1-1-1988

Database machines in support of very large databases

Mary Ann Kuntz

Rochester Institute of Technology
School of Computer Science

Database Machines in Support of Very Large Databases

by
Mary Ann Kuntz

A thesis, submitted to

The Faculty of the School of Computer Science,

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Systems Management

Approved by: _____
Professor Henry A. Etlinger

_____
Professor Peter G. Anderson

_____
Professor Jeffrey Lasky

Date: 10/12/88

Title of Thesis: Database Machines In Support of Very Large Databases

I Mary Ann Kuntz hereby deny permission to reproduce my thesis in whole or in part.

Date: October 14, 1988 Mary Ann Kuntz

Abstract

Software database management systems were developed In response to the needs of early data processing applications. Database machine research developed as a result of certain performance deficiencies of these software systems. This thesis discusses the history of database machines designed to improve the performance of database processing and focuses primarily on the Teradata DBC/1012, the only successfully marketed database machine that supports very large databases today. Also reviewed is the response of IBM to the performance needs of its database customers; this response has been in terms of improvements in both software and hardware support for database processing. In conclusion, an analysis is made of the future of database machines, in particular the DBC/1012, in light of recent IBM enhancements and its immense customer base.

Table of Contents

Chapter 1 - Introduction to Database Machines

Chapter 2 - A History of Database Machine Research

Chapter 3 - The Teradata DBC/1012

Chapter 4 - Database Processing Solutions by IBM

Chapter 5 - The Future of Database Machines

Table of Figures

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 1

Introduction to Database Machines

I. Introduction

The theory of dedicating hardware to support the performance of database management functions was first proposed by Slotnick in 1970 [SLOT 70]. By associating search logic with the apparatus of the read/write head of a fixed head (i.e. head-per-track) disk, processing of the data stored on each track could be accomplished in a fashion referred to as "on-the-fly". Selection of data[1] that satisfied some specific criteria, would be attained at the disk level; the data would then be forwarded to main memory for processing. Thus the notion of database machine technology was introduced. Since that time, this field has been the focus of a great deal of research and theoretical analysis.

A. Out of The Labs

It is only within recent years however, that database machines have gained acceptance in management information systems (MIS) communities. Credit for this emerging confidence must be attributed largely to the efforts of Teradata Corporation whose Data Base Computer, the DBC/1012 [DBC/ 86], is the only database machine on the market today capable of supporting very large, high transaction volume database installations.[2] Other vendors, Britton-Lee for example, have produced database machines targeted toward smaller, "department-size" databases.

B. IBM

Visibly absent from the database machine arena is International Business Machines (IBM). Many look to IBM to establish standards for the computer industry. The skepticism of some regarding database machines is fueled by IBM's silence on the subject. But while IBM has made no announcements of a database machine per se, an analysis of its hardware/software integration strategies over the next few years indicates an emphasis on the expanding role that hardware components will play in the performance of database functions [SICI 87]. Within a few years, IBM customers will likely find that database activities have been off-loaded to separate processors.

II. Database Beginnings

To gain a true appreciation for database machines which have now evolved to fill a significant niche in the industry, it is worthwhile to look back at the beginnings of database technology itself. It is through an understanding of why database software developed that we can better measure the effectiveness of dedicated hardware. Database machines pick up where software database managment systems leave off.

A. Why Database?

As computer applications developed, there emerged a need for centralized control over the volumes of data that were being produced. Three issues became most important: separation of the data, storage considerations and treating data as a resource.

---

1 Stored in records or "tuples".

2 Britton-Lee has introduced a large system database machine, the BL8000 [BRAU 87b] but it is met with skepticism by some because it does not offer support for the MVS operating system [BRAU 87d].

1. Separation of Data

As programming applications developed, data was stored on fixed format sequential files, most often on magnetic tape. The format of the data records was specified to suit the particular project. A typical record for a payroll program for instance, might contain fields for an employee's last name, first name, street address, city, state, zip code and salary, as shown in Figure 1. These fields were defined by data type and length.

Last_Name	First_Name	Address	City	State	Zip Code	Salary
Smith	Quincy	46 Andrews St.	Oakfield	KY	46830	55000

Figure 1. Payroll Record

Any programs making use of this data were to incorporate its format into the actual code - the fields conforming to the established structure. Records read into a program or written to a file by a program could never deviate without causing an error. Figure 2 shows the PL/1 record format of this payroll record's structure.

```
PAYROLL_REC;
        LAST_NAME     CHAR (12);
        FIRST_NAME    CHAR (10);
        ADDRESS       CHAR (15);
        CITY          CHAR (10);
        STATE         CHAR (2);
        ZIP_CODE      CHAR (5);
        SALARY        FIXED DEC (7);
```

Data structure with fixed length fields.

Figure 2. PL/I Payroll structure.

Eventually, the format of the data would have to change; new fields added, existing fields removed or altered in some way. This occurrence for a data dependent scenario such as that just described, demanded a great deal of program maintenance. Even if a program did not use a new field that was added, the code would have to be modified to read in or write out the record according to its new configuration. Recompiliation was always required (see Figure 3).

Action Required	Programs using new field		Programs not using new fields	
	Traditional	Database	Traditional	Database
File Definition change	Y	N	Y	N
Logic change	Y	Y	N	N
Recompile	Y	Y	Y	N
Test	Y	Y	N	N

Figure 3. Adding a new field.

It became increasingly evident that the structure of records had to be made independent from application programs. Programming changes of the nature described above were commonplace and simply too costly.

2. Storage Considerations

Virtually every department in newly computerized organizations found applications for data procsessing. Sales, payroll, employee benefits - each increased its productivity by developing programs to organize pertinent information and to keep it current.

Soon however, problems arose because much of the same information was duplicated across the different programming groups. Storage space on both tape and disk rapidly became consumed by redundant data. One solution was to have the data that was used by these different applications stored once, in one central place.

3. Data As A Resource

Organizations quickly began to realize that their data was more than just information; it was seen as a resource capable of being misused, stolen, corrupted or lost. Consideration was given to the need for control over this valuable resource. Its integrity had to be ensured and security over it maintained.

B. Database Solutions

Database Management Systems were developed in an effort to solve the problems introduced by the situations discussed above. See [DATE 86] for more information on the history of database systems.

1. Data Independence

The initial benefits provided by software database management systems fall into two broad categories - data independence and centralized control. The former involved defining "views" of data [DATE 86] so that only those portions of an entire record needed by a program were ever accessed by that program. Knowledge of the format of a record or how it was stored was no longer required. Changes to records no longer affected the programs using them.[3] Access to data became increasingly more flexible.

Programming was simplified as well. Since the programmer had only to issue "calls" to the database, it was not necessary to keep track of input or output files, physical location of data, etc. Obviously program maintenance was greatly reduced. Changes to the data had little or no effect on the program itself. Often the only requirement to be met was to recompile (see Figure 3).

The independence of the data from the program by virtue of the database management system facilitated the continued growth of databases. Systems programmers who maintained the actual physical files or "datasets" could for example, move a file to another device or allocate more space for it on the same device without affecting the application programs in the least. Thus performance of the system could be enhanced invisibly to the users of the data.

2. Centralized Control

The other major advantage offered by software database management systems was the ability to implement control over the database through a central administrator.[4] This control was manifest not only in the form of security control, but also in control over database efficiency and performance.

---

3 Unless, of course, changes were made to fields contained in a program's view.

4 Commonly referred to as the Database Administrator, or DBA.

Security over data as a resource has become more and more critical over the years. By means of facilities inherent in the database management systems, the DBA became capable of restricting the use of data to only those users specified. Some users were given "read only" access; others the ability to update records; and still others, no access at all. In some systems, a security "trail" could be implemented in an effort to determine the source of unauthorized access attempts.

Centralized control afforded sharing of data among various applications, reducing or possibly eliminating data redundancy. Due to the highly sensitive nature of some data it was, of course, necessary to store it redundantly in a separate, confidential database. In general however, the use of the same data by more than one application was coordinated through the DBA.

Data integrity was an additional benefit of centralized control. If the same data were stored separately by each application that made use of it, updates made in one instance probably would not be carried over to the other data store. Difficulties arose in determining which of the data was current and accurate. With one centralized database, the currency and integrity of the data could be more readily maintained.

Finally, centralized control facilitated the implementation of standards - standards of programming style, documentation and methods of accessing data. With an overall plan for the database environment in a given organization, application development could be greatly enhanced and data treated as a justifiably valuable asset.

III. Problems Arising From Software Database Solutions

Even though database management systems held answers to many of the problems facing data processing centers, some issues remained unresolved and some new matters needed to be addressed. Five key issues illustrate the importance of this situation. The pursuit of further solutions has lead many to the continuing research on database machines.

A. Referential Integrity

Probably the single most important problem that has not yet been satisfactorly met by software database management systems is referential integrity [DATE 86]. Briefly, this situation arises when two or more relations in a relational database have a key field in common. This field might be for example, SUPPLIER-NAME found in both a relation called SUPPLIER (see Figure 4) and also in the PART-ORDER table.[5]

In this example, the SUPPLIER table contains information about vendors that supply materials to the company. The PART-ORDER table on the other hand, keeps track of parts used and when to reorder more. If the company were to end its business dealings with supplier JONES, the database entry for that supplier should be deleted from the relation SUPPLIER. But without referential integrity, a reference to supplier JONES might still exist in the PART-ORDER relation, putting the database in an inconsistent state. If properly implemented, referential integrity would make it known that the second table contained the same supplier's name.

---

5 In relational databases, the terms "relation" and "table" are used synonomously.

SUPPLIER

Supplier_Name	Supp_ID	Address
Smith	A-250	Biscayne, MD
Jones	M-477	DuBois, PA

PART_ORDER

Part_Name	Reorder_ct	Supplier_Name	Inventory
Drill bits	100	Smith	10000
Nails	1000	Jones	9500

Figure 4. Database Relations.

Several solutions have been offered for the inconsistencies that result from the lack of referential integrity [DATE 86] but as yet, no completely satisfactory resolution has been made. IBM has recently announced that DB2 Release 2 will further support referential integrity [IBM 288-196]. A further example serves to describe these solutions and illustrate the problems inherent in them.

Let us assume that SUPPLIER-NAME has been specified as a "primary key" in the SUPPLIER table, and as a "foreign key" in PART-ORDER. A primary key is simply a unique identifier; no two records in SUPPLER may have the same value for the primary key field. Also, every SUPPLIER record must contain some value for SUPPLIER-NAME (i.e. null values are not allowed in primary key fields). A foreign key on the other hand, is a field whose value matches that of a primary key of some other relation. In this case, SUPPLIER-NAME is a foreign key in PART-ORDER. It allows references to be made between this relation and the SUPPLIER relation.

The questions to be answered in trying to solve the problem of referential integrity are:

- What should be done when an attempt is made to delete a primary key value from a table when that primary key is a "target" for a foreign key in another table?
- What should be done when an update for such a primary key value is attempted?

1. Cascading

The first method proposed for maintaining referential integrity is to "cascade" the deletion or update of a primary key value to the corresponding foreign key value in another relation. In our example, if supplier JONES changed its name to JONES & SMITH, a cascading update to the foreign key value in the PART-ORDER table would be appropriate (see Figure 5). If however, JONES were deleted from SUPPLIER, it would not be appropriate to delete those foreign key records from PART-ORDER, since this would adversely affect the information concerning parts that were previously supplied by JONES.

2. Restriction

Another solution is to restrict updates and deletions of primary key values to only those which have no corresponding foreign key in another relation. This poses some problems in our example, JONES could not be deleted from SUPPLIER until such time as there were no more parts in inventory that JONES supplied. Such an inconsistency might lead to future orders being placed with JONES when none should be. Also, there is no viable reason why an update such as a name change should not be implemented for JONES in the SUPPLIER table, just because PART-ORDER contains the same value in a foreign key field.

SUPPLIER

Supplier_Name	Supp_ID	Address
Smith	A-250	Biscayne, MD
Jones & Smith	M-477	DuBois, PA

PART_ORDER

Part_Name	Reorder_ct	Supplier_Name	Inventory
Drill bits	100	Smith	10000
Nails	1000	**Jones & Smith**	9500

Figure 5. Cascading a key change.

3. Nullification

The third alternative is to nullify a relation's foreign key value when that value corresponds to an updated or deleted primary key value from another relation. This would result in our example in some sort of error when new parts were to be ordered from a supplier whose name had been blanked out (see Figure 6). A special exit would probably have to be programmed to handle such an occurrence. While acceptable solutions in some cases, it appears that none of these alternatives provide an adequate overall answer.

SUPPLIER

Supplier_Name	Supp_ID	Address
Smith	A-250	Biscayne, MD
~~Jones & Smith~~	~~M-477~~	~~DuBois, PA~~

PART_ORDER

Part_Name	Reorder_ct	Supplier_Name	Inventory
Drill bits	100	Smith	10000
Nails	1000		9500

Figure 6. Nullifying a key value.

B. Cost

A second problem with software database solutions is that software database management systems can be very expensive.

1. Purchase Price Plus

In 1985, prices quoted for the most popular IBM mainframe compatible software database management systems were astounding [PERR 85]. Datacom/DB from Applied Data Research ("ADR") had a price tag of nearly $110,000 . This did not include the cost of ADR's fourth generation language IDEAL, or any of its other database-related products.[6] The cost of an ADR system containing these primary components ranged from $225,000 to $275,000. Cullinet Software's IDMS, Datacom/DB's major competitor, cost in the neighborhood of $250,000. The graduated one-time charge for Release 2.0 of DB2[7] costs from $108,000 to $172,800, depending on the number of processors on the machine on which it is installed; monthly licence charges amount to $3,600 [IBM 288-196].

Additional costs must be added to the initial purchase price for these systems. Most large companies have complex, pre-existing data processing environments. The expense of "fitting" a new database system into that environment can amount to another 25 - 50% of the new system's list price. To this must be added the cost of technical support from the vendor, upgrades to new software releases and possible site license agreements. Fees such as those mentioned generally amount to an additional 15% of the original purchase price, annually. With prices like these, companies considering the acquisition of database software must carefully weigh all of the factors involved before making any purchasing decisions.

2. Overhead

The purchase price of software database management systems is only a part of the overall cost of such an implementation. Equally as important to consider is the cost incurred in establishing a mainframe environment - a major expense for a new installation.

The cost of mainframe processors is usually measured in terms of millions of instructions per second (MIPS). One MIPS on an IBM 3090-xxx[8] processor costs approximately $160,000; the 3081-xxx, closer to $300,000/MIPS [WEIN 87]. Given that most computer operation centers do not feel comfortable running their processors at more that 65% capacity, the effective price goes even higher.

DB2 is designed to run most efficiently on IBM's extended MVS operating system, MVS/XA. The cost to upgrade to XA in order to take full advantage of DB2's features may prove prohibitive for many installations. It is estimated [SCHI 87] that IBM's pricing policies, coupled with future enhancements slated for XA, will increase its cost to about 37% of a customer company's three year operating budget.[9] DB2 Release 2.0's incorporation into IBM's recently announced ESA architecture (see Chapter 4) will present further upgrade expenses.

---

6 A "standard" ADR Database system includes the following products: Datacom/DB, IDEAL, Datadictionary and Dataquery.

7 IBM's increasingly popular relational database management system. Release 2.0 under MVS/ESA will be available in late 1988 [IBM 288-196].

8 Denotes model numbers. For example, 400 or 600 in the case of the 3090.

9 Up from a 9% share in 1987.

C. Performance

The third problem to be mentioned here Is the performance of software databases; they have not met the performance requirements of a growing number of data processing centers. Rates in terms of the number of transactions processed per second are used to measure a system's performance. A small installation with less than 50 users for example, would probably be satisfied with rates of 30 transactions per second (tps) - each user would receive approximately one second response time. Larger shops however, with hundreds or even thousands of users demand performance far beyond these rates - in many cases beyond the capabilities of today's software database management systems.

Estimates for high volume, high transaction rate systems in the year 1990 show that such demands will exceed 1500 tps [BRAU 86c]. The 1990 goals set for IMS[10] are only 200 tps [INSI 85]. DB2 presently performs in a range (depending on the benchmark reviewed) from 5% to 33% of IMS's capacity [MART 86]. Recent benchmarks [INSI 85] taken for DB2 running on a 3081 processor under the MVS/XA operating system yielded rates of 16.6 tps - not all that impressive. However, another test run on an unencumbered 3090-200 yielded results of 48 tps [BABC 86b]. IBM claims to have achieved rates of 438 tps with Release 2.0 [IBM 288-196]. This is still far from the 1500 tps rate mentioned above.

