DATABASE MANAGEMENT SYSTEMS ON A MULTI-COMPUTER COMPUTER

M. Sinutko
Department of Defense
Washington, D.C.

ABSTRACT

This paper discusses fitting a database and database management system onto a particular research status computer constructed of a large array or network of closely-connected component computers. The findings may have application to other distributed computer architectures.

Preface

One effort in a particular research project to produce future computer architectures involves applying existing problems and/or current problem solution procedures to the candidate structures. Depending upon the results of such tests, the research thereby receives a measure of encouragement or discouragement. If discouragement, an analysis is then made to determine if the architecture is faulty.

The discussion that follows addresses some selected database management systems as applied to one new multicomputing architecture. Although originally intended for use only within the aforementioned project, the findings may have application to other multiprocessing structures.

I wish to thank Drs. C. Cook and B. Shneiderman of the Department of Defense and the University of Maryland, respectively, for their review, consultation, and suggestions.

SECTION 1

Introduction

This paper examines ways in which a unified* database (DB) could be accommodated (contained in) and how well each of three recognized models for data base management systems (DBMSs) might perform on a postulated but unrealized, research status computer having atypical architecture. The three DBMSs to be used as bases for performance judgment are E. F. Codd's relational DBMS [2], the network-approach CODASYL Data Base Task Group (DBTG) version [10], and the hierarchical Information Management System (IMS) [11]. Each DB/DBMS will be evaluated for amenability to accommodation* in our computer, and the facility with which it allows access to the contained information.

*See Appendix 1 for definition.

This study is not intended to be exhaustive. Instead, it is of a first-pass exploratory nature, intended to uncover basic peculiarities of affinity or aversion among a representative population of DBMS models as applied to a particular computer. Also, it assumes of the reader, a general familiarity with the selected DBMSs as background. For those unacquainted with these models, the references in the preceding paragraph comprise sufficient prerequisite reading.

Before discussing the DB/DBMS tests, a brief description of the subject computer will be given. Although its architecture may appear straightforward, it has no well-defined operating schemata. These schemata are elusive and hence are principal objectives of the associated on-going research.

1.1 A Multi-Computer Computer

Our research is in pursuit of methods by which "very high" performance computers can be constructed from assemblages of "low" performance machines. Development of such methods have the potential of yielding computers which are eminently enhancible, downgradable, customizable, reliable, maintainable, and resistant to technological obsolescence. Given success, operating costs incurred for large scale computer operations as compared to today's could be significantly lessened.

One architecture currently under investigation was selected as our host machine, which we have named, "Multi-Computer Computer" (MCC). It is constructed of a heterogeneous mix of a large number of uni-programming (as opposed to multiprogramming) computers and/or processors arranged in a closely connected array to form a single supercomputer*. "Large", as used above, means having such cardinality (e.g., nundreds, thousands, ...) that it exceeds all known current practical means for their useful electrical interconnection. "Closely connected", also used above, is specified to imply the exclusion of large geographically distributed computer networks.

At each node of the array (or, "array node"*) there exists an autonomous "array computing element"* (ACE) composed of "elemental processors"* (ePs) and/or "elemental computers"* (eCs). This hardware array, when properly interconnected, forms a supercomputer, our MCC. To assist the reader in understanding the scale of the MCC, a realization of it as of the date of this paper would probably consist of an array of microprocessors (μPs) and/or microcomputers (μCs). The MCC's physical size is governed primarily by the volume occupied by its ePs and/or eCs (abbreviated "eP/Cs"), the supportive circuitry, and other associated hardware.

Consider the actual array structure, of which the ACEs are nodes, to be determined and possibly unique for a given set of ACEs. Direct input/output (I/O) between each ACE and the "user space"* is allowed, but not necessary. At least one channel for both input and output must be provided between the total MCC and the user space. I/O capabilities

*See Appendix 1 for definition.

among the ACEs are required. Tasks can be initiated in any non-busy ACE via its "user I/O port" (UIOP) or its "array I/O port" (AIOP). Multiprogramming at any array node is accomplished through duplication of the eP/Cs (i.e., multiprogramming is accomplished through multiprocessing). Figure 1 shows a generalized MCC.

An instance of the MCC could perform in a manner analogous to the utility network described by Marill and Stern [6], but on a closely-connected rather than geographic scale (as theirs is). The MCC does not restrict its ACEs to function as narrowly as their "Datacomputer", but they can be so assigned. In fact, such a dedication of not only one, but a set of ACEs, is a solution to a database accommodation problem discussed in section 2. That set of ACEs (with some other hardware) functions similarly, scale considered, to Marill and Stern's Datacomputer.

1.1.1 Operation of the MCC

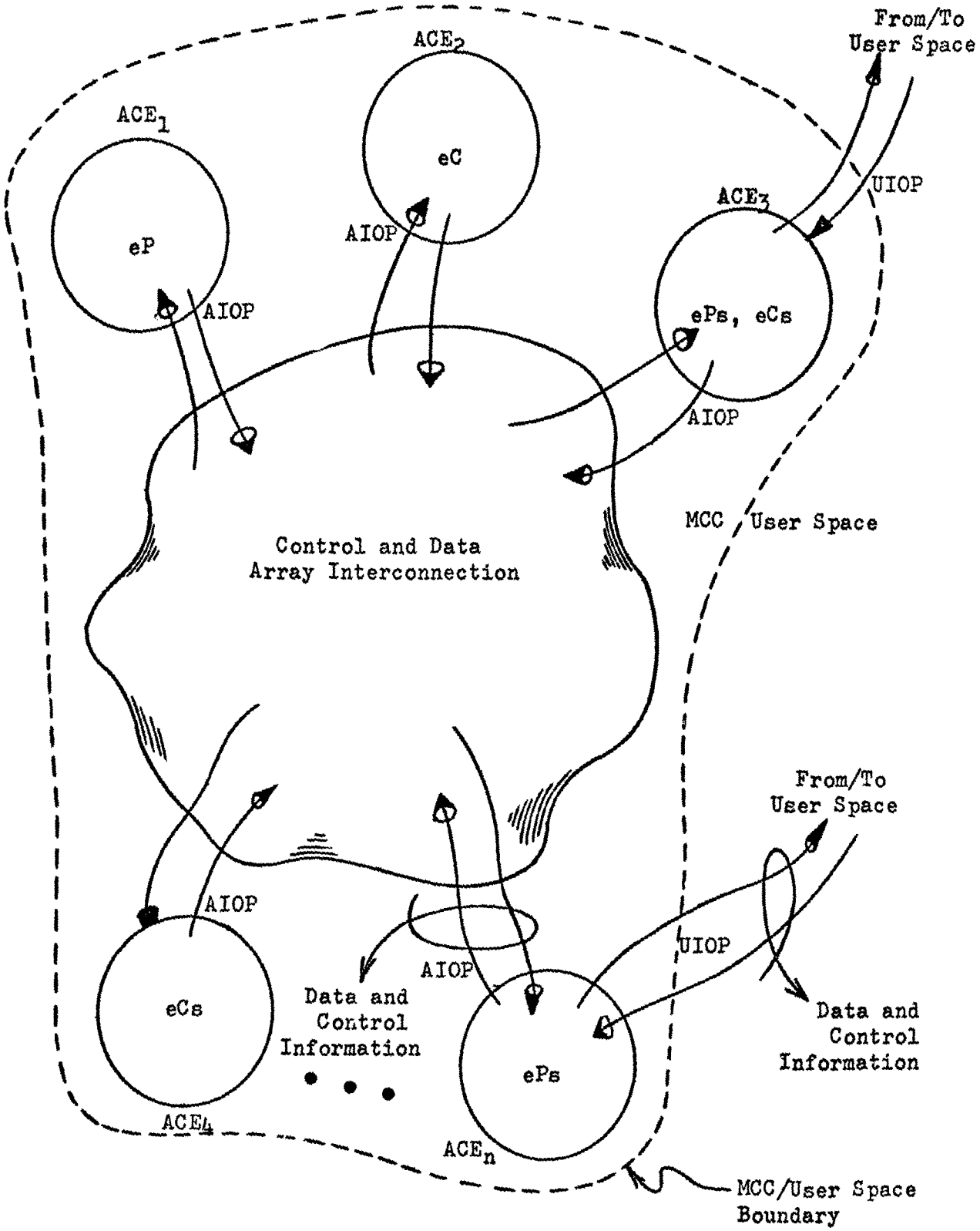
Each ACE is described by a "characteristic equation" (CEQ)*, and the set of all CEQs for the MCC thereby forms a logical description of the MCC. Graphically, this logical description and the physical implementation of the MCC do not have to be the same. The MCC can operate in at least one of the following two modes:

