068A RTE-IVB Programmer's Reference
HP59310 HP-IB for HP1000 User Manual
HP021000 Decimal String Arithmetic Routines
HP24998 DOS/RTE Relocatable Library Ref.
HP02142 Greeting Started Model 5
HP92840 Graphics Plotting Software: Users
HP92060 BASIC/1000D Ref. Manual
HP92067 RTE-IV Assembler Ref.
HP92060 RTE FORTRAN IV Ref Manual
HP92068 RTE-IVB Manager's Manual
HP92068 RTE-IVB On-Line Generator Ref.
HP92068 RTE-IVB Batch & Spooling Ref.
HP92068 RTE-IVB Terminal Users Ref.