D. Very Large Databases

Fourth, there are database applications today that have outgrown the processing capacity of the largest available mainframe configurations. Alternative solutions have been sought in an effort to avoid splitting these databases across two or more mainframes. Maintaining one central database is highly desirable for most installations. Also, the need to provide for continued growth of the database can be a major consideration. A system is required that will expand with the application, not limit its potential by placing restrictions on size. Such restrictions are an all too unfortunate reality with software database management systems.

E. Vulnerablilty

Finally, software database management systems are susceptible to what Teradata refers to as a "single point of failure" [DBC/ 86]. If the database management software were to abend[11] the entire database installation could be brought to a halt. Even with backout provisions to ensure data integrity, the failure of a high volume system can mean the loss of very large amounts of money.

IV. Database Machine Research.

The need to enhance the execution of database functions spawned a flurry of database machine research beginning in the early 1970's. While it is true that only a very few machines were actually produced and even fewer successfully marketed, it is nonetheless important to examine the technology as it emerged. So many theories have been proposed, prototypes derived and papers published, that it is beneficial to formulate some sort of classification scheme by which to organize the various approaches taken in this field.

---

10 IBM's "production" database management system.

11 An acronym for "abnormal end".

A. Hsiao's Classification Scheme

One of the most noted scientists in the study of database machines is David K. Hsiao [HSIA 83] who has proposed three different classifications (see Figure 7). The first of these is according to application; the use to which the machine is put. Under this heading come text search and retrieval database machines and formatted database machines.

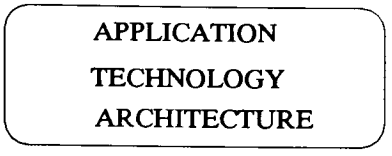

Figure 7. Hsiao's Classifications.

1. Application

Text search and retrieval machines apply pattern matching against stored text (i.e. character strings). This information store is typically "archival"; that is, it requires virtually no updating and by its very nature, tends to become extremely large. Such machines have value for libraries and information services, but lie outside the scope of this report.

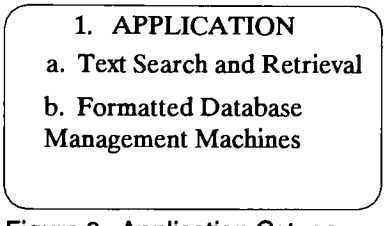

Figure 8. Application Category.

Formatted database management machines are on the other hand, the type of machine that has gained the most publicity in recent years. The data stored in these machines follow specific models and data formats. Predicates or "search criteria" are employed in searching the information store; facilities also exist for update, backup and often security.

2. Technology

Hsiao's second classification plan focuses on the technology utilized in the design of the machine. The first category here is that of the "Now" machines; those commercially available today. Most prominent among these is the aforementioned DBC/1012, manufactured by Teradata Corporation.

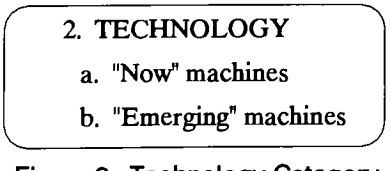

Figure 9. Technology Category.

Next are machines whose availablilty is contingent upon the maturation of certain software and hardware technologies. One such field is that of very large scale integrated (VLSI) circuits, an area

into which a great deal of research has been recently funneled. Another technological barrier to be overcome is in communication lines. Undoubtedly, fiber optics will play an important role here.

3. Architecture

Hsiao's third category is perhaps the least nebulous of all - that based in architecture. Four types of architectural approaches are detailed.

The first architectural classification is the Software Single-Backend (SSB) system comprised of one mainframe that is dedicated to database support. It may be connected to one or more other mainframes, but there is no hardware modification made to the computer performing the database functions. This method was used by Bell Laboratories in 1974.

The second configuration, Software Multiple-Backend, also employs no hardware modifications. It is essentially the same plan as that of the SSB, except that there is more than one backend machine, each executing the same database management software.

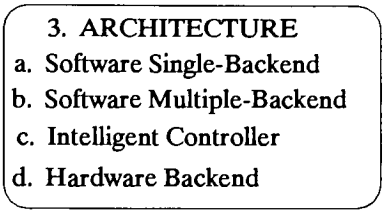

Figure 10. Architectural Category.

Third is the Intelligent Controller, a database machine equipped with customized (micro)processors, memory stores and lines of communication with which to coordinate the database functions that are requested by the host computer. A high-level data manipulation language is run on the host machine.

Finally, the Hardware-Backend design is what most truly characterizes database machines. Here database functions and input/output (I/O) routines are completely off-loaded from the host, processed entirely on a separate database machine which utilizes highly specialized software and hardware.

B. Qadah's Taxonomy

In 1985, Ghassan Z. Qadah [QADA 85] established a thorough, albeit confusing taxonomy of database machine research. In it, Qadah divides the field of study into what he calls "three-dimensional database machine space" (see Figure 11).

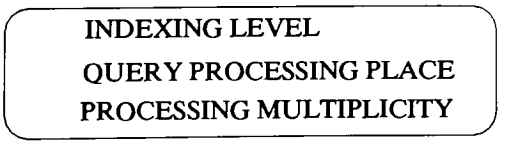

Figure 11. Qadah's Taxonomy.

The first dimension is the indexing level - that which is determined by the smallest addressable unit of data used as an index for making selections. The second is the query processing place. This shows where in the system the actual selection criteria are applied against the data. The third dimen-

sion is termed "processing multiplicity" and concerns itself with the number of operations performed and the amount of data processed simultaneously.

1. Indexing Level

There are three indexing levels in Qadah's scheme: database, relation and page. Theories of database machines that fall into the database indexing level category take the "associative search" approach. Here, the database itself acts as the smallest unit of addressable space; it is the entire database that is scanned. Any item of data found that satisfies some selection criteria is retrieved from the database for processing. While not the most efficient method, it does mark the first developed in the early 1970's.

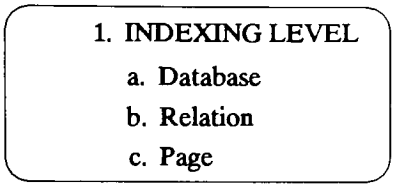

Figure 12. Indexing Level.

At the relation indexing level, the relation or table is the basis for indexing. This is often referred to as the "quasi-associative" approach. A mapping occurs from the selection criteria onto certain relevant relations. These relations[12] alone are scanned.

Finally, in the page indexing level category memory pages or blocks act as the smallest addressable units. This level is virtually the same as the relational-level above, except for the indexing unit.

2. Query Processing Place

Qadah names three different places where the processing of a query may occur: Off-Disk, On-Disk and Hybrid. The term "disk" in this instance refers to any unit of secondary storage on which the database resides. This includes not only fixed- and moving-head disks, but also electronic disks, such as charge coupled devices (CCD's) or magnetic bubble memory (MBM). According to the Off-Disk scheme, data units[13] are transferred from the disk to separate processors. These processors perform the queries and any necessary updates. Modified units of data are then returned to the database.

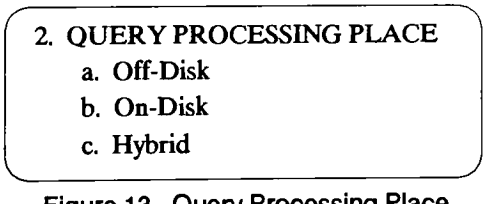

Figure 13. Query Processing Place.

---

12 A fairly small percentage of the entire database in most cases.

13 Whether the entire database, a relation, or a page, as discussed above.

Machines in the On-Disk classification are generally referred to as logic-per-track ("LPT") devices [SLOT 70]. The disk head is itself equipped with logic units and certain amounts of memory. Queries and updates are performed directly on the disk where the data are stored.

The Hybrid grouping consists of machines which handle query processing in a manner that is actually a combination of Off-Disk and On-Disk. Data unit selection takes place on the disk; those units selected however, are then transferred to a separate level of memory for further processing.

3. Processing Multiplicity

Three categories exist in this dimension of database machine space. They are: single operation stream-single data stream (SOSD), single operation stream-multiple data stream (SOMD) and multiple operation stream-multiple data stream (MOMD).

3. PROCESSING MULTIPLICITY
   a. Single Operation Stream-Single Data Stream (SOSD)
   b. Single Operation Stream-Multiple Data Stream (SOMD)
   c. Multiple Operation Stream-Multiple Data Stream (MOMD)

Figure 14. Processing Multiplicity.

SOSD devices perform a maximum of one database operation[14] on one record of the database at any given time. SOMD's on the other hand, are capable of executing one operation at a time, but on more than one record, simultaneously. MOMD machines may perform one or more operations on one or more records at the same time.

C. A Chronological Categorization

Viewing the progress of database machine research more logically lends itself to a chronological, historical survey. By following the developments in the field as they occurred in time, a clear delineation emerges, depicting the changes that were made in basic approaches to database machine theory, in general. A chronological classification scheme splits into two categories: pre-1979 and post-1979.

1. Pre-1979

All of the major contributions to database machine research from this period fall into Qadah's database indexing level classification. The very first machines proposed were Off-Disk, SOMD devices that made use of associative memory techniques [DEFI 73]. Next came the LPT machines, the earliest of which were SOMD's followed by MOMD designs. Finally, the Content Addressable File Store (CAFS) [BABB 79] arrived on the scene. This model was a Hybrid SOSD machine that virtually marked the end of the database indexing level genre of database machine prototypes.

---

14 Select, project, join or update.

2. Post-1979

RAP.2 [OZKA 85][15] was the first to utilize relational indexing level techniques. It falls into the Off-disk, MOMD classification. Research then made an almost exclusive turn toward Hybrid models, making use of both relational and page indexing levels.

V. DBC/1012 - The Teradata Database Machine

It is clear that database machine research has been prolific over the past fifteen years. But the number of machines that have actually been produced is minimal. As previously mentioned, the DBC/1012 is the only commercially available database machine capable of supporting very large databases in an MVS environment. The name of the machine is derived from its capacity - 10^{12} (1 trillion) bytes, or a "terabyte". Represented in terms of information on paper, this amount is the equivalent of 349 million typewritten pages.

The DBC/1012 can be classified in Hsiao's application category as a formatted database management machine; under technology as a "Now" machine; and as a hardware-backend architecture. According to Qadah's categorization, it employs indexing at the relational level; a hybrid query processing place; and a multiple operation stream-multiple data stream design.

A. An Alternative to Software Database Systems

The major features of the DBC/1012 offer solutions to many of the problems associated with present software database management systems.

1. Cost

The purchase price of the DBC/1012 starts in the neighborhood of $450,000 [PERR 85].[16] This offers an attractive alternative for some major installations. Citibank, N. A. for instance, has a DBC/1012 configuration with the capability of processing 168 MIPS[17] [KULL 86]. Requests are channeled from two IBM 3090 mainframe computers. The system cost $7 million; a similar installation utilizing software database management systems running on mainframes would have cost four times as much, according to Citibank.

The primary reason for the DBC/1012's relatively low price tag is its utilization of cost effective microprocessors. Release 3.0 can be equipped with up to 1,024 Intel 80286 chips. Each performs asynchronously, processing approximately 1 MIPS at a cost of $39,000 per MIPS [KULL 86] as opposed to the $160,000/MIPS on the IBM 3090 processor, mentioned above.

2. Performance

Teradata ran extensive performance benchmarks against the system installed at Citibank [TERA 86]. Among the four types of tests run was one designed to examine transaction processing rates. Brief-

---

15 An improvement over an earlier machine, RAP.1

16 This cost increases as the size of the system configuration expands.

17 Equivalent to processing 100,000 records per second.

ly, the results showed that transaction rates for typical banking transactions ranged from 46 tps with 40 processors[18] to 130 tps with a 128 processor configuration.[19]

3. Very Large Databases

The decision to install a database machine usually occurs when a database begins to outgrow the capacity of standard mainframe configurations. Organizations maintaining such databases are met with the realization that software database management systems often require hardware upgrades in order to support continued database growth without significant performance degradation. Such growth may eventually lead to splitting the database across two or more mainframes - an expensive proposal.

Incorporating the DBC/1012 into such an environment, with its terabyte capacity, offers a logical alternative. Very large and growing databases can continue to be supported without enhancing the mainframes or deviating from the central database approach.

4. Multiple Hosts

Inherent in the DBC/1012 design is the ability to support multiple host computers such as IBM 370 Models 148, 155, 168, 3032, 3033, 30xx, 43xx, as well as PCM's[20] by NASCO, Amdahl and IPL. The Host System Communication Interface[21] allows simultaneous access by not only more than one host, but more than one type of host as well. This feature facilitates the efforts being undertaken by many organizations to consolidate their databases presently located on different mainframes into one central database, with minimal impact on existing applications.

5. Vulnerability

Teradata feels that it has diminished the risk of database vulnerability by addressing the "single point of failure" issue plaguing software database management systems. An option may be enabled on the DBC/1012 which provides for the creation and subsequent maintenance of a duplicate copy of every record in the database. The data storage algorithm places these records on different disks from the original data. In addition to the redundant data store, the DBC/1012 is also equipped with two Ynets,[22] each acting as a backup for the other.

6. Other Features

There are other attributes of the DBC/1012 which appeal to prospective purchasers. First, its modular design allows additional processors to be readily added to the original configuration. This system can thus provide continued support for a database environment as it expands. Second, there is one and only one language - DBC/SQL - that is used for data definition, data manipulation, data control and queries. Conforming for the most part to the SQL standard, this language is easy to use and very powerful. Third, the user's view of the database is in tables, allowing a "relational", easily conceptualized model. Finally, both interactive and batch modes are supported by the DBC/1012, offering the users additional flexibilty.

---

18 8 IFP's and 32 AMP's - an "8 x 32" system. See description of IFP and AMP, below.

19 A "24 x 104" machine.

20 Plug-compatibles.

21 See details, below.

22 Teradata's intelligent communications bus. See details, below.

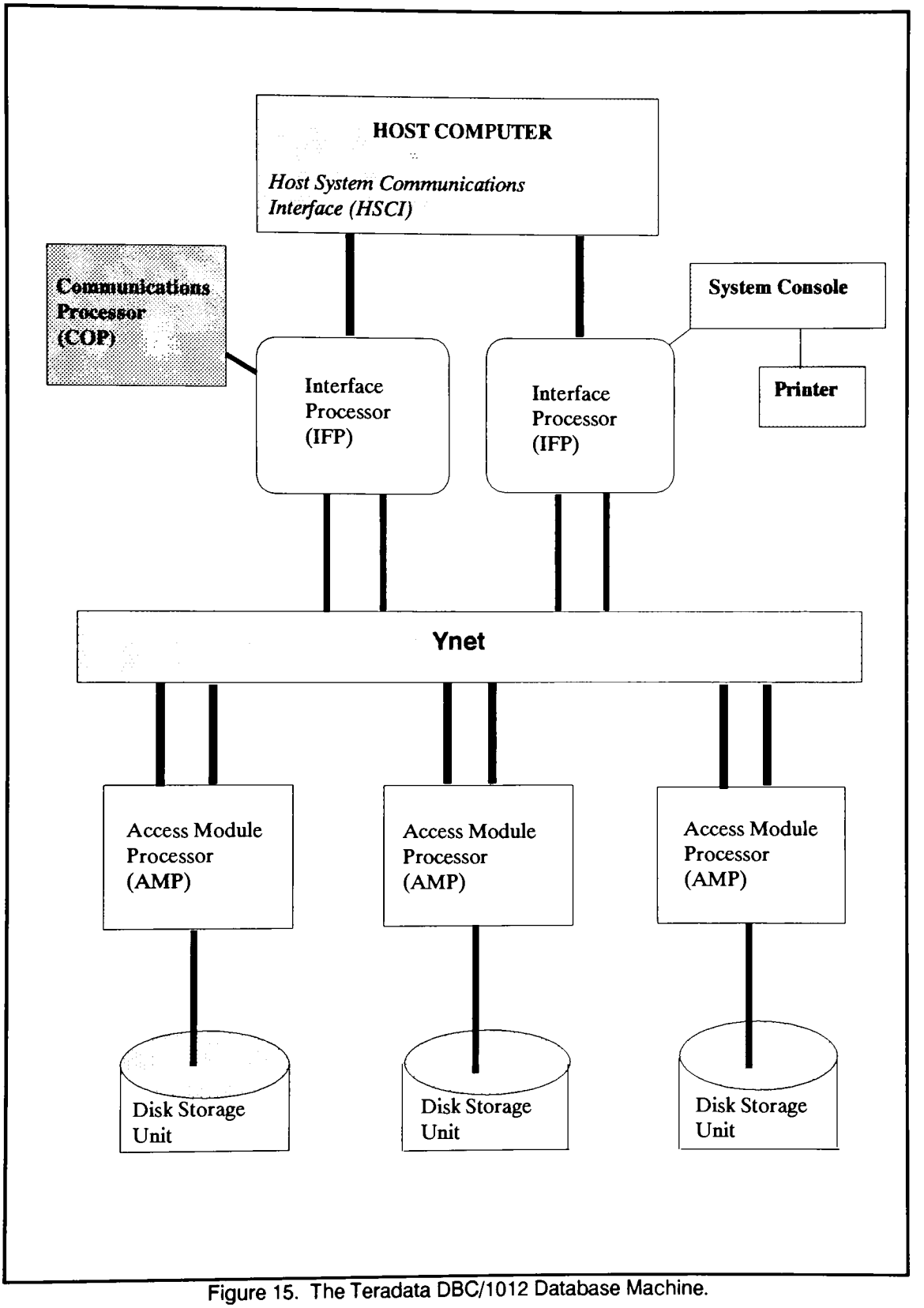

Figure 15. The Teradata DBC/1012 Database Machine.