Resource Request Mode - An ACE broadcasts, to the rest of the array, synthesized CEQs representing resources it is seeking in order to accomplish a task it has received via its user I/O port. The broadcast may have been initiated due to insufficient resources at the transmitting ACE or because it is looking for a better way to perform its task. If greater-than or equal capabilities matches are found in the array, the possessing ACEs volunteer to participate in the broadcasting ACE's (subsequently partitioned) task. The handling of presently-busy but otherwise "volunteer" ACEs varies.

Stored CEQ Mode - Each array computing element is allowed to store the complete set or some subset of CEQs, and operates its task partitioning (if optimal, as opposed to completing a task itself) by effectively selecting an optimal set of ACEs from its allotted menu of CEQs. In this mode, each ACE can be made to think that the MCC array has a different topography. Further, the array can be dynamically logically reconfigured by simply loading in a new set of array CEQs.

Other modes to support ACE-to-ACE task distribution and concurrent processing may be devised, and there are advantages and disadvantages to operating in either of the two modes just described. However, investigation of the associated pros-and-cons is outside the scope of this paper. We select the Stored CEQ Mode as the MCC operating mode for the remainder of this paper.

*See Appendix 1 for definition.



...e 1 - A generalized MCC.

SECTION 2

Databases on the MCC

We divide databases into three broad categories:

- unshared, disjoint databases
- shared, unified databases
- a combination of the preceding two.

The MCC, with its essentially independent ACEs, architecturally supports unshared, disjoint databases. By "unshared" is meant, for the MCC, that use of a particular database is restricted to one ACE. Clearly, since the ACEs are autonomous, problems in database storage and management for this situation are parochial. However, if we want ACEs to be the primary database accessors in a shared, unified** database environment, we have at least two problems:

1. accommodation of the database
2. selection of a driving DBMS.

It seems that the modular, distributed nature of the MCC is not amenable to database unification.

The "combination" situation above, possibly representative of schema/subschema DBMS implementations, might be reducible to the problem of accommodating the unified database in the MCC, since subschemata "views" [1] of the MCC's array are controllable via its characteristic equations. Database partitioning applicable to the accommodations discussed in sections 2.1.2 and 2.1.3 could be influenced by this consideration.

The remainder of this section will discuss the accommodation of the unified database in the MCC (our "first problem" on the preceding page) operating in the Stored CEQ Mode. Section 3 will discuss selection of the driving DBMS (our "second problem").

2.1 A Unified Database on the MCC

We will assume that the ACEs are to be the database sharers. In addition we will limit ourselves to the general problem of access to this database in the MCC, and not attempt to address any finer problems of access such as privacy and read/write permissions.

2.1.1 Database Accommodation No. 1

One place to store a unified database is in the user space in a

** We will use "unified" to mean "shared, unified" from here on.

storage peripheral attached to some ACE's user I/O port (see figure 2). This is equivalent to the "back-end" approach to database management [7,3]. We will disallow as an unrealistic solution, the extreme approach of attaching a storage peripheral containing a copy of the unified database to each ACE. The problems of numerous host ACEs and the required database maintenance effort are intractable using current technology. Further, duplication of the database in user space counters the unified database features of informational reliability, ease of information maintenance, and single-point access control. An advantage of the single back-end situation is simplicity in design and interfacing. A disadvantage of this approach is physical unreliability. If the ACE is disabled, use of the database is denied to the whole MCC. Similarly, if the database is stored in one single-access-channel device (which fails), database use would also be denied to the whole MCC. The latter problem can be solved by storing the database on a modular storage device having multiple access channels. That way, use of the entire database is not lost upon non-catastrophic failure. However, the reliability problem associated with accessing the database through the single ACE remains.

2.1.2 Database Accommodation No. 2

Another place to store the database is within the MCC boundary, in some number of storage devices or in one modularized storage device having multiple-input electronics. In general, the idea is similar to Marill and Stern's Datacomputer. Each storage device or input port would be electronically and systemically configured to look like an ACE from the point of view of the rest of the array. A database could be arbitrarily distributed among these "ACEs" or broken into logical entities (e.g., according to subschemata). A copy of the DBMS would be supplied to each remaining ACE permitted to have access to the database. An interesting option here is that the copies supplied do not have to be identical. That is, for whatever the reason (e.g., experimentation, optimization, operational restrictions, political, etc.), the database could be accessed by heterogeneous DBMSs. How one would structure the stored data for access by different DBMSs seems to be a good question. Perhaps Senko's [9] binary "FACT" relationship is the answer.

In our current approach, (i.e., "No. 2"), access failures can still occur, but they will have a significance no greater than that attributed to the use-denial of the affected database portions. Database-wide dependence upon the survival of a single ACE is also gone, and all (if so allowed by the CEQs) ACEs have direct access to the unified database. A new problem involving simultaneous accesses (involving alternate incremental file locking) to the database could arise now, but King and Collmeyer [5] claim to have an effective deadlock detection procedure which appears to be applicable. Since the database storage hardware is configured to look like several ACEs, that hardware architecturally lies on the MCC side of the MCC/user space boundary. This implies that the database storage hardware can have user I/O ports, function like ACEs (albeit narrowly) without requiring any exceptions to the MCC architecture, and possibly permit database updates directly from the user space and/or the ACEs (see figure 3).

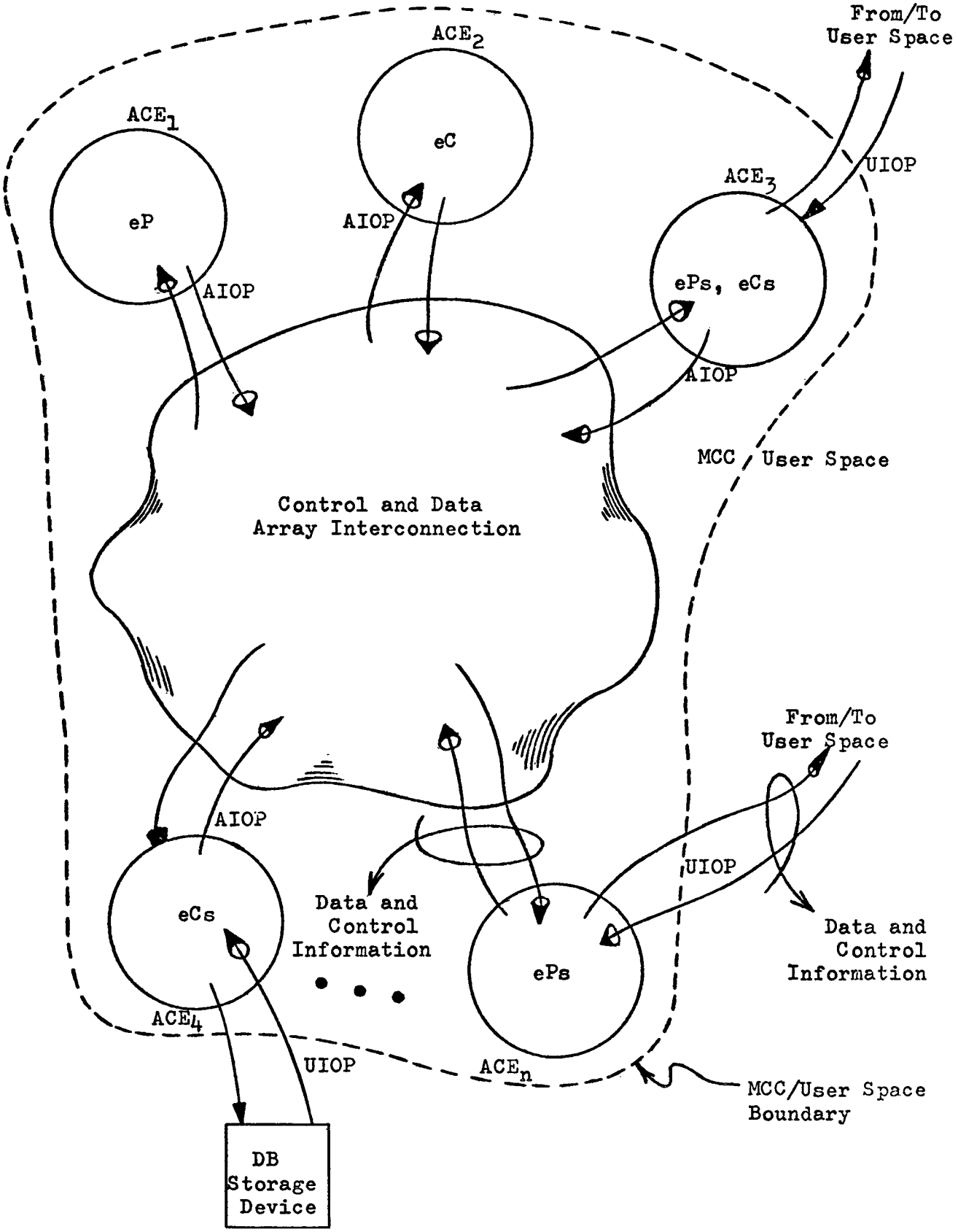


Figure 2 - A generalized MCC with an example of DB accommodation No. 1.

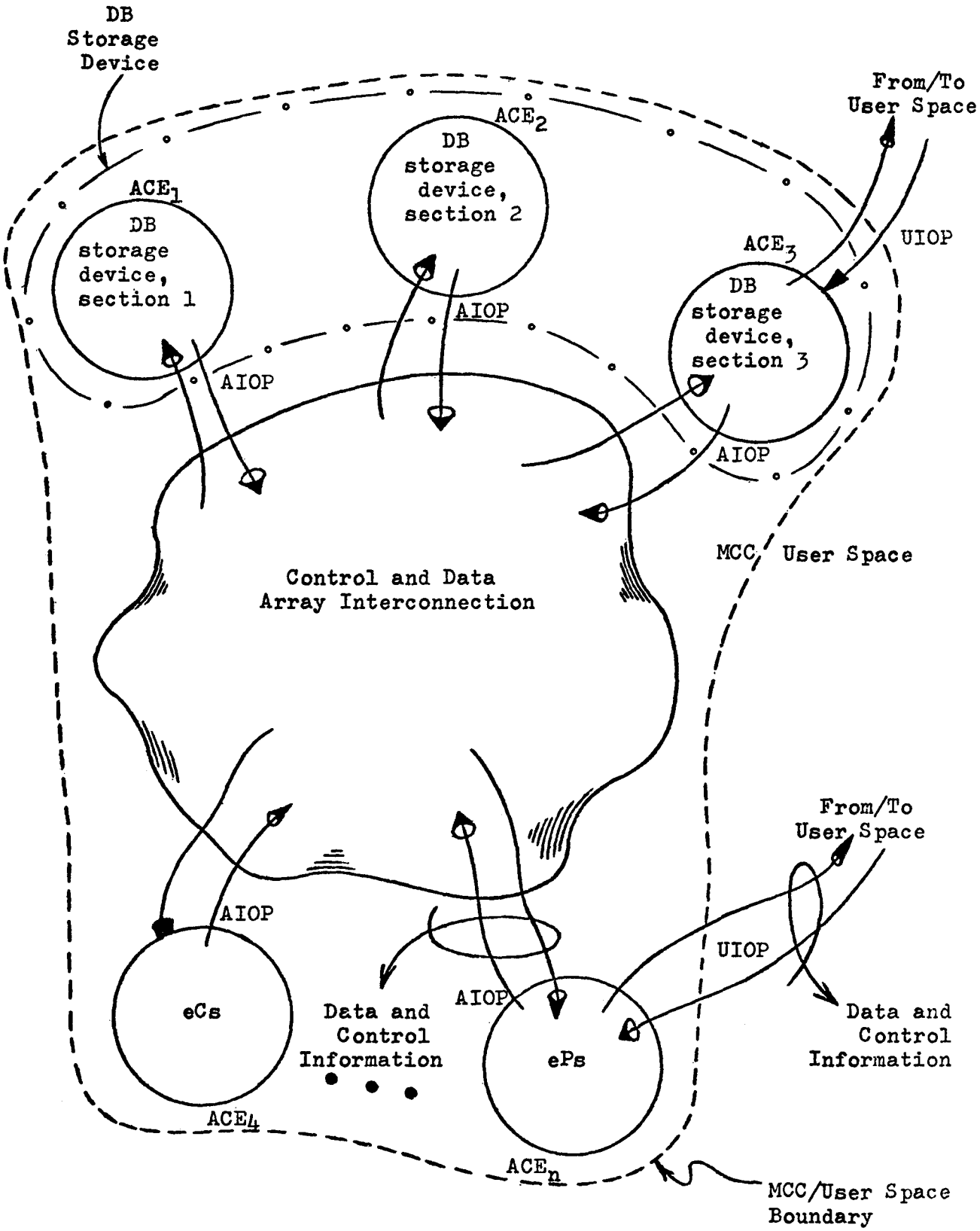


Figure 3 - A generalized MCC with an example of DB accommodation No. 2.