B. Architectural Overview

A more in depth study of the architecture of the DBC/1012 is given in Chapter 3. Here, a brief overview of the system is appropriate.

1. Host System Communications Interface (HSCI)

The DBC/1012 system is comprised of seven subsystems (refer to Figure 15). The first of these is the HSCI. This is a software library consisting of an interface program and a number of service routines. Residing on the host computer, the HSCI provides the necessary connection between the DBC/1012 and the host. Moreover, the ability to function in interactive or batch modes is coordinated by this subsystem.

2. Interface Processor (IFP)

The IFP subsystem coordinates the flow of information between the host and the DBC/1012. Requests are received by the IFPs from the host and are then translated into internal DBC/1012 commands. These commands are forwarded to the other system components; responses from these components are processed by the IFP and the appropriate information returned to the host.

3. Ynet

Teradata's pride is the Ynet, a proprietary intelligent bus[23] that connects all of the IFPs and AMPs. The tasks required to manage the many processors in the system are performed by the Ynet hardware. As mentioned earlier, the DBC/1012 is equipped with two Ynets, each serving as backup for the other.

4. Access Module Processor (AMP)

This is the subsystem that performs the actual manipulation of data. Requests to the AMPs are routed from the IFPs by way of the Ynet. AMP responses are then returned over the Ynet, back to the IFPs. The AMPs operate in parallel; additions of AMPs linearly increase the performance of the system as a whole.

5. Disk Storage Unit (DSU)

The fifth subsystem consists of the actual data storage devices.[24] Initially configured with one DSU for every AMP, a second DSU may be added per AMP. Each DSU contains a portion of the total data store, with databases being evenly distributed over all the units.

6. System Console

The sixth subsystem is that of the system console which may have a printer attached as well. The console offers better control over the system to the computer operators. Information can be gathered regarding the status of the system and its performance; diagnostic operations can also be executed from here.

7. Communications Processor (COP)

The seventh and final subsystem is the newest for Teradata. Introduced in February 1987 [ZENG 87], the COP allows connectivity to the DBC/1012 directly from workstations, without going through

---

23 An array of "active logic".

24 Winchester-type disk units.

the mainframe. Teradata feels that this puts them in direct competition with IBM mainframes, and marks the beginning of a "mainframe-less" environment.

VI. Open Issues

In spite of the fact that database machines are beginning to attain acceptance in the industry and to demonstrate that they are a viable solution to a number of pressing problems, there are some issues that remain unresolved by the database machine alternative.

A. Referential Integrity

The first among these problems is the issue of referential integrity, an enigma residual of the software database management systems. Some are concerned that even Release 2 of DB2 cannot guarantee integrity in all situations [BABC 88a]. Until database systems, implemented through software or by database machines, resolve this incongruity to the satisfaction of their users, installations will have to rely on in-house, site-specific system programs to handle discrepancies which are pertinent to data integrity. While of little consequence to many smaller data centers, this is a very real and open issue for most large systems.

B. Conversion From Software Database Management Systems

The transition to database machine utilization from existing software database management systems is generally no simple task. Proponents of such a conversion may need to overcome the hurdle of management, convincing them that starting over with a database machine is in the best interests of company advancement. Change is not always readily accepted in some circles.

Once the decision to convert has been made, another barrier may exist. Presently used software database management systems may follow a non-relational scheme.[25] This would require massive efforts to completely reformat existing data to conform for instance, to the DBC/1012 model. Added to this is the program maintenance required to create an interface with the new machine. All of these obstacles may prove formidable enough that many companies would decide against conversion to a database machine.

C. Functionality

The real value of the database machine is exhibited in its ability to accommodate large databases, complex transactions and high transaction volumes. As has been seen for these large systems, a database machine may be the only answer. But many software database management systems have been available longer; their vendors have deleveloped very powerful end-user facilities, database administration aids and programming tools. This is not to the same extent true for today's database machines.

The issue then becomes - for installations who may wish to change to a database machine, but are not complelled into such an arrangement - one of functionality. Some users [KULL 86] give higher marks to software systems than to database machines for certain functional aspects, such as error detection and recovery, programming language interfaces and data dictionaries.

---

25 i.e. hierarchical or network.

In order for database machines to unequivocally compete with present database software systems, this discontinuity must be corrected. Developing facilities for database machines such as those available in software systems would be advantageous in achieving this goal.

D. IBM

Perhaps the single most important issue to be resolved before database machines gain full acceptance is the stance to be taken by IBM, the de facto maker of industry standards in the eyes of many. There are those who fear that IBM will embrace an approach incompatible with database machine technology, per se. Rather than run the risk of finding themselves on a deviant route, many computer centers have adopted a "wait-and-see" attitude, until IBM clarifies its policy. Unfortunately, given IBM's record of design announcements, this may take many years; longer than some organizations can reasonably be expected to wait.

VII. IBM's Response

As has been mentioned, IBM is considered by many to be the leader in the computer industry. It sometimes appears as if the company has a virtual teflon coating; impervious to directions taken by other vendors. If a standard is to be set, focus is on IBM to set it.

Two of the most popular software database management systems today, IMS and DB2, are IBM products. While seeming to promote its software database solutions to the exclusion of database hardware alternatives, IBM is first and foremost a hardware company [MART 84]. An analysis of its proposed hardware designs indicates that IBM intends to dedicate hardware to the support of database management functions. Evidence of this can already by seen in two of IBM's largest selling products - System/38[26] and the Sierra.[27]

A. System/38

System/38 ("S/38") is IBM's data processing system of the future [IBM 85]. It finds its place in the middle tier of IBM's three-tier architectural grand scheme. First available commercially in 1980, the S/38 has proven to be a great success. Its users are said to be IBM's happiest. By 1984, it earned that company's National Marketing Division, revenues of $10 billion, the largest share of any other product [ANDR 86].

Database management functions have been incorporated into the operating system - the Control Program Facility ("CPF") - as well as into the hardware design. The S/38 is so well equipped that attaining comparable functionality in a 4300 series computer would require a minimum installation of VM/CMS, DOS/VSE, CICS and SQL [ANDR 86].

Because of its power and ease of use, many organizations install the S/38 as their primary computer for all applications, including but not limited to database support. The current highend S/38 model is the 700, announced in June 1986 [IBM 186-103]. This machine makes use of a new one million bit dynamic random access memory chip (DRAM) which provides up to 32 megabytes of internal memory. The internal speed of the new Model 700 is reported to be 390% faster than the old Model

---

26 IBM's "department-sized" computer.

27 The 3090 processor mainframe environment. This line will soon be replaced by the Summit computers.

4 [IBM 186-104]. There is no doubt that IBM has great faith in S/38 and that it will be a major player in future IBM systems.

B. Mainframe Trends

But what of IBM's plans for mainframe support of database machine functions? Since very large databases are increasingly prevalent and their numbers continue to grow, these are the applications toward which IBM will likely target major efforts in the next few years. S/38 will probably fill the database machine niche in smaller installations; IBM has other plans for the mainframe environment.

1. Intelligent Disk Controller

Rumors developed in November 1986 that IBM would soon release its Intelligent Disk Controller [BABC 86a]; contrary to the company's announcements. Indeed, the 3990 Controller was announced in 1987 for release in 1988. In some ways, this unit can be seen to conform to Hsiao's third architectural database machine classification in that it will manage the databases' data store on direct access storage devices. The host machine will run the software database managment system, but it will virtually be freed from the overhead of I/O associated with database search, retrieval and updating. These functions will be performed exclusively by the Controller, independent of the host.

That the Intelligent Controller is to be incorporated into IBM's overall plan is evidenced by the fact that certain products have been designed in a manner conducive with the Controller's implementation.

a. *Data Facility Product*

The Data Facility Product ("DFP") controls disk storage for MVS/XA. Media Manager, the access method portion of DFP, allows users to request data using logical addresses,[28] as opposed to physical addresses.[29] This is the same access method utilized by IMS Fast Path Version 2 and by the Intelligent Disk Controller; indeed, Media Manager is capable of supporting such a device [BABC 86a].

b. DB2

It is also anticipated that DB2 is slated for support by the 3990 Controller [BABC 86a, MORA 87]. Presently, DB2's search argument, SARG, runs on the mainframe central processing unit (CPU). This could easily be moved off onto the Controller, thus freeing the main CPU from this type of function. Utilizing Media Manager and the Controller in support of DB2 would greatly increase that product's performance capabilities.

c. Data Facility/Hierarchical Storage Management

The Data Facility/Hierarchical Storage Management ("DF/HSM") component is another feature of the MVS operating system that has already been separated out, making it a candidate for Controller support as well. A sectioning of the I/O function of MVS, DF/HSM creates an interface between these functions and the other parts of the system. Some predict [FERT 86] that DF/HSM will eventually be integrated with DB2, allowing that product to manage very high volume transactions. IBM would then, it is thought, be in a position to introduce a full-fledged database machine [FERT 86].

---

28 Called "logical record processing".

29 Or "physical record processing".

2. Summit

IBM's Sierra mainframe line will soon give way to the next generation - Summit [MORA 87]. Providing support for high performance, intensive I/O processing installations, the Summit will make extensive use of microcode and parallel processing techniques. Three categories of processors will perform different functions: a general purpose processor group to support applications, a group that will perform storage management, and finally a group to handle communications and transaction processing.

The feature that is anticipated to make the Summit unique is the concept of "cluster processing". Here, groups of four to sixteen processors will be linked together by a common systems manager[30] and will have access to a global main memory store. Such an approach will require a complete revamping of MVS to support the partitioning of operations among the clustered processors.[31]

Operation partitioning will allow separate processors to support for example, a DB2 master file as well as the DFP. The 3990 Controller will undoubtedly be vital to the performance of the overall system. Although IBM is not naming any part of this design a "database machine" per se, that is essentially the role that will be played by the processors, separated out in this fashion.

VIII. Summary

This chapter has been concerned with the evolution of database machines out of software database technology. A brief description of the early years of database software illustrates the reasons why this type of solution was pursued. While a complete account of this subject lies outside the scope of this document, it is nonetheless important to an investigation of the initial phases of work done in the database machine field.

Problems have resulted from software database solutions. Some of these amount to needs not having been met by software database management systems in general. Among these is support for very large databases, an area where software database management systems have been found deficient. Other problems appear to be inherent in the software solutions to database management. The most important of these is referential integrity, a dilemma that desparately needs to be addressed.

Because there has been a great deal of research done in the field of database machines, classification schemes help to organize the research into a more intuitive formula. Three such organizational approaches were discussed; Hsiao's classifications by application, technology, and architecture; Qadah's three-dimensional database machine space - indexing level, query processing place and processing multiplicity; and finally the chronological categorization showing the pre-1979 and post-1979 research.

The Teradata DBC/1012 database machine was introduced with a brief description of its seven subsystems. The main focus of this work will be on the DBC/1012, to the exclusion of smaller, department-sized machines, such as Britton-Lee's IDM 500. The solutions offered by Teradata in response to the problems of software database managment systems were briefly discussed.

---

30 Possibly implemented in DB2, supported by the 3990 Controller.

31 Part of a project code-named "Jupiter", the rewriting of MVS is underway [MORA 87]. MVS/ESA is the first announcement of this project [IBM 288-059].

There are numerous open issues that need to be addressed; problems either not solved by database machines, or else created by their implementation. As an example of the former, referential integrity still plagues database installations. A problem inherent in database machine implementation is the task of converting from software to hardware database management solutions.

Finally, there is a great deal of concern in the data processing world about IBM's response to the issue of database machines. It does appear that IBM has targeted the System/38, a database machine, to the second tier of its three-tier architectural scheme. The uncertainty lies in IBM's mainframe environments. While the company has announced no plans to implement database machines in the mainframe tier, it is evident that increasing amounts of hardware support is slated for database management functions in the Sierra and In the new Summit lines.

In the next chapter, the major contributions to the field of database machine research will be discussed in detail. While eluding to other classifications schemes, research will be viewed in a chronological order, lending a better understanding of and appreciation for the progress made, as it was made. Numerous approaches to database machine design will be reviewed and their impact on the science as a whole discussed.

The third chapter will focus exclusively on the Teradata Corporation and its DBC/1012 database machine. The seven subsystems introduced in this chapter will be thoroughly examined and illustrated. Also included will be an in depth analysis of the results of four performance benchmark projects conducted against the DBC/1012.

The fourth chapter will look at IBM's plans to incorporate more and more hardware into its systems, resulting in what may be construed as a de facto database machine implementation. Two main areas will be focused on, the System/38 and the mainframe environment. Concerning the former, a detailed discussion of the S/38 will illustrate that it is indeed IBM's database machine today. The System/38 and System/36 hybrid machine developed under a project code-named Silverlake will also be reviewed. In the area of mainframes, the Sierra line with its latest enhancements will be discussed, as well as proposals made for the next series, Summit.

The final chapter will be an analysis of where database machine technology is leading. New vistas for database machine implementation will be reviewed, as well as new emerging technologies whose impact on the field of database machines in general warrants discussion. An in depth look at the future of the Teradata database machine will also be made.

Chapter 2

A History of Database Machine Research

I. Introduction

This chapter follows the history of database machine research. While making reference to the classification schemes established by Qadah [QADA 85] and Hsiao [HSIA 83], the various database machine proposals discussed here will be reviewed chronologically, in order of their publication in major scientific journals.

The volume of research published over the last fifteen years is so great that representative models had to be chosen for purposes of this discussion. An attempt has been made to exemplify the various combinations in Qadah's taxonomies of query processing place, indexing level and processing multiplicity. This scheme gives us a sampling of the most important contributions to the field of database machine research. The examples split between Hsiao's intelligent controller and hardware backend architectural categories. They fall for the most part however, into the formatted database application category and the "Now" machine technology classification, but that is in keeping with the focus of this research.

Machine Design	Classification						
	I.L.	Q.P.P	P. M.	Appl.	Tech.	Arch.	Chron.
IFAM	0	0	1	1	0	2	0
CASSM	0	1	1	1	0	2	0
RAP.1	0	1	1	1	0	3	0
CAFS	0	2	0	1	0	3	0
SURE	0	1	2	1	0	2	0
RAP.2	1	0	2	1	0	2	1
DIRECT	1	0	2	1	0	3	1
DBC	2	2	1	1	0	3	1
DIALOG	1	2	2	1	0	3	1
RAP.3	1	0	2	1	0	3	1

I. L. (Indexing Level)
0 - Database
1 - Relation
2 - Page

Q. P. P. (Query Processing Place)
0 - Off-disk
1 - On-disk
2 - Hybrid

P. M. (Processing Multiplicity)
0 - Single Operation Stream-Single Data Stream (SOSD)
1 - Single Operation Stream-Multiple Data Stream (SOMD)
2 - Multiple Operation Stream-Multiple Data Stream (MOMD)

Appl. (Application)
0 - Text Search and Retrieval
1 Formatted Database

Tech. (Technology)
0 - "Now"
1 - "Emerging"

Arch. (Architecture)
0 - Software Single Backend (SSB)
1 - Software Multiple Backend (SMB)
2 - Intelligent Controller
3 - Hardware Backend

Chron. (Chronological)
0 - Pre-1979
1 - Post-1979

Figure 1. Database Machine Categories.