A disadvantage to using this ACE-disguised database storage hardware is the lack of non-volatile microminiaturized parallel-sequential bulk storage devices to replace disk and drum. This situation could present physical realization and control problems in trying to package current technology hardware (disks, drums) within the MCC. However, this all may soon be overcome by cost-effective, low-power, bubble memories, charge-coupled devices, and the like.

Another item to consider in our ACE-disguised method is that current bulk storage devices, stock commercial items not amenable to modification, will probably need special front-end electronics added to each I/O channel to make them appear as ACEs. But then, all ACEs will most likely require some array-interfacing electronics anyway for the purpose of providing electrically standard array connections.

2.1.3 Database Accommodation No. 3

A third way of accommodating a unified database on the MCC represents a merge of the first two methods (and again, resulting in a mechanism somewhat similar to the Datacomputer). This time, direct access to the database is permitted only to some number of specified ACEs, with loads, queries, and updates allowed through one or more of their array I/O ports. The immediate database user community is represented by the remaining ACEs in the array. The database itself is distributed among n storage devices (or n independent modules of one device), with each of n access channels to the storage device/modules connected to one user I/O port in each of n ACEs (see figure 4). Copies of the DBMS would be supplied only to the limited number of ACEs having direct access to the database. Again, there is no theoretical restriction against heterogeneous DBMSs, but the stored-data structure problem mentioned in section 2.1.2 would apply here also. In operation, should some database service request enter an ACE that does not have direct access to a required portion of the total database, ACE-to-ACE interaction will be necessary to complete the task.

An advantage of this third database accommodation method is that only the group of ACEs having direct access to the database need be supplied with copies of the DBMS(s). Another advantage (over the method of section 2.1.2) is that no unusual ACE-mimicking interfacing electronics need be added to the database storage hardware. Although nearly direct access to the database is maintained, we incur another level of delay between the user-ACEs and the database (a disadvantage as compared to accommodation No. 2). Such is a cost for manageability. Again, as with method No. 2, the simultaneous access possibility would still require attention. Another disadvantage to our third method involves the programming difficulty; inter-ACE cooperative software is required. For section 2.1.2, no such interaction would be required since knowledge of the database structure is resident with each copy of the DBMS supplied to the ACEs. User access to the database in our current method is through the direct-access subset of ACEs, which must determine by itself, the servicing procedure.

The general structure of the Relational Associative Processor (RAP) of Ozkarahan, Schuster, and Smith [8] approximates one

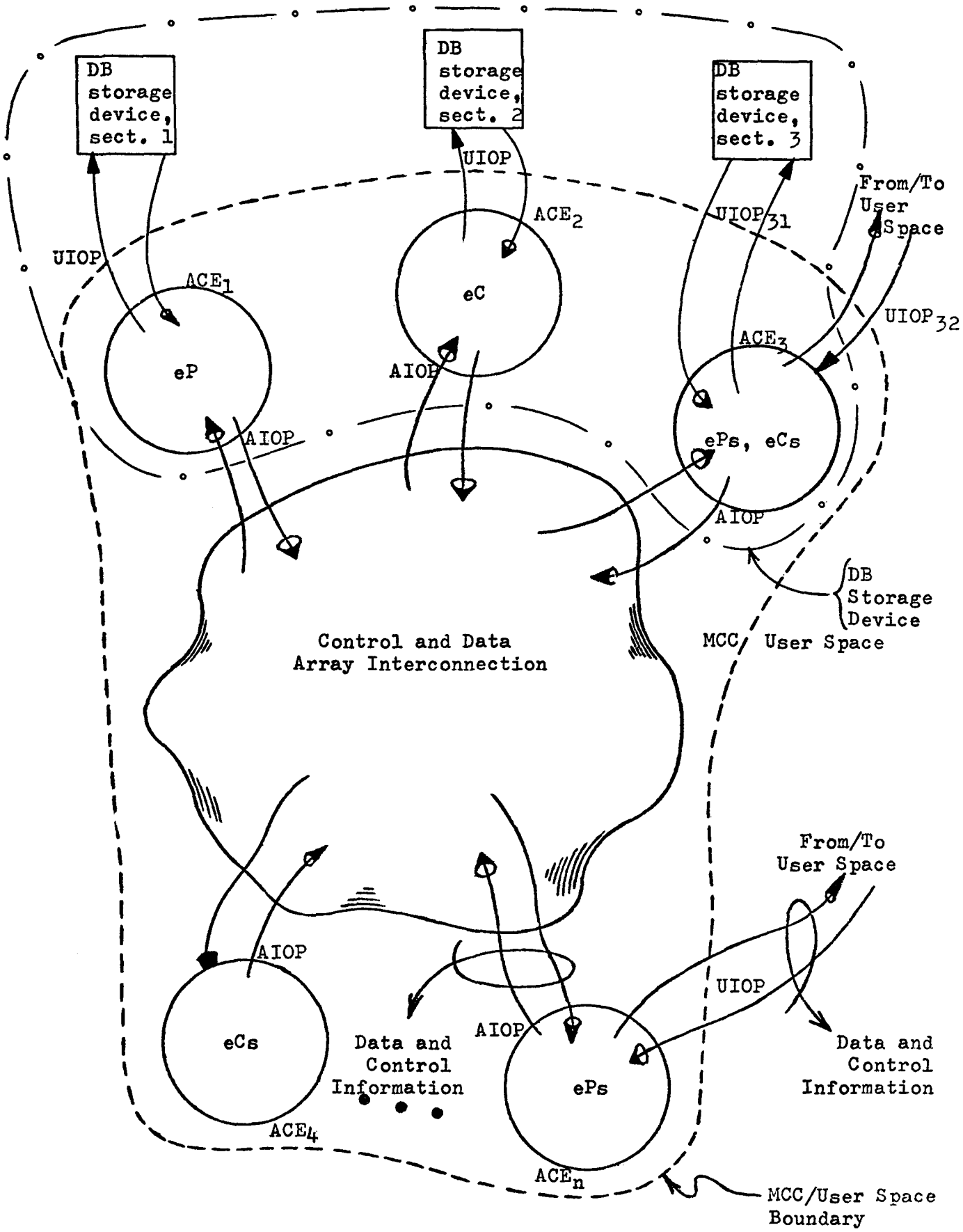


Figure 4 - A generalized MCC with an example of DB accommodation No. 3.

module (i.e., ACE plus database storage device) of our current database accommodation method. Mapping their device onto the MCC, the "cells" of their RAP would correspond to the database storage device. Their general purpose computer and the RAP's "controller" and "set function unit" would correspond to an ACE, and their "users" port would correspond to the same ACE's array I/O port. The connections from the RAP's "cells" to the "controller" and "set function unit" would map to the ACE's user I/O port. If the RAP is intended to contain an entire unified database, then it shares the same unreliability problem as did our accommodation of section 2.1.1. However, if only some segment of the unified database is stored on a given RAP, then in a parallel implementation, the complete set of RAPs would resemble an instance of our accommodation No. 3.

2.1.4 The Chosen Accommodation

Although none of the three accommodations appears attractive, we believe that the least troublesome one is the third, and we will select it for implementation on the MCC. That method provides architectural consistency (with the MCC), localized but distributed (over a limited portion of the MCC) direct database access control, and reliability.

SECTION 3

DBMSs on the MCC

We proceed, using database accommodation No. 3 as described in section 2.1.3. A problem for the corresponding DBMS software was recognized in that section as a relative difficulty of, or perhaps complexity in, producing that inter-ACE cooperative, DB-accessing software. As an aid to understanding the database accessing process and the reason for the complexity, consider the following simple sequence. Let some ACE in the MCC array query another ACE which is configured especially for data manipulation and high I/O rates, and (therefore apriori assigned to be) a member of the direct-DB-accessing set of ACEs (the "DB Storage Device" in figure 4). The original call may have been made on the basis of CEQ matching or some other method. The first-contacted ACE calls upon other ACEs in the database Storage Device to provide their portion of the service. This assumes that complete service could not be provided with the database portion allocated to the first-called ACE. This action implies a DBMS whose schema will optimize overhead (including time during which other users might be locked out of the database), turn-around time, and any interconnection net occupation time. Hence, the software complexity. With these implied objective functions in mind, and limiting ourselves to the three DBMSs stated in section 1, we will proceed to evaluate those schemata for application to the MCC.

3.1 A Relational DBMS on the MCC

Ozkarahan, et al., cite a generality sufficient to build other

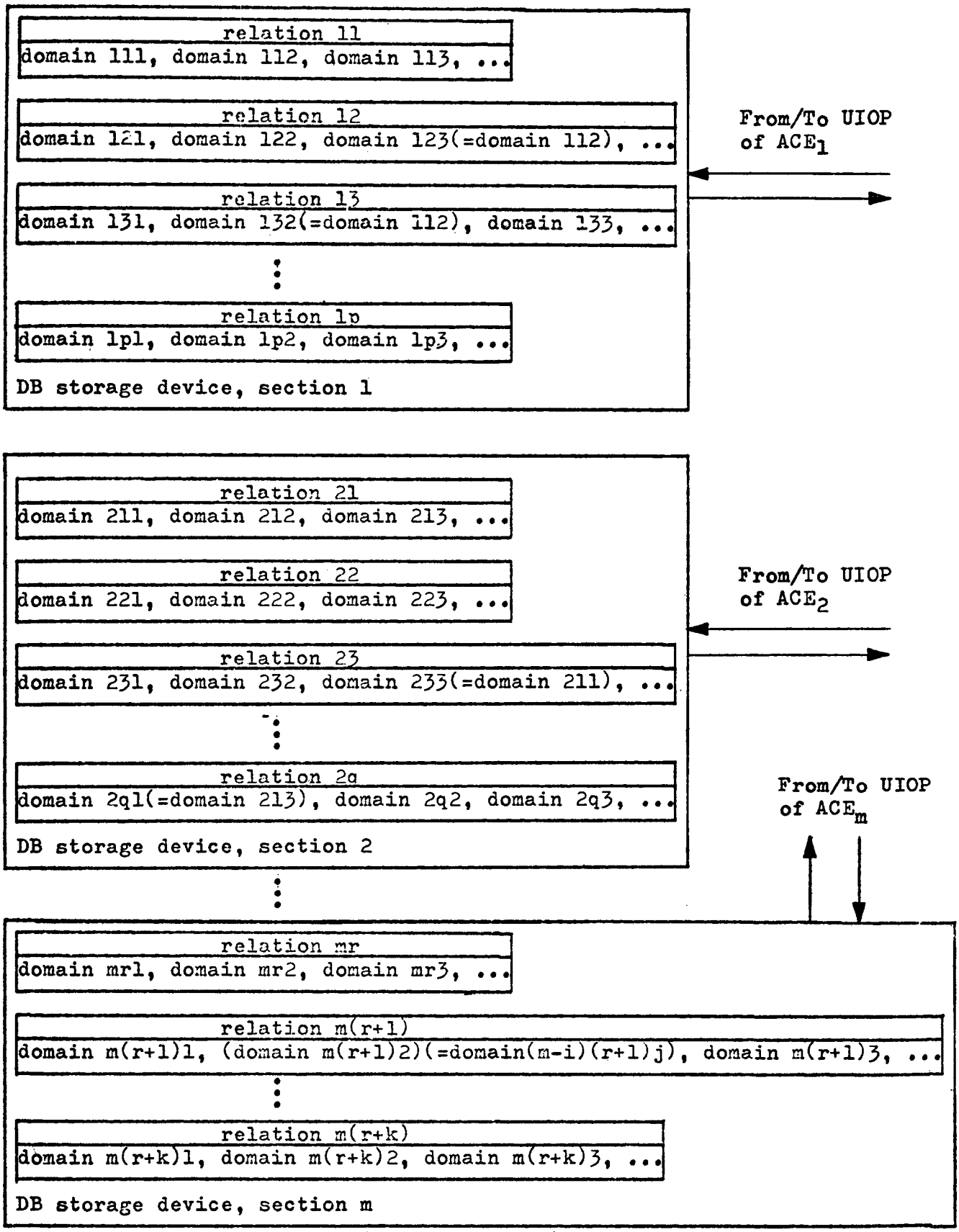
data structures, an ability to convey the same information as other data structures, and an ability to contain that information with a high degree of data independence, as reasons for selecting the relational model for emulation in hardware (the RAP). (We attempt a paraphrase by saying they selected the relational model because it is factorable from other DBMS models.) Gerritsen [4] states five relative advantages of the relational model as simplicity, uniformity, data independence, integrity, and security. For our purposes, we consider the fact that the data itself contains the relation-linking information (the primary key/related domain sets) as a principal advantage. To get to the desired data if not in the storage module at hand, an exchange of information among the ACEs in the Database Storage Device (recall figure 4) would be needed to link the involved relations. When the sought relation(s) is(are) found, the query/update can be executed. Another advantage of the relational schema is that new (but properly constructed) relations can be added to the database by storing them wherever there is room in our modular, distributed storage hardware, exercising the relational model's data independence. In practice it may be desirable to ensure that when a number of relations have identical domains, those relations are accessible through one ACE. This could help to reduce inter-ACE traffic.

Identical domains among the different relations, inherent in the relational schema, introduce a burden of repetitive data storage. Although the attributes of these domains may be repeated data, they are not redundant, supplying information for semantical linking. Also, the modular structure of relations favors the MCC architecture. Figure 5 illustrates an example of relational data storage in the MCC.