Still the chronological categorization serves to illustrate two critical elements - how research was influenced both by previous endeavors and by advances in computer technology. It is in chronological order therefore, that the examples discussed here will be presented. See Figure 1 for a list of the designs to be reviewed and the corresponding categories to which they belong.

Three important topics need to be discussed before examining the early stages of database machine research. The first concerns the use of magnetic disks as secondary memory devices; the second, the concept of associative or content-addressable memory; and third, Slotnicks's proposal to associate search logic with head-per-track disk devices [SLOT 70].

A. Magnetic Disk Memories

In the early 1970's, direct access magnetic memory devices were fast becoming the preferred medium for secondary storage of data, replacing magnetic drums and core extensions.[1] Like drum memory, these devices made use of magnetic properties for storing information. However, they soon proved to be faster and more economical than the drum units.

The disk devices resemble a stack of plates rotating around a common spindle or axis (see Figure 2). Each plate, called a "disk", is coated with a magnetic surface and rotates under a fixed coil, called a "head", which in turn is supported by an "arm". Each arm is a component of an assembly of arms called a "comb". Located on each arm is a set of read/write heads, by means of which magnetic patterns are recorded onto the disk surface ("writing"), or sensed from that surface ("reading"), as the disk rotates through. Information is stored on each disk in sections called "tracks". There are two different types of magnetic disk devices: fixed-head-disks and movable-arm-disks.$&Figure 2[v]

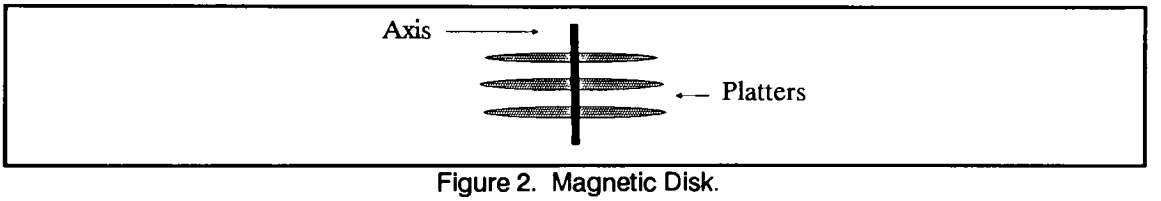

Figure 2. Magnetic Disk.

1. Fixed-Head-Disks

Also called a "head-per-track" device, here the arm/comb assembly does not move - it is "fixed" (see Figure 3). Each arm contains several heads (one for every track) which read from or write to the various tracks on the disk. The time required to retrieve data using such a device is calculated by adding together the rotational speed of the disk and the transfer time to the heads.

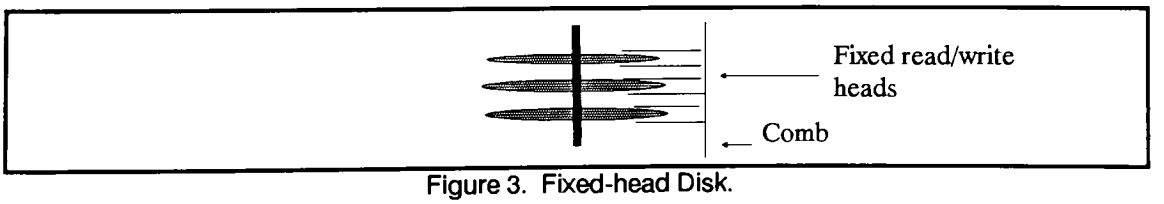

Figure 3. Fixed-head Disk.

1 See [BAER 80] for a more complete discussion.

2. Movable-Arm-Disks

The arm on such a device is not stationary, but rather moves in and out along the radius of the disk assembly. All of the heads move together and are aligned along the same track on the different disks. Tracks thus aligned are termed a "cylinder". Retrieval time for a movable-arm-disk device includes the same parameters as for the fixed-head-disk, plus the time required for arm movement called "seek time".

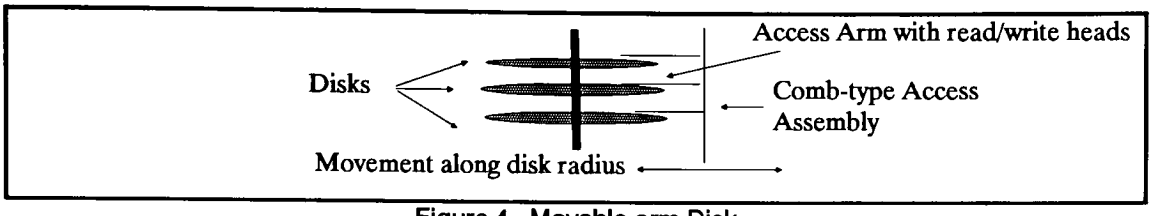

Figure 4. Movable-arm Disk.

B. Associative Memory

The concept of associative memory is important to the study of database machine research because most of the major contributions to the field utilized associative memory search techniques for locating data in a database. By way of such techniques, data items are addressed by content, not by location.[2]

Briefly, an associative memory makes a determination that answers the question "is there a memory location which contains data item 123?", instead of answering "what is the content at memory location 0945E?". In order to make such a determination, all memory locations must be searched for a match on the item in question (i.e. the "search criterion"). The methods by which the various database machine designs handled this search of all of the memory locations provide a foundation for studying the evolution of database machine research.

C. Slotnick

D. L. Slotnick [SLOT 70] of the University of Illinois made a proposal in 1970 which sparked an interest in associative memory searching techniques and soon provided the foundation on which database machine research developed. Slotnick's design involved head-per-track rotating disk devices and small, solid-state signal switches.

Factors contributing to Slotnick's proposal are a reflection of the state of computer technology from 1965 to 1970. The cost of signal switches had decreased tenfold in that time; their reliability had increased by a like factor. Head-per-track disk devices had become state-of-the-art and memory speeds had increased greatly. Yet applications which made use of large amounts of storage experienced poor performance due to the frequency of data tranfers from secondary storage.

By associating one hundred to several thousand switches for each track on a disk, data read from those tracks could be processed "on-the-fly" (i.e. as it passed under the read head). The focal point of the search mechanism was a processing unit comprised of thirty to two hundred logic gates lending a complete set of arithmetic and Boolean operations to the circuitry. The contents of an entire

---

2 Associative memory is also referred to as content-addressable memory.

disk could theoretically be scanned in one revolution of the disk. Such a system would most obviously be put to use as a large associative memory in a file-oriented application. Indeed, it was the database researchers who worked this new concept into their design for hardware support of database functions.

II. Database Machine Research

Slotnick's concept introduced a viable approach to content-addressable memory and marked the beginning of a great deal of database machine research. The following examples, taken in chronological order, depict the major contributions to the field.

A. Information Systems for Associative Memories (IFAM).

DeFiore and Berra in 1973 [DEFI 73] introduced a design which is representative of Qadah's off-disk, database indexing level category. Machines of this type have a general architecture such as that shown in Figure 5. A fixed-head-disk stores the database. The associative memory is loaded with data from the disk via a data channel; this same channel returns data from the associative memory to the disk after processing has taken place.

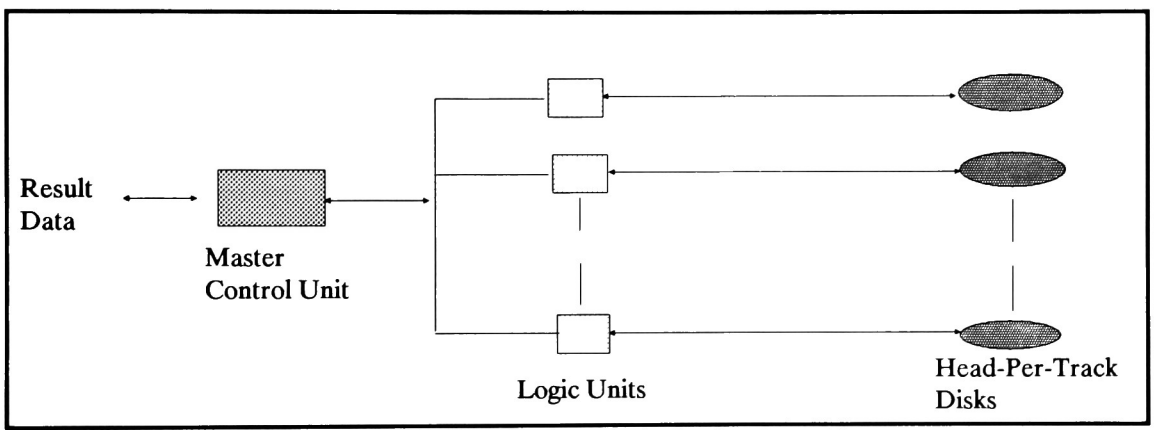

Figure 5. Off-disk, database indexing level machine.

The processing units themselves were simple hardwired logic units paired with sections of the associative memory called "words". These logic units were directed in their operations by a control unit. They performed in parallel, searching the contents of their assigned words for matches on a given operand.

IFAM provides a specific example of such an implementation. Its design is illustrated in Figure 6. The associative memory was developed by Goodyear and was comprised of 2,048 words of six bytes each. The logic units associated with this memory had no arithmetic capabilities, but could perform eleven basic search operations including equality, inequality, maximum and minimum.

The database contents were transferred for processing to the associative memory from the main computer (a CDC model 1604, second generation computer) via a Direct Memory Access channel

(DMA). The rate of transfer across this channel was about twelve microseconds[3] per word. Information was passed in one word increments. Once the associative memory was loaded with data, the search logic units operating in parallel could process the entire contents in about seventy microseconds, identifying those tuples which satisfied the search criteria.

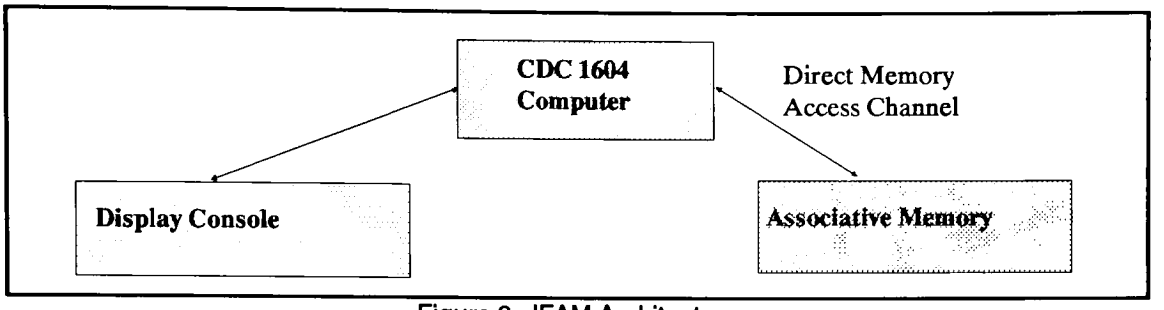

Figure 6. IFAM Architecture.

The designers of the IFAM system cited the elimination of indexing as the primary advantage of their model. Indices, they contended, were merely unnecessary repetitions of the data in the database requiring excessive amounts of additional storage and slowing query processing by a factor of ten. Since indices have to be updated whenever the data is updated, the researchers also claimed that response time decreased proportionally to the number of items indexed.

There were two main disadvantages in the off-disk DB model in general and in the IFAM prototype in particular. First, such systems were very expensive, especially those designed to support very large databases. This high cost had two sources - the incorporation of search logic units into the associative memory and the use of fixed-head-disks. Movable-arm-disks quickly proved to be a more economically feasible secondary storage device. The storage capacity of moveable-arm-disks approached an order of magnitude higher than that of the fixed-head-disks, making the latter a virtually obsolete technology.

The second major disadvantage of the IFAM-like designs was the bottleneck which occurred during the loading and unloading of the associative memory. While processing of the data once inside this memory was relatively fast, the transfer of data from the slowly rotating fixed-head-disk across the DMA channnel caused the effective overall processing time to be reduced to the speed of the disk. It soon became apparent that moving the contents of the data store off of the disk and into a separate associative memory for processing was not a feasible implementation for a cost-effective database machine.

B. Content Addressed Segment Sequential Memory (CASSM).

A different design was presented in 1973 by Copeland, Lipovski and Su [COPE 73]. It introduced a genre of database machines that would become known as "Logic-Per-Track" (LPT) devices. There are some similarities between CASSM and IFAM. Both fall into Qadah's categories of database indexing level and SOMD processing multiplicity. Also, they both utilized fixed-head-disk devices and a control unit to coordinate the logic units. But there the similarities end.

---

3 A microsecond is one millionth of a second.

Instead of moving data from the disk memory to the associative memory for processing as in the IFAM model, CASSM's design provided each head of the disk assembly with its own logic unit, allowing the data to be processed on-the-fly. Figure 7 shows the design of CASSM. Only those tuples satisfying the search criteria, which criteria were stored in registers in the logic units, were output for further processing. This was unlike IFAM where the entire database contents were moved, then processed.

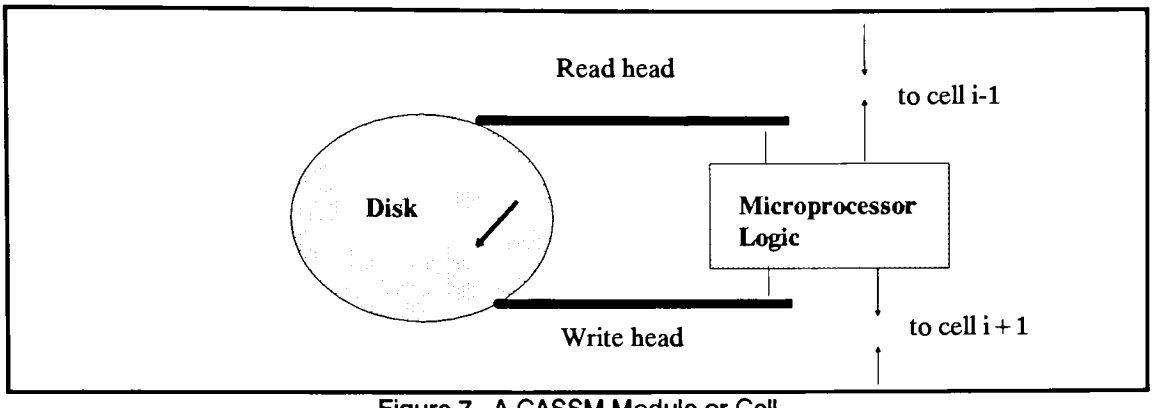

Figure 7. A CASSM Module or Cell.

To better understand the processing logic of CASSM, it is helpful to look at a representation of a database record in storage. Rows of a relation such as that shown in Figure 8, were stored in variable length blocks along a disk track. These blocks were comprised of sequences of 40 bit words (see Figure 9). The beginning of each record was marked by a one word record delimiter, thirty-two bits of which contained the relation name and certain control information. Following each record's delimiter on the block were more words that contained the actual record data. Thirty-two bits of these words contained column/value pairs (i.e. a column or field name followed by the value for that field in the relation). The remaining eight bits in each word functioned as "tag" bits to identify the contents of the word and as "mark bits" used for internal processing.

SUPPLIER RELATION

Supplier #	Company Name
01	Schubert
02	Nellis

Figure 8. CASSM database relation.

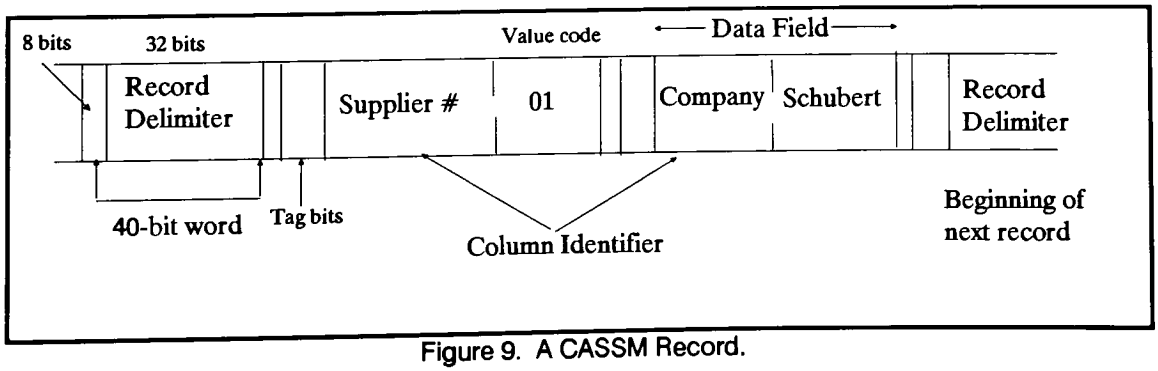

Figure 9. A CASSM Record.