3.2 A Network DBMS on the MCC

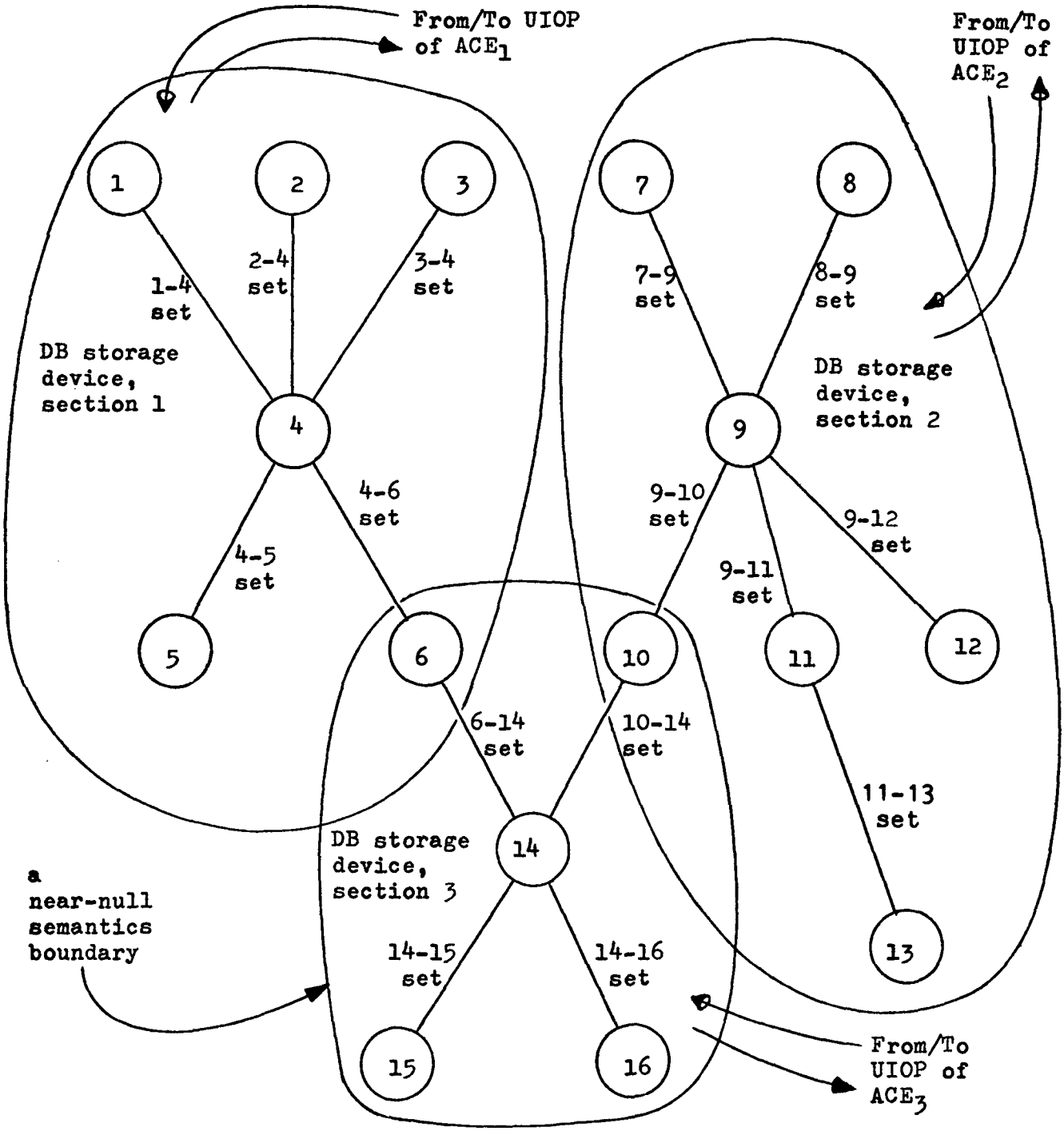
The DBTG model appears to be more difficult to operate on the MCC as compared to the relational DBMS, especially if random database record storage is allowed. Semantic information that defines the owner-member set lies in the imaginary space between data records in the form of fixed-path pointer information. In the random record storage case, this could result in a high inter-ACE overhead. However, this structure can be made somewhat modular if, on a (data) case-by-case basis, a satisfactory partitioning of the whole database along null (or nearly null)-semantic lines can be achieved (see figure 6). A satisfactory partitioning as just described cannot be guaranteed for the DBTG DBMS since that partitioning is dependent upon record semantics rather than the DBTG schema. Therefore, the DBTG model cannot be applied to the MCC with equal facility for all situations. As an example, consider a database that has one or more semantically defined partitions whose storage requirements exceed the capacity of one of the ACE-controlled hardware storage devices. This forms a data dependence between ACE-boundaries, and can result in an unsatisfactory overhead in operations involving that data. Some record repetition results from our suggested modularizing, and such records are identified as exactly those within the intersection of modules.

Gerritsen points out what are considered to be advantages of the DBTG DBMS; "...it provides a more natural structure for modelling the world, the semantics is built into the structure, and data repetition



Each mr is an indexing pair (not a quantity).
$0 \leq i \leq m-1$, i an integer; $1 \leq j$, j an integer; $0 \leq k$, k an integer.

Figure 5 - An example of relational data storage in a generalized MCC using DB accommodation No. 3.



Circles represent records in this Bachman diagram.

A copy of records 6 and 10 are required in each storage device.

Figure 6 - An example of the DBTG DBMS on a generalized MCC using DB accommodation No. 3.

is reduced resulting in a more consistent database." Further, he states about the relational and network models, that "...the only difference between the two models is a question of generality: The relational model is more general because it does not consider implementation, e.g. the storage structures of data." Refering to Gerritsen's "difference" statement, our concern in this paper exactly involves implementation. So far, we have taken a computer architecture and two DBMSs and tried to fit the DBMSs to the architecture. The relational model fit with no alteration of its attributes. However, the "advantages" of the DBTG DBMS became less apparent on the MCC. The DBTG DBMS seems to be better suited to machines having less modular hardware data storage than we have adopted here.

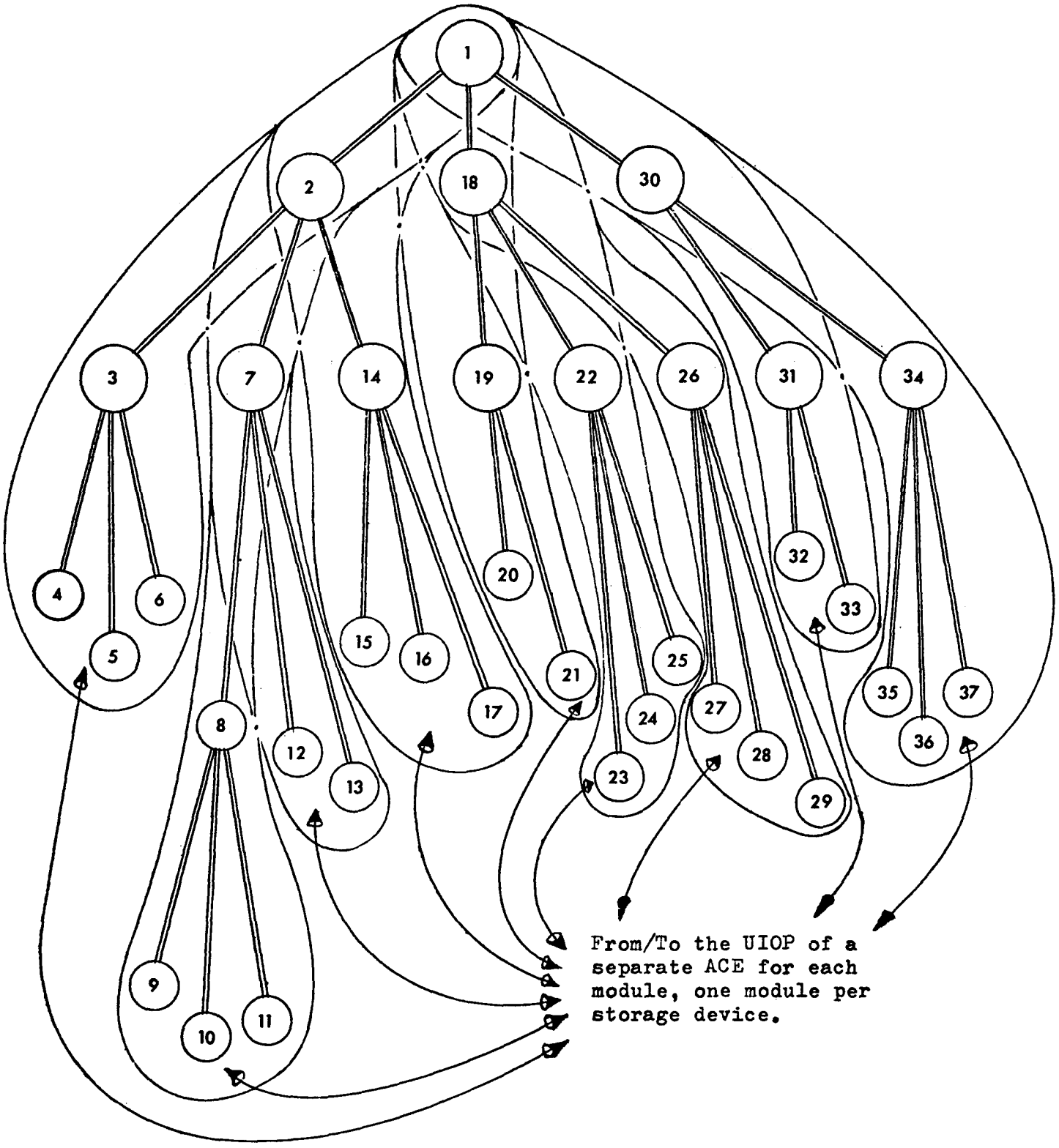
3.3 A Hierarchical DBMS on the MCC

The IMS DBMS on the MCC, as a schema for a unified database on the MCC, appears to be the least suitable among our three aspirants. However, some IMS-search compatible modularizing of the database, with one module under the control of only one ACE, can be accomplished if we encapsulate along views defined by routes of successful IMS searches that go directly from the root to leaf nodes (to include all the leaf nodes at the end of a travelled branch). This results in structures which admittedly do not fit too well with the IMS top-to-bottom, left-to-right, front-to-back general search algorithm. The primary reason for modularizing the IMS database in this manner is again to reduce inter-ACE traffic, and our proposal is not claimed to be optimal to that goal; a much more in-depth study would be needed to ascertain that. Near the end of this section, we suggest an alternative database storage method for a special case, the alternative being just a modification of our current approach. But in general, since a better database storage method for the IMS is not obvious at this time, we will proceed with the one described above.

The modularizing procedure envisioned would be influenced by the number of ACEs assigned to the formation of the hardware storage device for a given unified database. Further, due to the ordered nature of the database on hierarchical DBMSs, a module definition will be fairly rigid. Finding room to add new segments after storage might depend upon good fortune or the foresight to strategically allocate extra storage space to permit such expansion. Adding segments or ACEs to expand the capabilities of our MCC-based IMS DBMS could require a re-modularizing of the entire database.

Once the number of ACEs participating in the hardware storage device is determined, the IMS tree can be divided into that number of modules (see figure 7). Clearly, the trivial situation is one where there is only one module containing the whole tree, and the maximum situation is characterized by having the number of modules equal the number of leaf nodes.

Besides the inflexibility of the IMS on the MCC, another disadvantage is that as the number of modules increases, so does the number of repetitively stored segments. For example, for the trivial case described in the preceding paragraph, no segment storage repetition is required. For the maximum situation described, every



Each circle represents an IMS segment.

Figure 7 - An example of the IMS DBMS on a generalized MCC using DB accommodation No. 3.

segment other than those at the leaf nodes would be repetitively stored. It seems reasonable to say that the hierarchical model is increasingly resistant to increasing modularization (as we defined it for the IMS).

For the special case where a storage module intersection is "small" and the probability of searching outside of that intersection is acceptably low, one might consider assigning that area to the direct control of one ACE. The rest of the modules, which could be envisioned as lobes or fingers extending from the intersection area, would be assigned to other ACEs. Another (careful) examination of figure 7 may aid the reader in visualizing this latest partitioning.

3.4 Selection of a DBMS for the MCC

As a companion to the database accommodation selected in section 2.1.4, the relational DBMS is selected as the most acceptable, among the three candidates, for application to the MCC. The relational model is intrinsically modular, essentially insensitive to storage position (but some inter-ACE traffic optimization might be possible on a semantics basis), amenable to the introduction of new relations, and relatively simple to implement. But in the absence of data storage optimized for the minimization of inter-ACE traffic, it is not clear that such traffic would have an acceptably low event frequency. Intuitively, if the first-contacted ACE does not have the data in its directly controlled storage device, one broadcast of match-seeking domains to all ACEs in the DB Storage Device would be required for each needed relation. This implies that the inter-ACE traffic density would be a function of the complexity of the query.

The DBTG DBMS places second for our purpose. It does not naturally operate in a modular environment such as was postulated here. However, some semantic clustering of data can result in fairly independent data modules. But since such clustering is necessary to efficient operation in the MCC, the DBTG DBMS is not as good a solution on the MCC as the relational model. A less than acute awareness of the data semantics when installing new records could significantly impact the efficiency of database operations. During normal, efficient operation, it seems that inter-ACE traffic would appear as bursts of activity, and only when records at the module intersections are accessed. Whether or not this represents a more acceptable situation as compared to the relational model is not clear. Possibly in support of the comments by Gerritsen mentioned in section 3.2, we believe that a DBTG DBMS is an instance (snapshot) of a relational implementation. Conversely, the relational model could be thought of as a DBTG DBMS with dynamically redefinable semantics. Further, we see IMS data structures as special cases of DBTG structures.