The most important feature of CASSM was its provision for record selection at the disk level. This was accomplished by applying the principles of parallel processing to the database search mechanisms. Conceptually, the database was divided into "x" number of sections. The overall task of searching the entire database was then broken down into "x" concurrent search tasks, conducted by "x" physically identical modules. Each of these modules processed 1/x of the database. A basic concept in CASSM's design was that more search modules could readily be added to the system as the database grew in size, instead of assigning more data to be processed by existing modules.

CASSM's search modules were comprised of two components: an Instruction Control and Rewrite processor (IRC) and a Random Access Memory (RAM). This RAM was small, containing one bit for every record to be processed by the search module. It was used for marking tuples that met the selection criteria.

The other element involved in the search procedure was the data portion of the record delimiter which, as mentioned above, preceded each tuple as it was stored along the track (refer to Figure 9). This delimiter stored the name of the relation to which the tuple belonged, a six bit stack called a "B-stack" and two one-bit fields, the "S" and "Q" bits. A query example is useful in illustrating the interaction of the various components involved in a CASSM search operation.

Let us use the following query:

- "SELECT FROM RELATION SUPPLIER ALL ROWS WHERE CITY = DALLAS AND PART = BOLTS".

First, a "context" search is performed. During the first disk revolution, the delimiter word of each tuple is searched for the relation name "SUPPLIER". The S bit of each tuple containing that name is set, indicating that this tuple should be processed further. On the second revolution, the Q bit of these same delimiter words are set (both the S and Q bits are set). The column/value pair words of each record marked in this second revolution are searched during a third revolution to locate those words with the contents DALLAS. This denotes the beginning of the "content" phase of the search process.

During the content phase of the search, if a match is found, a mark (i.e. a "1") is pushed onto the B-stack of the delimiter word for that tuple. A fourth revolution provides a search against these same records for a column/value pair word PART,BOLTS. If a match is found here, a 1 is logically ORed with the contents of the top of the B-stack for that tuple. The results of this operation will be "1" if a match was found on both column/value words. If this result is 1, the RAM bit corresponding to the tuple's sequential number is set (i.e. if the fourteenth tuple has been selected, the fourteenth RAM bit is set). The final state of the RAM determines which tuples met the selection criteria. These tuples would then be output to the user.

CASSM offered many advantages over the off-disk DB designs like IFAM. It had the advantage of faster selection of data, as well as the processing of data at the disk level. Only data that resulted from the context search phase was moved off of the disk. Also, parallel processing was found to offer certain advantages to database processing. First, all of the search modules were identical, allowing for cost reductions attributable to quantity. Second, the software required to control the search modules was the same for all of the modules, regardless of their number. Third, the design costs for both hardware and software components of the system were independent of the size of the database supported. Finally, since each of the x modules searched 1/x of the database, the time to search the entire data store was likewise independent of the database size.

Still there were several major disadvantages common to LPT devices in general. First, as in IFAM, the cost of fixed-head-disks was prohibitively high. Moreover, the search logic associated with the

disk was an additional expense. Even though more search modules could theoretically be added to CASSM to support larger and larger databases, as a practical matter this design was not cost-effective for large data stores. Second, as was demonstrated in the previous example, numerous disk revolutions were required to process a fairly simple query. Processing more complex operations was not feasible due to the number of revolutions required.

C. Relational Associative Processor - RAP.1

A third significant contribution to database machine research was made in 1975 by Ozkarahan, Schuster and Smith [OZKA 75] of the University of Toronto. Later redesigned twice more, the original model became known as RAP.1 and was representative of LPT devices in Qadah's database indexing level (on-disk DB) category.

The primary difference in the RAP.1 model over those previously mentioned is that it was designed to function autonomously as a "backend" processor, separate from a host or general purpose computer ("GPC"). See Figure 10 for a description of the RAP.1 architecture. The GPC performed multiple functions in the RAP.1 system, not the least of which was to provide a communications interface for the user. Queries were received by the GPC, compiled into RAP instructions (called "primitives") and transferred to RAP's control. Scheduling of query execution was the responsibilty of the GPC, as were system paging and database security and integrity.

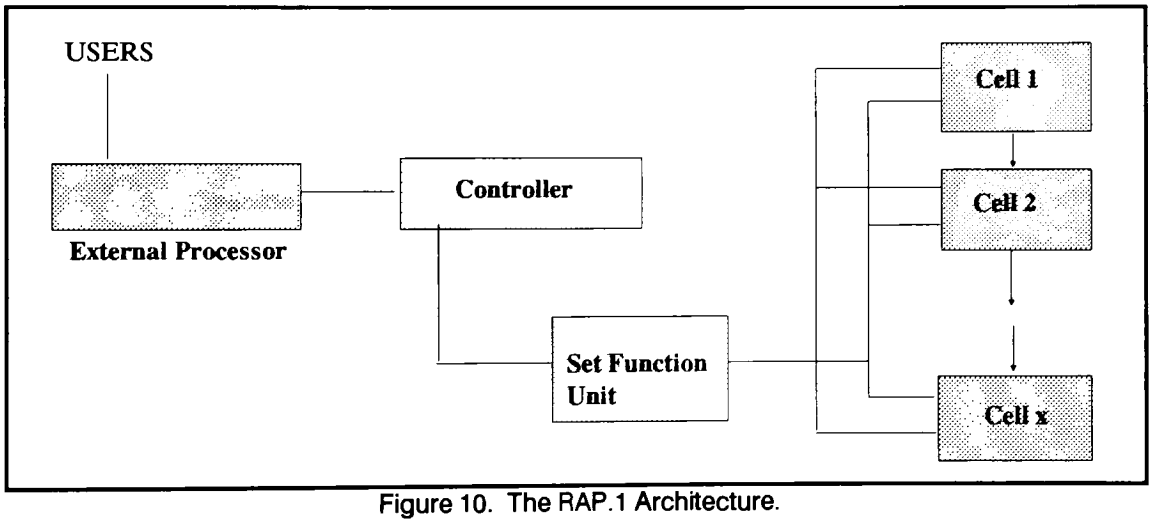

Figure 10. The RAP.1 Architecture.

The RAP.1 system itself was based upon the concept of cellular architecture. Each cell was equipped with a special purpose hardwired microprocessor containing processing logic. This logic was distributed over a large capacity sequential circulating memory, such as a track of a disk. This is unlike CASSM with its very small associated RAM. The other components of the RAP.1 design were the Controller and the Set Function Unit (the SFU).

The Controller was responsible for the overall coordination of activity within RAP.1. Primitives from the GPC were received by the Controller which in turn sent control sequences or "initiation signals" to the cells, instructing them to execute a specified database operation in parallel. The Controller was enhanced by the SFU which computed a summary of all of the cells' results, obtaining an overall result for the total memory contents.

A RAP.1 cell's structure is shown in Figure 11. Each cell was positioned on a serial communications path with its successor and predecessor cells. They also had I/O signal connections with the Con-

troller and SFU. The Information Search and Manipulation Unit (ISMU) decoded the commands sent to the cell by the Controller and evaluated the given search criteria. It was also responsible for inter-cell communications.

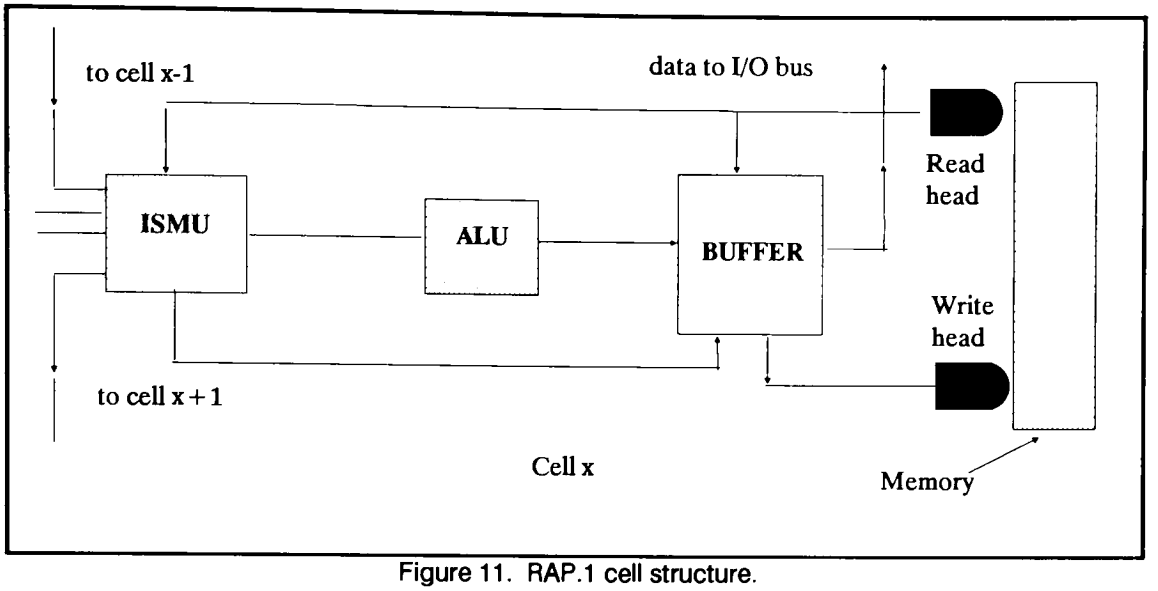

Figure 11. RAP.1 cell structure.

Read and write mechanisms on fixed-head-disks were associated with each cell so that data could be processed on-the-fly as memory rotated under the heads. One complete revolution of the memory store was thus required to read the contents in its entirety.

As data was read from the track by the read mechanism, it was passed to a 1,024 bit buffer where it was processed by the cell logic. Here selection, updates and arithmetic operations took place on individual tuples. The arithmetic and logic unit (ALU) contained a serial adder, a multiplier and logic for intermediate set function calculations (e.g. maximum and minimum). It was controlled by the ISMU. Intermediate results from each cell's processing activities were held for possible later use by the SFU in calculating overall results.

RAP.1's data representation scheme is shown in Figure 12. Information was stored in blocks on the tracks. Tuples of a relation were stored one per block, with tuples of only one relation allowed on any track. Each track began with a Track Marker block indicating the beginning of the track. This was followed by a "header" block containing the name of the relation being stored on that track. A second header block held the column names for the relation's tuples, in order as they appeared in the tuple. The remaining blocks each contained the field values for every tuple of the relation, stored in order by the column names in the second header. A string of "mark bits" preceded each block of concatenated tuple values indicating the length of those values. Gaps occurred on the tracks between blocks. The maximum tuple length for RAP.1 was 1,024 bits, dictated by the size of the cell buffer.

Search criteria were initialized at the GPC and transmitted simultaneously to each cell by the Controller. As the disk rotated, tuples were read one at a time from the tracks associated with each cell. Then they were placed in the buffers where they were processed by the cell logic on-the-fly. Tuples meeting the selection criteria were immediately marked for output. The entire memory store was processed for each instruction.

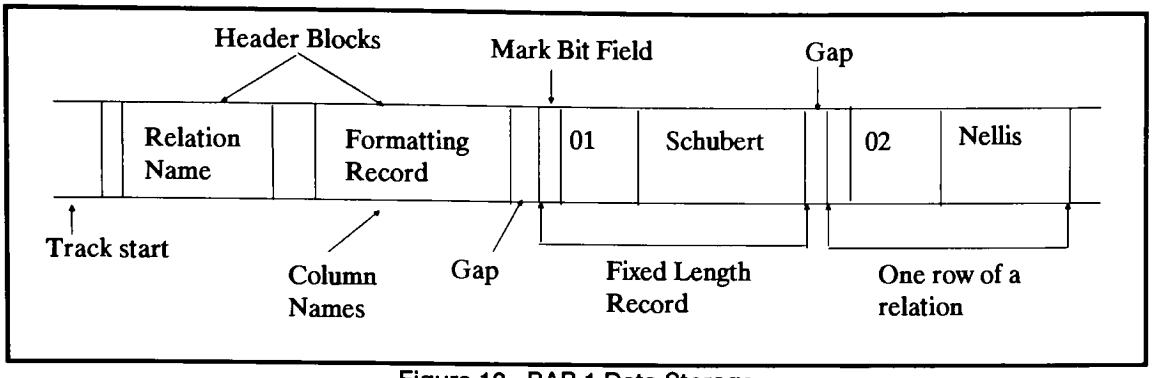

Figure 12. RAP.1 Data Storage.

There were two very serious disadvantages in the RAP.1 design, the first of which was just mentioned. Since queries are often complex, they would be translated into a number of RAP instructions. Unlike CASSM, where a context search was performed to indicate which tuples should be processed further, RAP.1 processed each tuple of the database for each primitive. Thus, while response time was deemed independent of database size due to the parallelism of the cellular structure, it was very much dependent on query complexity.

Second, RAP.1 was designed with a single I/O channel used by all of the cells in the system to output selected tuples. Read out contention posed a serious bottleneck when more than one tuple was selected for output, an occurrence of some considerable regularity.

D. Rotating Associative Relational Store - RARES.

Lin, Smith and Smith [LIN 76] published the specifications of RARES in 1976. This is another example of a logic-per-track device. The RARES design was unique however, in that it used an "orthogonal" storage layout technique.

Both CASSM and RAP.1 stored tuples in blocks along the tracks of fixed-head-disk devices, search modules being associated with each head, one head per track. Tuples were processed in parallel as the disk rotated. An output bottleneck occurred since selected tuples could only be received sequentially by the output channel. Such tuples may have been selected concurrently however, by the search logic units. To address this problem, RARES stored tuples across the tracks of a disk (i.e. along the radius of the disk), instead of along them.

The first byte of a tuple's first field value was stored on a track in a one byte "cell". The second byte of the tuple was placed on the next adjacent track, in the same position, and so on (see Figure 13). This layout was in "byte-parallel" order. The radial sequence of cells was known as a "sector". The set of tracks used to store a relation was called a "band". According to the RARES model, a band could be comprised of no more that sixty four tracks (the number of tracks was called the "bandwidth"). Some relations would have enough tuples such that the use of more than one sector was required to store them; these sectors became "rows" of the band. Notice in Figure 13 that there are unused cells in the last row of each tuple. The bandwidth for the SUPPLIER relation could be reduced to minimize this wasted space.[4]

---

4 If a field value were split across two rows however, matches on that field required two disk revolutions.

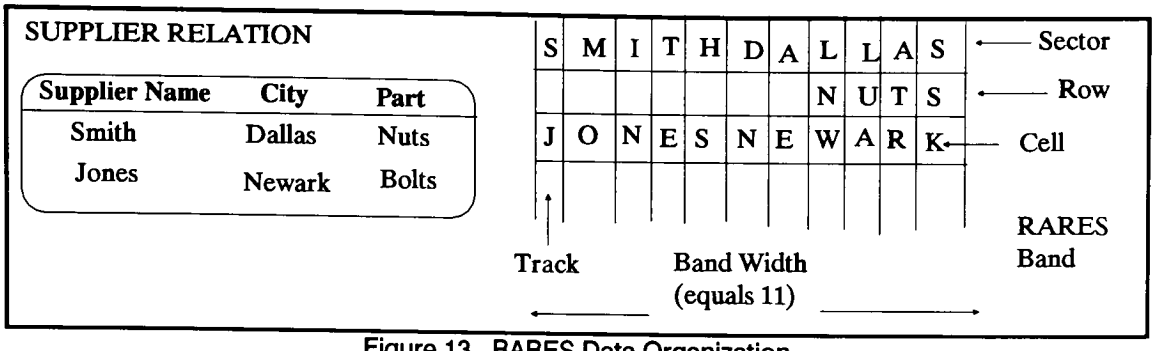

Figure 13. RARES Data Organization.

Several components interacted in the RARES system to form its "Content-Addressing Mechanism", the method by which tuples meeting some search criteria were selected for output. RARES was equipped with a number of search modules, each of which was assigned to a set of logically adjacent tracks (see Figure 14).