The IMS places third, and essentially is not acceptable for application on the MCC. It is resistant to modularization in that segments require greater repetition in storage as the number of modules (such as we have defined) increases. In the maximum-number-of-modules situation, we approach nearly a duplication of the entire unified database for every ACE involved with its

storage. The whole structure then approximates a database accommodation scheme discussed in section 2.1.1. That scheme was discarded as unrealistic.

In our study of inter-ACE traffic volume, we encountered a peculiar but understandable phenomenon with the IMS DBMS. As the modularization of its tree increased, the inter-ACE traffic decreased, since the resultant repetition of stored segments reduces the need for inter-ACE communications. More globally, we deduce that in a graph with increasingly resource and informationally autonomous nodes, the necessity for internodal communications decreases.

SECTION 4

Summary

Three DBMSs served as vehicles for the non-exhaustive study of DBMS models applied to the MCC. The relational model was the most acceptable, followed by the network and hierarchical models, respectively. Repetition of stored data occurred with each of the three DBMSs and in increasing amounts according to our most-to-least acceptable ordering. Some exceptions may be found. Also, all three models derived at least some performance improvement from semantic ordering in the MCC.

The basic MCC architecture required no modification to implement any of the unified database accommodations.

APPENDIX 1

Some Definitions as Used in this Paper

accommodation - Containment of a database in a computer system.

array computing element (ACE) - Exists only at an array node, with one ACE per array node. The ACE consists of one or more functionally connected elemental processors (ePs) and/or computers (eCs). The ACE is capable of the autonomous execution of at least some subset of the MCC's instruction repertoire.

array node - The logical intersection of data and control information channels within the array boundary.

characteristic equation - Describes the processing, computational, and memory properties of an array computing element.

elemental computer (eC) - Consists of one or more uniprogram microcomputers and/or other uniprogram computing hardware, with each eC containing one or more microprocessors or other processing hardware, respectively.

elemental processor (eP) - consists of one or more microprocessors or other data processing hardware arranged to perform a fixed set of information processing functions.

microcomputer (μC) - A computer essentially totally dependent on one or more component microprocessors in the performance of its arithmetic logic unit functions.

microprocessor (μP) - A (usually) large scale integration (LSI) realization of an arithmetic logic unit (ALU), including the LSI and/or other lesser scale control and memory hardware necessary to the transformation of input to generated result in the LSI ALU.

shared - (only when describing unified databases) A unified database where the users of the subsumed databases now use the unified database.

supercomputer - A computer having other computers as subsets.

unified - (only when describing databases) A database formed by performing (essentially) a disjunctive operation on smaller databases. An effect is the reduction of data storage redundancy within the data set formed by the smaller databases.

user space - That physical space, external to the MCC, occupied by information handling facilities unnecessary to the internal, self-sufficient, non-malfunctioning operation of the MCC. Typically, this space includes attendant I/O devices and user-humans, and excludes any unscheduled maintenance and/or repair hardware and/or people.

APPENDIX 2

Abbreviations Used in this Paper

ACE - array computing element
ALU - arithmetic logic unit
AIOP - array input/output port
CEQ - characteristic equation
DB - database
DBMS - database management system
DBTG - Data Base Task Group
eC - elemental computer
eP - elemental processor
eP/C - elemental processor and/or computer
IMS - Information Management System
I/O - input/output
LSI - large-scale integration
MCC - multi-computer computer
μC - microcomputer
μP - microprocessor
UIOP - user input/output port

REFERENCES

[1] CHAMBERLIN, Donald D., and GRAY, J. N., and TRAIGER, I. L., "Views, authorization, and locking in a relational data base system", DATABASE MANAGEMENT SYSTEMS, Vol. 1 of The Information Technology Series, Ed. by Ben Shneiderman, AFIPS Press, 1976, pp. 79-84.

[2] CHAMBERLIN, Donald D., "Relational Data-Base Management Systems", ACM Computing Surveys, vol. 8, No. 1, March 1976, pp. 43-66.

[5] COLLMEYER, Arthur J. - see KING, 1976.

[3] CULLINANE, J., and GOLDMAN, R., and MEURER, T., and NAWARA, R., "COMMERCIAL DATA MANAGEMENT PROCESSOR STUDY (BABBAGE)", Cullinane Corp., 20 William Street, Wellesley, Mass. 02181, December 1975.

[7] FISHER, P. - see MARYANSKI, F., 1975.

[4] GERRITSEN, Rob, "THE RELATIONAL AND NETWORK MODELS OF DATA BASES: BRIDGING THE GAP", DATABASE MANAGEMENT SYSTEMS, Vol. 1 of The Information Technology Series, Ed. by Ben Shneiderman, AFIPS Press, 1976, pp. 127-131.

[3] GOLDMAN, R. - see CULLINANE, 1975.

[1] GRAY, J. N. - see CHAMBERLIN, Donald D., "Views, ...", 1976.

[5] KING, Paul F., and COLLMEYER, Arthur J., "Database sharing - An efficient mechanism for supporting concurrent processes", DATABASE MANAGEMENT SYSTEMS, Vol. 1 of The Information Technology Series, Ed. by Ben Shneiderman, AFIPS Press, 1976, pp. 107-111.

[11] LOCHOVSKY, F. H. - see TSICHRITZIS, D. C., March 1976.

[6] MARILL, Thomas, and STERN, Dale, pp. 107-111. "The Datacomputer - A network data utility", AFIPS Conference Proceedings, 1975, vol. 44, pp. 389-395.

[7] MARYANSKI, F., and FISHER, P., and WALLENTINE, V., "USABILITY AND FEASIBILITY OF BACK-END MINICOMPUTERS", U. S. Army Computer Systems Command (USACSC grant no. DAHC04-75-G-0137), Fort Belvoir, Va. 22060, June 1975, prepared by Dept. of Computer Science, Kansas State U., Manhattan, Kansas 66506.

[3] MEURER, T., - see CULLINANE, 1975.

[3] NAWARA, R., - see CULLINANE, 1975.

[8] OZKARAHAN, E. A., and SCHUSTER, S. A., and SMITH, K. C., "RAP - An associative processor for data base management", National Computer Conference, 1975.

[10] RANDALL, L. Frank - see TAYLOR, Robert W., March 1976.

[8] SCHUSTER, S. A. - see OZKARAHAN, 1975.

[9] SENKO, Michael E., "Specification of Stored Data Structures and Desired Output Results in DIAM II with FORAL", (no date), Mathematical Sciences Department, IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

[8] SMITH, K. C., - see OZKARAHAN, 1975.

[6] STERN, Dale, - see MARILL, 1975.

[10] TAYLOR, Robert W., and RANDALL, L. Frank, "CODASYL Data-Base Management Systems", ACM Computing Surveys, vol. 8, No. 1, March 1976, pp. 67-103.

[1] TRAIGER, I. L. - see CHAMBERLIN, Donald D., "Views, ...", 1976.

[11] TSICHRITZIS, D. C., and LOCHOVSKY, F. H., "Hierarchical Data-Base Management: A Survey", ACM Computing Surveys, vol. 8, No. 1, March 1976, pp. 105-123.

[7] WALLENTINE, V., - see MARYANSKI, 1975.