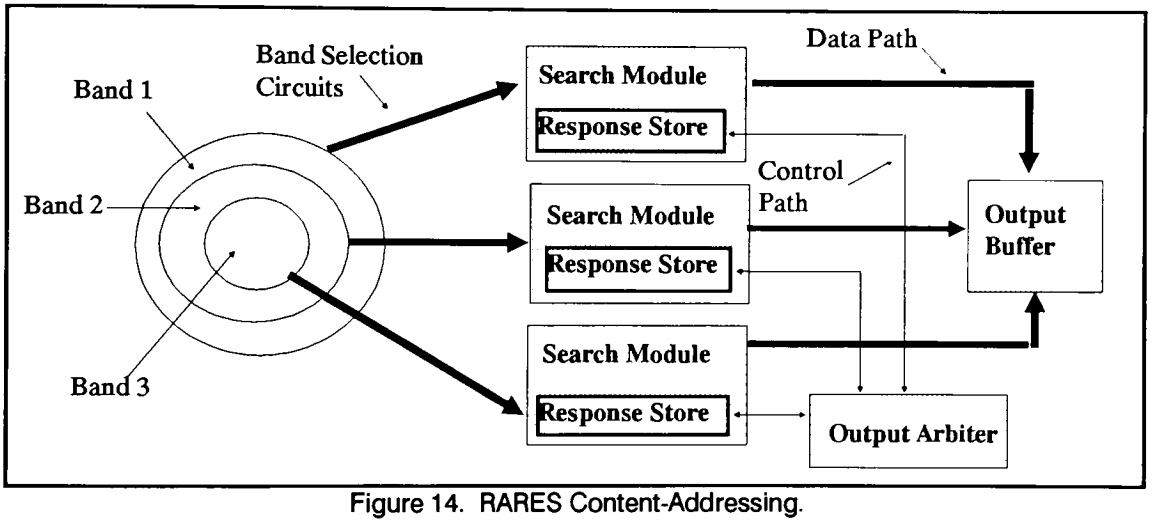

Figure 14. RARES Content-Addressing.

Within its assignment of tracks, each search module could process any band located on those tracks, one band at a time. For example, if a search module were assigned to 253 tracks, it could process twenty three different bands of width eleven, one at a time. Track assignments of the search modules were allowed to overlap. A control program determined which band the search module would process at any given time. Where CASSM and RAP.1 required a search module for each track of the disk, RARES only required on search module per band.

The other components included a response store, an output arbiter and an output buffer (see Figure 14). The response store was a small, fast access memory store associated with each search module's current band. It contained mark bits for each tuple within that band. The output arbiter allowed a search module to transfer a selected tuple to the output buffer. If more than one search module was ready to send such a tuple, the output arbiter instructed one module to send its tuple and the other(s) to set mark bits in the response store, indicating that the tuple might be sent in subsequent disk revolutions. Finally, the output arbiter transmitted selected tuples into the data channel where they were returned to the user.

During the search process, a row or sector of a band (refer to Figure 13) was sent by the read head to its assigned search module by way of sixty four data paths originating at the heads and terminating at the output buffer. The search module then compared that row with the search operand. If a match occurred and no further processing were required, the search logic would wait for a signal from the output arbiter to either send the row through to the output buffer, or to place a mark in the response store indicating that that row could be sent in subsequent revolutions. If a match occurred and further processing were required[5] a mark corresponding to that tuple was placed in the response store and processing continued onto the next revolution.

If matches were found in a row on a previous revolution, the response store would contain a mark so indicating. The search module would combine the result from the current comparison with the previous results and either mark the response store with the accumulated result in anticipation of further processing, wait for instructions from the output arbiter, or disregard that tuple as not having met the selection criteria.

RARES was designed to provide certain advantages over the CASSM and RAP.1 models. To alleviate the problems of output contention experienced by those earlier models, RARES depended on its orthogonal layout to moderate the flow of information to the output channel. Since tuples were stored across tracks, RARES required many fewer search modules, each processing up to sixty four tracks at one time. The simultaneous selection of tuples by the individual search modules was not the bottleneck it was in CASSM and RAP.1 where there was a search module for every track on the disk. So the output buffer in RARES could generally send out all of the tuples held in the output buffer within the time required for one disk revolution.

Still, there were disadvantages to the RARES model that were inherent in the LPT devices as a whole, namely the slow speed of the fixed-head-disks. This made the design non-cost-effective when applied to large databases. There were other disadvantages specific to RARES, however. One was the wasted space that sometimes appeared at the end of a band row In the orthogonal layout. The bandwidth could be made smaller, but this might result in a field value of the tuple being split across two band rows, requiring more than one disk revolution to process. In any event, if the bandwidth were reduced to for example, thirty two, one half of the sixty four data lines from the read heads would be transmitting data, while the other half would remain unused.

E. SURE - A Search Processor For Database Management Systems

The next database machine design to be discussed was started in 1974 and published in 1978 by Leilich, Steige and Zeidler [LEIL 78] of the Technical University of Braunschweig, Germany. Like the previous three models, SURE belongs to Qadah's on-disk DB category, but had a multiple operation stream - multiple data stream processing multiplicity. Primarily, SURE was developed to offload selection operations from the GPC which proved to be very inefficient in those tasks.

While previous machines used fixed-head-disks and associative memory search techniques, SURE utilized moving-arm-disks and linear searches. The disk was modified to allow parallel reading from all of the tracks comprising one cylinder. The data streams from the cylinders were joined together and sent for processing by a number of parallel search modules. These modules were designed to evaluate the data sent to them according to independent queries, making this an MOMD design.

---

5 For a more complex query.

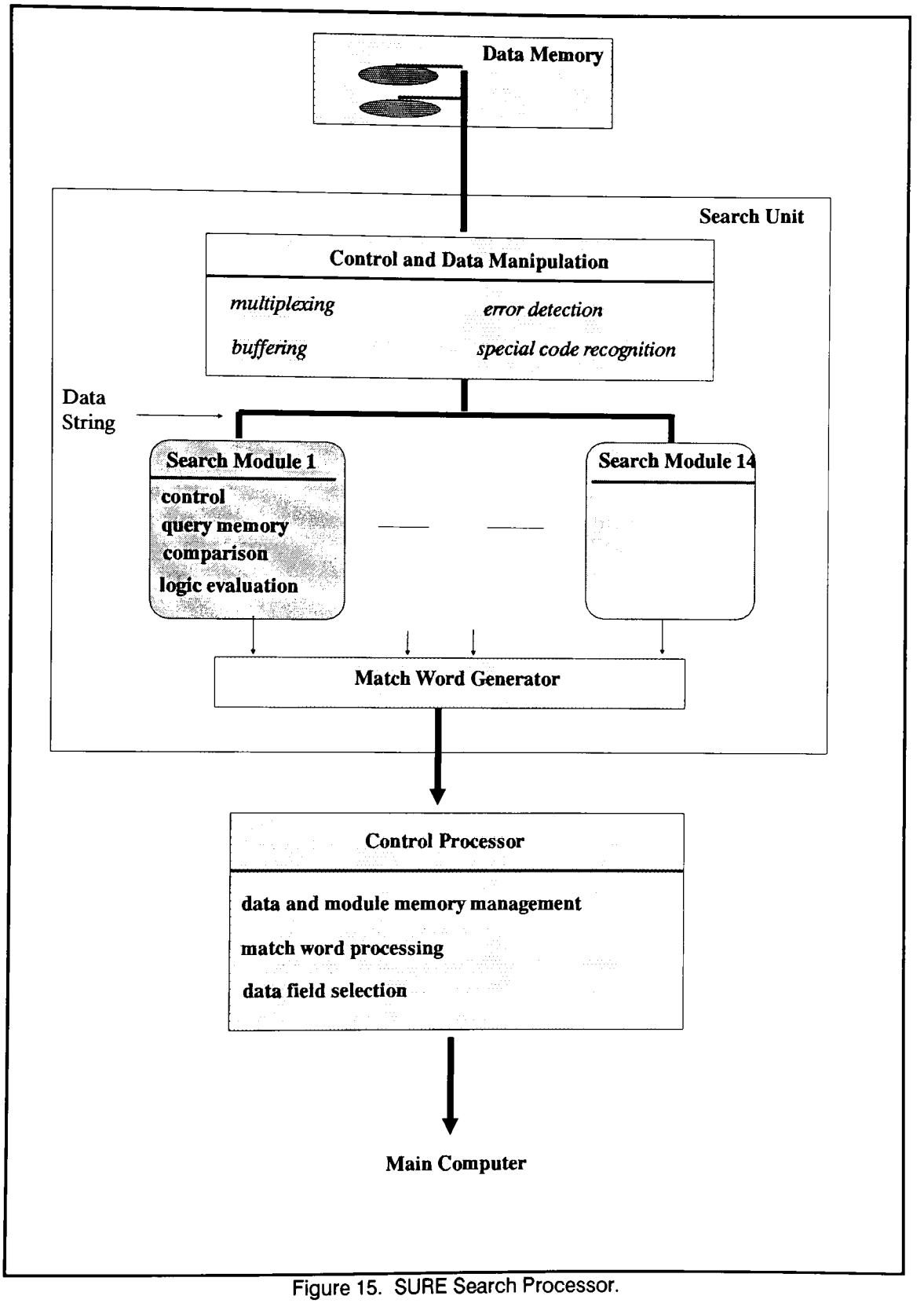

Figure 15. SURE Search Processor.

The SURE search processor had three major components (see Figure 15): data memory, control processor and the search unit. Data memory was comprised of a nine surface movable-arm-disk device with one head per surface. The heads were modified to allow concurrent reading mechanisms to be active on any one head or on all nine simultaneously. Writes could take place from only one head at a time.

The control processsor was a *minicomputer* which communicated with the main computer and was connected to it by a data channel. Its connection to the search unit was through a channel on the match-word generator and through the control and data manipulation segment. This processor controlled reads and writes performed by the search units, as well as controlling those units' evaluation of data.

The third component of the SURE structure was the search unit. The search unit itself had three parts: control and data manipulation, match-word generator and up to fourteen separate search modules. The search unit could be directed, in its control data manipulation segment, to read all of the data from a cylinder in one revolution and to evaluate all of the tuples read by as many as fourteen different queries simultaneously.

Each search module contained a special hardwired processing unit and a small RAM where the query was stored. The query consisted of instructions and operands which together acted like a program. This program was restarted with each tuple that passed through the search module and determined if a match was found between the record and the search operands. If there were such a match, a signal was sent to the match-word-generator.[6]

The match-word-generator held the address of each tuple as it was being processed by a search module. Also, this generator contained one bit position for each of the fourteen search modules. If a module was processing a tuple that met the selection criteria, the bit for that module was set. If at least one bit was set by the end of a round of processing, the match-word-generator sent its "match-word"[7] to the control processor. This component in turn, returned the appropriate tuple to the GPC.

SURE had some advantages over previous designs. It was less expensive because it used a moving-arm-disk device, which was also faster. It had the ability to perform multiple comparison operations in parallel. However, the main disadvantage was its poor cost/performance ratio. Since the search units could only process tuples from one cylinder at time, the total selection operation required as many disk revolutions as there were cylinders on the disk.

F. Content Addressable File Store - CAFS.

The CAFS design was first introduced in 1976 [MITC 76] but by 1979, enhancements had been made in the system structure making it one of the only database machines capable of supporting full joins and projections on relations. The original CAFS design is shown in Figure 16. Moving-arm-disks were used to store the database. Each disk could read only one track at a time, but the data from up to twelve disk units were combined into a single data stream and sent for further processing.[8]

---

6 Much like setting a "mark bit" in RAP.1 or RARES.

7 i.e. a series of bits, one for each search module.

8 Somewhat reminiscent of the SURE design.

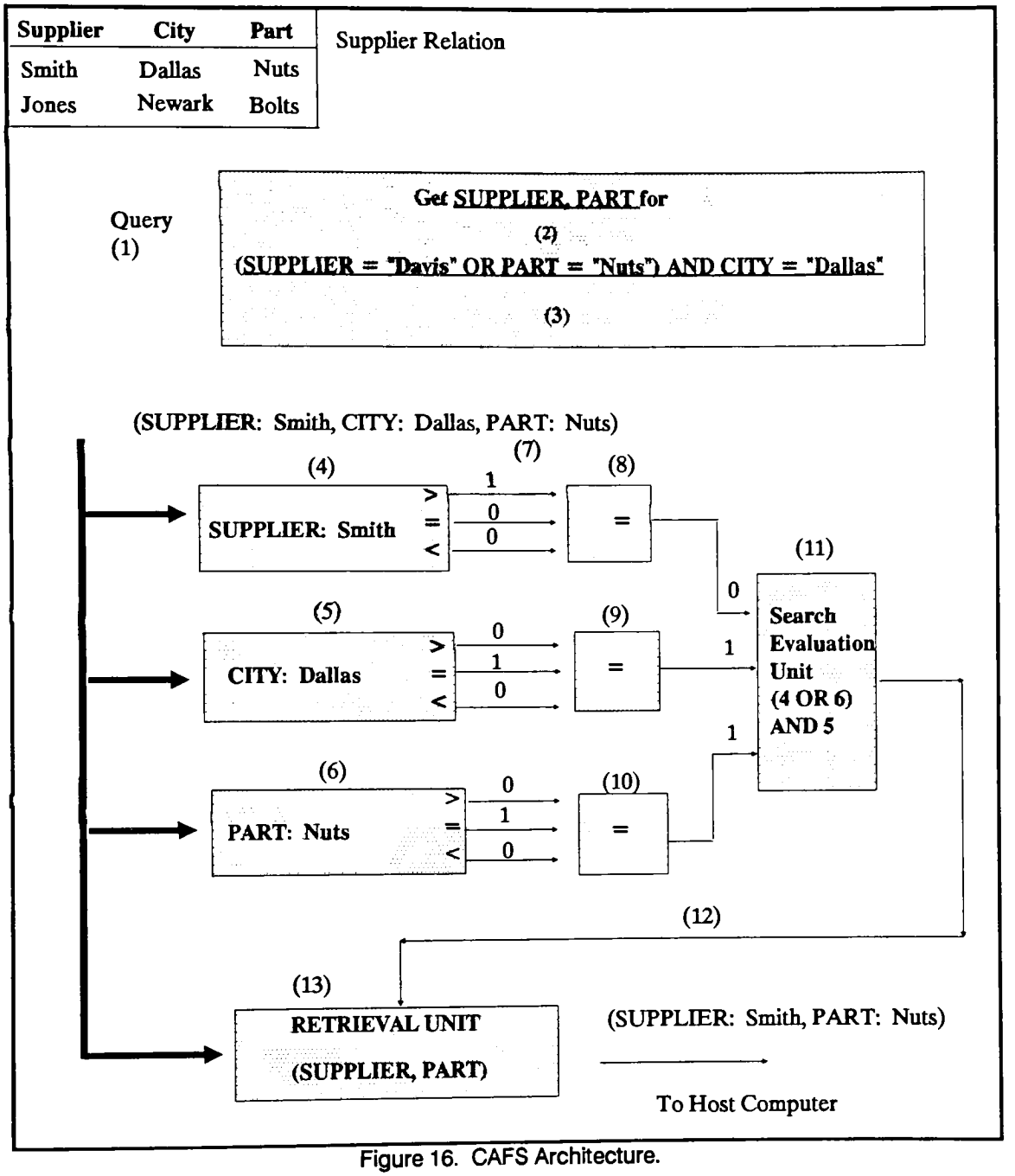

Figure 16. CAFS Architecture.

Referring to Figure 16, the CAFS instruction (1) consisted of the components to be retrieved (2) and the selection criteria (3). These criteria were used in setting up comparand values in key registers (4-6). Tuples read from the disk were fed sequentially into each of the key registers and set a latch (7) indicating the match status of the component with its respective key register value.

For example, CITY:DALLAS located in key register 5, matched the component found in the retrieved tuple. This latch would then be set to 1.[9] The latch comparators (8-10) contained the acceptable conditions determined by the selector portion of the query. Depending on the input provided these comparators by the latches set in the key registers, the output sent to the search evaluation unit (11) was set to 0 or to 1. The search evaluation unit used this input as part of a logical expression, determined by the terms specified in the selector. It tested to see if this expression were true. If the expression was true, a signal (12) would be sent to the retrieval unit (13) which extracted the desired components from the tuple and forwarded them to the host computer.

The search evaluation unit executed one operation at a time against the single data stream comprised of the output from up to twelve disks. This feature places CAFS in Qadah's SOSD processing multiplicity category. What is unique in this design, compared to those previously mentioned, is that the query processing place is considered a "hybrid" between on- and off-disk. The on-disk level is processing that took place up to and including the search evaluation unit. Selected tuples were then sent to the off-disk level, the retrieval unit. Here the desired fields were projected from the tuple and sent to the host.

The enhancement to CAFS that provided for the processing of full joins was a single bit array store. The purpose of this store was to map a list of field values in a concise and compact manner. If in a relation there were some number (x) of unique field values, a kind of index could be associated with each member, resulting in x field value/index pairs. For example, if the only values for the field CITY in a relation SUPPLIER were DALLAS, BUFFALO and ATLANTA, then the fields value/index pairs would be {(BUFFALO,1),(DALLAS,2),(ATLANTA,3)}.

All or part of the original list of field values could be uniquely codified into an x-element bit array. Each bit position corresponded to a different field value, determined by the various field value/index pairs. A subset of the original list containing only the values DALLAS and BUFFALO for example, would be represented in the three bit array as (1,1,0). This could be translated as representing indices (1,2), labeling the sublist (BUFFALO,DALLAS).

An extra field called a "coupling index" was added to CAFS relations stored in the database (see Figure 17). This index field was mapped into the key values that could be used in a join operation. The enhanced CAFS architecture contained from one to thirty bit array stores (M_1...M_{30}) addressed by such indices. Figure 17 shows that the tuples in the SUPPLIER relation with the value DALLAS in their CITY field have a coupling index (I_{CITY}) value of 2. Those with BUFFALO have the value 1. The corresponding values are found in the coupling index fields for the PARTS relation.

SUPPLIER RELATION

S#	NAME	CITY	I_{city}
S1	Smith	Dallas	2
S2	Jones	Buffalo	1
S3	Johnson	Atlanta	3

PARTS RELATION

I_{city}	P#	PNAME	CITY
1	P1	Bolts	Buffalo
1	P2	Nuts	Buffalo
2	P3	Nails	Dallas

Figure 17. Coupling Indices in CAFS.

9 Indicating equality.

The CAFS structure with the bit array store enhancement is shown in Figure 18. The key additions here were the bit address filter and of course, the bit array store. The filter extracted the index value I from a coupling index field on the tuple being processed. This was passed to the bit array store where it could either be set or interrogated.

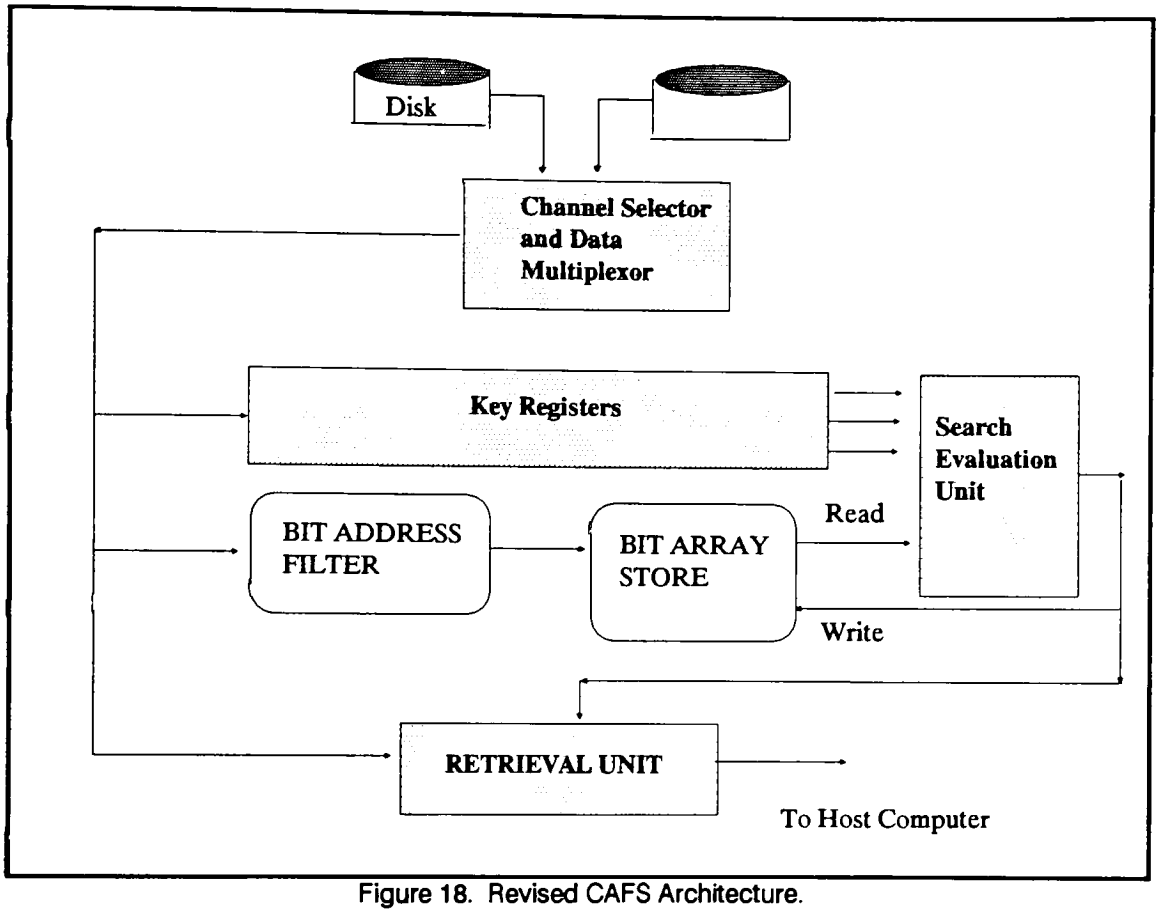

Figure 18. Revised CAFS Architecture.

Up until this time, joins on two relations had been implemented by performing selections on those relations separately and then performing the join operation in the derived subsets. CAFS however, using the bit array store concept, first performed the join and then the selection. An example of a query execution illustrates the concepts of the join and of the bit array store.

Let us use the following query in this discussion:

- Get a list of parts supplied and used in Dallas

First, the bit array store M was cleared representing an empty list of parts. For every tuple in SUPPLIER (S#,NAME,CITY,I_{CITY}) where CITY has a value of "DALLAS" (i.e. I_{CITY} = 2), the corresponding bit in the array store was set to 1. M(I_{CITY}) := 1. M = (0,1,0). This pattern in the array store M represented "DALLAS".

For each tuple in PARTS (I_{CITY},P#,PNAME,CITY), the corresponding index value is interrogated. In other words, if M(I_{CITY}) = 1 (i.e. M(2) = 1), then the value of PNAME from that tuple was put into a list, P. The final list P represented all of the parts supplied and used in DALLAS. By storing the cou-

pling values across relations in the bit array store in this manner, the volume of data returned from the join operation was minimized. The list P would be sent by the retrieval unit to the host.

CAFS offered several advantages over some previous designs. It utilized moving-arm-disks, making it faster than those models using fixed-head-disk units. Also, as was demonstrated, the bit array store made full joins on two relations possible.

Several disadvantages are apparent however, the first involving updates required to the coupling indices and ultimately to the dimensions of the bit array store as more unique key values were added to the relations in the database. Also, processing of data was performed in linear time; since the tuples read from the various disk devices were combined into one data stream, the time required to process that stream was directly proportional to the number of tuples it held.

G. RAP.2

RAP.2 was designed to overcome many of the problems associated with its predecessor, RAP.1. It is the first of the database machines reviewed here belonging to the relation indexing level of Qadah's taxonomy. Its query processing place is off-disk[10] and it has an MOMD processing multiplicity.[11]

Two specific problems inherent in RAP.1 that were solved in RAP.2 were limitations on the size of the database and the inability to process multiple simultaneous operations. The database in RAP.2 was stored on moving-arm-disk devices which had much higher storage capacities than the fixed-head-disks of its predecessor. Thus, RAP.2 could support a larger database than RAP.1

Further, RAP.2 could process multiple operations simultaneously under certain circumstances. Relations from the database were sent to an off-disk processing level by way of a data channel (see Figure 19). This level was comprised of a fixed-head-disk with logic units that searched all of the tracks in parallel. If a complete relation could be stored on the fixed-head-disk, then more than one database operation could be performed on that relation, providing MOMD processing multiplicity.

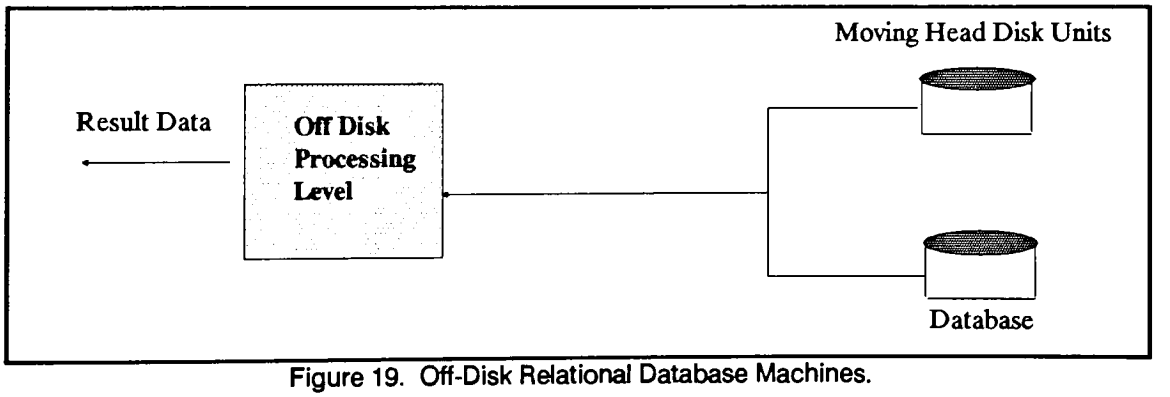

Figure 19. Off-Disk Relational Database Machines.

---

10 Unlike all of the previous designs mentioned, except for IFAM.

11 Like SURE.

The improvements implemented in the RAP.2 model involved both changes in architecture and in the organization of the physical data. Architecturally, the Controller was the most significantly enhanced component over RAP.1's design (see Figure 20). While a hardwired and more or less inflexible unit in the earlier model, RAP.2's Controller was implemented by a PDP 11/10 computer.[12] This offered a considerable redistribution of the workload from the cells onto the controller.

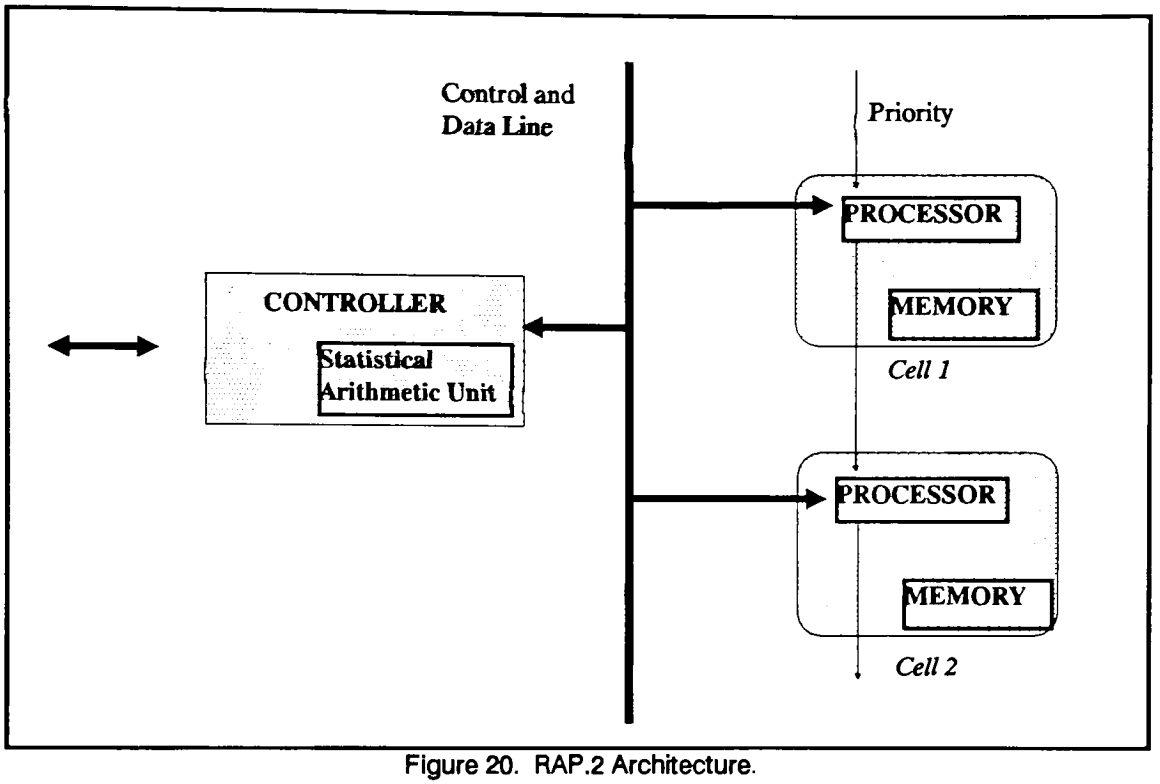

Figure 20. RAP.2 Architecture.

The cells were now required to perform only those tasks directly related to the tracks with which they were associated. They no longer communicated with every other cell in the system, but rather sent their information along a bus to the controller. The controller also was able to provide instructions to the individual cells to execute sets of instructions that were different from those executed by any other cell.

A "priority" line ran between all of the cells. This line was used to gather information from the cells quickly by sequencing the controller's access to cells that had completed their processing. This was done instead of polling each cell sequentially to get the required information.

While the changes in the physical organization of data from RAP.1 to RAP.2 were not as extensive as the changes resulting from the new controller, they were nonetheless significant and should be mentioned. With regard to the track layouts, RAP.1 stored the relation name and the address of the processing cell at the beginning of the track. This information was followed by tuples which were

---

12 A product of Digital Equipment Corporation, this is the predecessor of the VAX 11/780. See [BAER 80] for more details.

separated by gaps. RAP.2 on the other hand, determined the cell address by interrogating a switch handled by the controller. The relation name was stored in a sixteen bit register and was defined by the programmer.

In the record format for RAP.1 two bit length codes preceded every field value in a record. While space for these codes remained in the record formats of RAP.2, the information was actually stored in a register called a "length code RAM".

While addressing specific problems of the RAP.1 design, RAP.2 was not without its shortcomings. First, for efficient processing at the off-disk level, any relation in the database had to be able to be loaded in its entirety onto the fixed-head-disk. This placed some limitations on relation sizes if efficient processing was to be achieved. Second, and more significant, a bottleneck was generated as a result of the difference between the speed of the two types of disk devices used. The moving-arm-disk could transfer relations from the database much more quickly than the fixed-head-disk could process that information. The overall speed of the system then was reduced to the speed of the slower device.

H. Direct

DeWitt introduced the DIRECT database machine design in 1979 [DEWI 79]. Placed in the same classifications os RAP.2, DIRECT was also intended to overcome some of the problems with the RAP.1 design. This machine however, appeared to be altogether a more sophisticated design than either of the RAP models.

DIRECT was developed as an online, mulit-user database machine, a concept which no other model had undertaken up to that time. Utilizing paralliel processing and a backend controller, DIRECT was capable of supporting both intra-query concurrency[13] and inter-query concurrency.[14]

The architecture of DIRECT (see Figure 21) had six main components: a host processor, mass storage devices, a back-end controller, query processors, charge-coupled-device (CCD) memory modules and an interconnection matrix. The host was a PDP 11/45[15] running under the UNIX operating system. Its responsibilities included handling all communications with the users, as well as running a relational database management software system. This software compiled the users' queries into individual "query packets" of relational algebra operations. The database itself was stored on moving-arm-disk devices, much the same as in RAP.2.

A direct memory access (DMA) link connected the host to the backend controller, a PDP 11/03. Query packets were tranferred to the controller which determined for each such packet, the number of processors to be assigned to its execution. This determination was based on the priority of the query, the size of the relation that was being referenced by the query, and the type of relational algebra operations involved. The controller also handled paging to the memory modules (see below).

---

13 Each processor searching a subset of the same relation.

14 Two processors executing different queries on the same relation.

15 This, the PDP 11/40 and the PDP 11/03 referred to below, are other members of the PDP family of minicomputers.

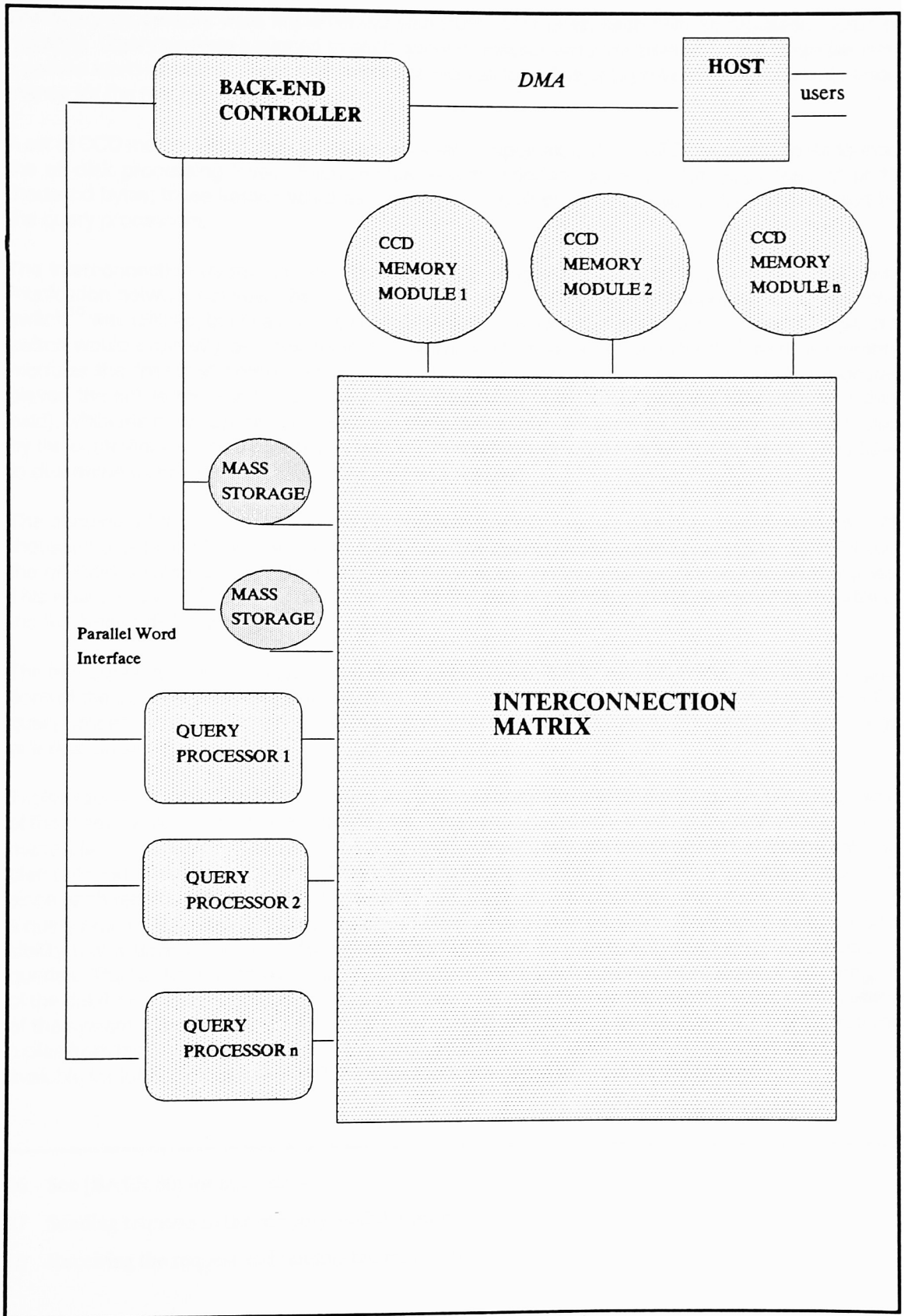

Figure 21. DIRECT System Design.

The query processors were implemented with PDP 11/03 units, each with 28 thousand words of memory. Query packets assigned to each query processor were transmitted by the controller over a parallel word interface. Processors were not allowed to be idle; this provided more efficient performance for the machine overall.

A set of CCD memory modules comprised another component of the DIRECT system. These formed the off-disk processing level. Each module was partitioned into fixed size "page frames" of 16 thousand bytes; these frames acted as a cache to hold all or part of a relation being processed by the query processors.

The Interconnection matrix provided the key to the processing capacity of the system, the communication network between the query processor and the CCD memory modules. A cross-point switch[16] was utilized, but in a non-traditional manner. Whereas the processors connected to such a switch would ordinarily be considered the "active" components of the network[17] and the memory modules the "passive" components,[18] in DIRECT, the opposite was true. The memory modules played the active role by "broadcasting" their contents (i.e. by identifying the relation(s) that they held), while the query processors assumed the passive role. Whenever a processor was instructed by the controller to search a particular relation, that processor would "listen" to the memory modules to determine which one held the relation to be processed.

The contents of the page frame in the appropriate memory module was then transferred to the 28 thousand byte RAM of the query processor where the query operations were executed. DIRECT had the capacity to support relations of any size since each relation was partitioned into page frames. This was in contrast to the RAP.2 design where the size of the relation was restricted to the size of the fixed-head-disk memory.

The interconnection matrix allowed the query to quickly switch to new page frames containing portions of the same or different relations. Also, by moving the page frame contents into the RAM of the query processor, multiple processors executing different query packets could search the same page of a relation simultaneously.

Tuples found to satisfy the search criteria were written back into a temporary relation in a page frame of the memory module from which they came. This method was different than the RAP models which made use of mark bits. The developers of DIRECT had three main reasons for choosing this implementation. First, mark bits would severely curtail the performance of DIRECT's query processors, since each relation would have to be "locked" (i.e. marked unavailable) by the processor executing a query against it; intra-query concurrency would not have been possible. Second, there was a possibility that a user would want the selected tuples added to the database, or used in subsequent queries. The backend controller had the capacity to make the temporary relations a permanent part of the database. Finally, the temporary relation was seen to enhance the overall output performance of the system. These relations acted as a sort of buffer inasmuch as the controller transferred the tuples from the temporary relation to the host. Had mark bits been used, the relations would not be avaiable for further processing until the selected tuples were transferred to the host.

---

16 See [BAER 80] for more details.

17 Sending requests to the memory modules for data.

18 Receiving the request and sending the information.

DIRECT did encounter some problems with bottlenecking between the mass storage devices and the CCD memory modules. Also, performance of the machine as a whole was degraded by the amount of time required by the controller to coordinate the simultaneous execution of different query packets and to manage the CCD memory modules. Still, DIRECT was perhaps the most conceptually refined design of any examined so far.

I. Database Computer - DBC

Much as the designers of DIRECT endeavored to provide a system with capabilities beyond those of previous models, the creators of DBC developed a machine intended to directly compete with existing software database management systems [BANE 78], [BANE 79]. DBC was designed to support very large online databases[19] and to provide a mechanism for security implementation. The latter was a function performed, if at all, by the host computer in previous database machine designs.

The DBC acted as a hardware backend machine to one or more than one host (referred to as the Program Execution System - PES). User programs resided on the PES which communicated with DBC, sending it processing commands. Requested records were returned by DBC to the host.

Structurally, the DBC architecture is as shown in Figure 22 and was comprised of seven major subsystems: the keyword transformation unit (KXU), structure memory (SM), mass memory (MM), structure memory information processor (SMIP), index translation unit (IXU), database command and control processor (DBCCP) and the security filter processor (SFP). Each component was a physically separate unit with its functions distinct from all others. In addition, each component was individually optimized for its specific function in order to prevent any one from causing a performance bottleneck.

In attempting to provide support for very large databases, DBC's designers were faced with the ongoing problem of the infeasibility of associative memories in processing large amounts to data. Their solution was to divide the memories into a number of blocks, each of which was content-addressable. DBC utilized a concept called "partitioned content-addressable memories", or PCAMs. The memory components (MM, SM and SMIP) were each designed as PCAMs. Area pointers stored in and managed by the SM, served as indicators to the PCAM partitions in the MM. Data in the MM component were located using content-addressing techniques.

DBC's architecture was divided into two parts, referred to in the design as "loops"; the data loop and the structure loop. The former was comprised of the MM and the SFP and was responsible for storing and accessing the database, as well as enforcing the security mechanism. The second loop included the KXU, SM, SMIP and IXU. These components determined which records were authorized for insertion into the database. Both loops were connected by the DBCCP.

The MM component was implemented using moving-arm-disks, each cylinder of which represented one PCAM block as a page.[20] MM was modified with logic units called track information processors or TIPs. The number of these units was equal to the number of tracks on a cylinder. Working in parallel, the TIPs were capable of associatively searching the contents of an entire cylinder on-the-fly in one disk revolution. Selected records were sent to the SFP where access security was enforced by checking those records for the proper authorization levels.

---

19 10^{10} bytes or a "gigabyte".

20 Putting this machine into Qadah's page indexing level category.

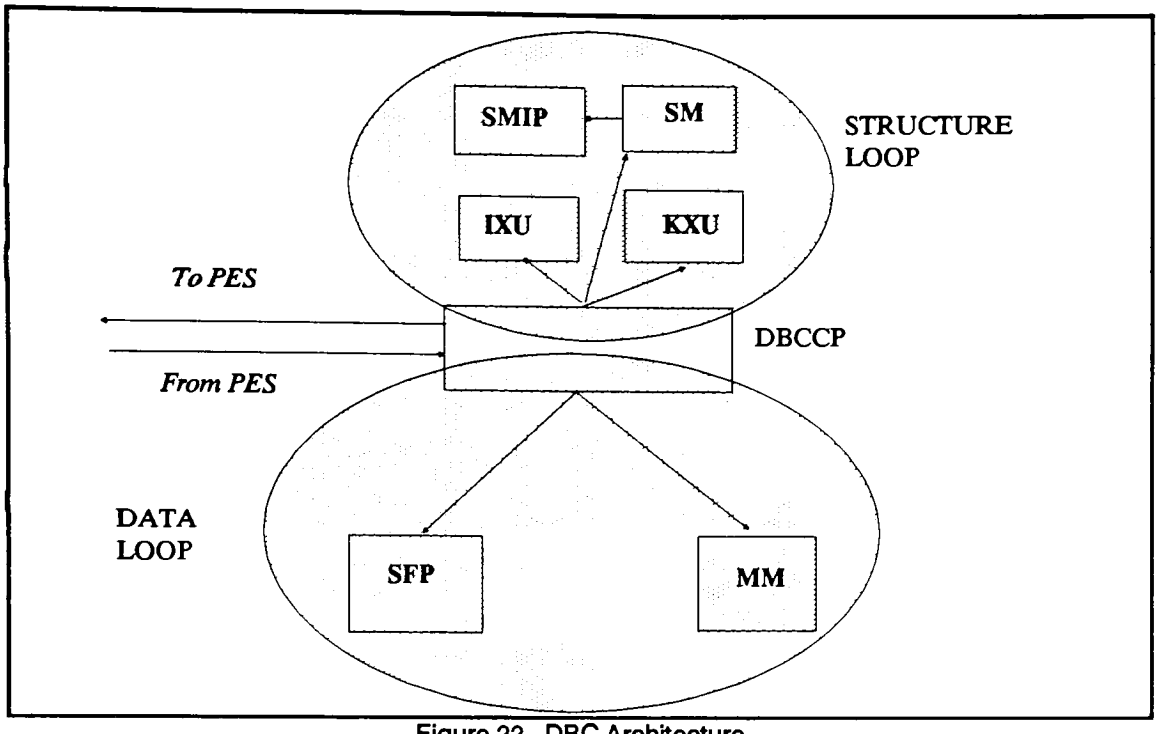

Figure 22. DBC Architecture.

In the structure loop, the SM contained additional information about the database, such as the area pointers to the MM partitions mentioned earlier. Much smaller than the MM, the SM was also implemented as a PCAM. Its main purpose was to determine which of the MM cylinders were to be searched. The roles of the other components are best illustrated by a query processing example.

Every record stored in the DBC database had three components; the record body, a set of keywords (attribute/value pairs) and a security atom (a number indicating the security specifications for the record). Indices for the keywords were stored and maintained in the SM, providing information regarding where in the MM[21] records with such keywords were stored.

The user's query was translated by the PES into DBC commands. These commands were received by the DBCCP component which controlled the operations of the backend system. The DBCCP sent intructions to the KXU, telling it to access the SM keyword index to determine which entries in that index were appropriate to the search. All of the index terms selected were sent to the SMIP. Here, the aggregate of the index references was made available to the DBCCP.

The DBCCP in turn, examined the security atom portions of each index term and eliminated those to which the user did not have the proper access authority. The final set of index terms was sent to the MM where the TIPs processed the tracks of the appropriate cylinder, as determined by those index values. Selected tuples were returned by the MM to the PES via the DBCCP.

---

21 In which partition.

The DBC system encountered performance bottlenecks in the MM, DBCCP and SFP due primarily to their complex design. There were no provisions in the model for concurrency control of queries as was present in the DIRECT design and so this remained an SOMD machine. Yet the attempt to provide database security and the concept of the PCAMs made the DBC a significant contribution to the field of database machine research.

J. Distributed Associative Logic Database Machine - DIALOG

DIALOG, introduced in 1980 [WAH 80] is a backend database machine belonging in Qadah's MOMD, hybrid, relation indexing level category. The designers attempted to produce a machine utilizing simple but intelligent processing logic acting directly on the storage devices in order to minimize the amount of data transferred. They, like the designers of DBC, wanted a machine to support very large databases with high performance, yet low cost.

The general architecture of DIALOG is shown in Figure 23. The database was stored on fixed-head-disk units on the system's data module components. Each data module was connected to every other data module by the Communications Network. A group of data modules (called a "cluster") was connected to the host computer by the Backend Controller. DIALOG supported modular expansion of the system by interconnecting multiple data module-communications network-backend controller groups together. These groups were coordinated by a higher level backend controller which in turn, was connected to the host.

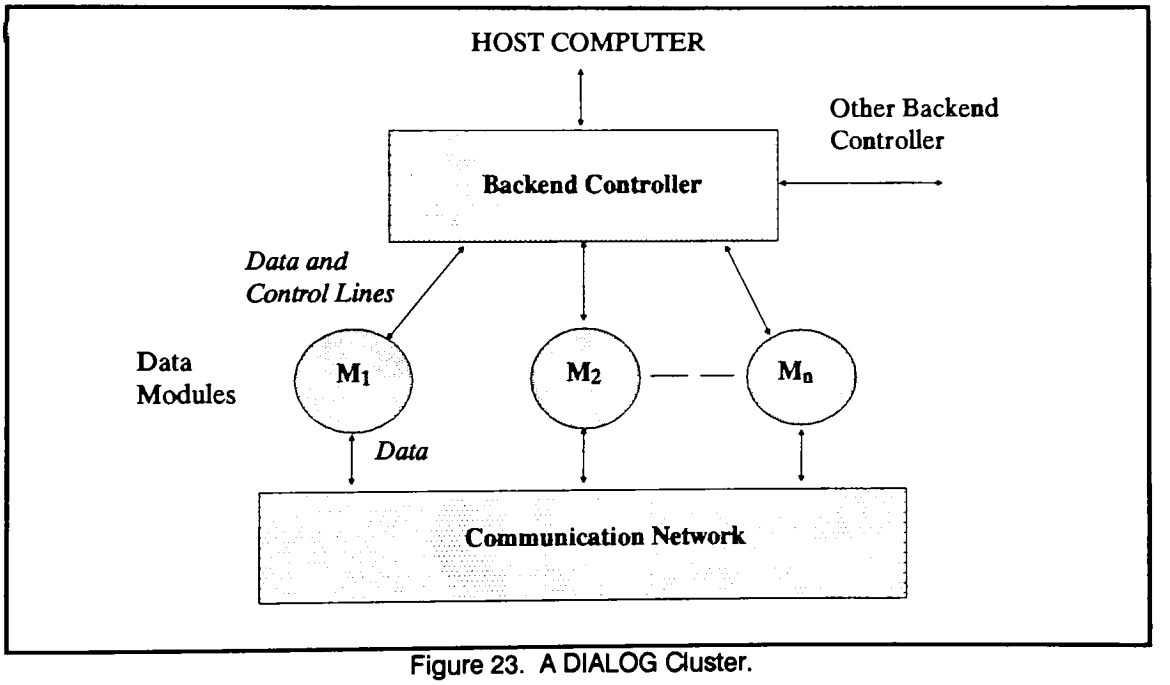

Figure 23. A DIALOG Cluster.

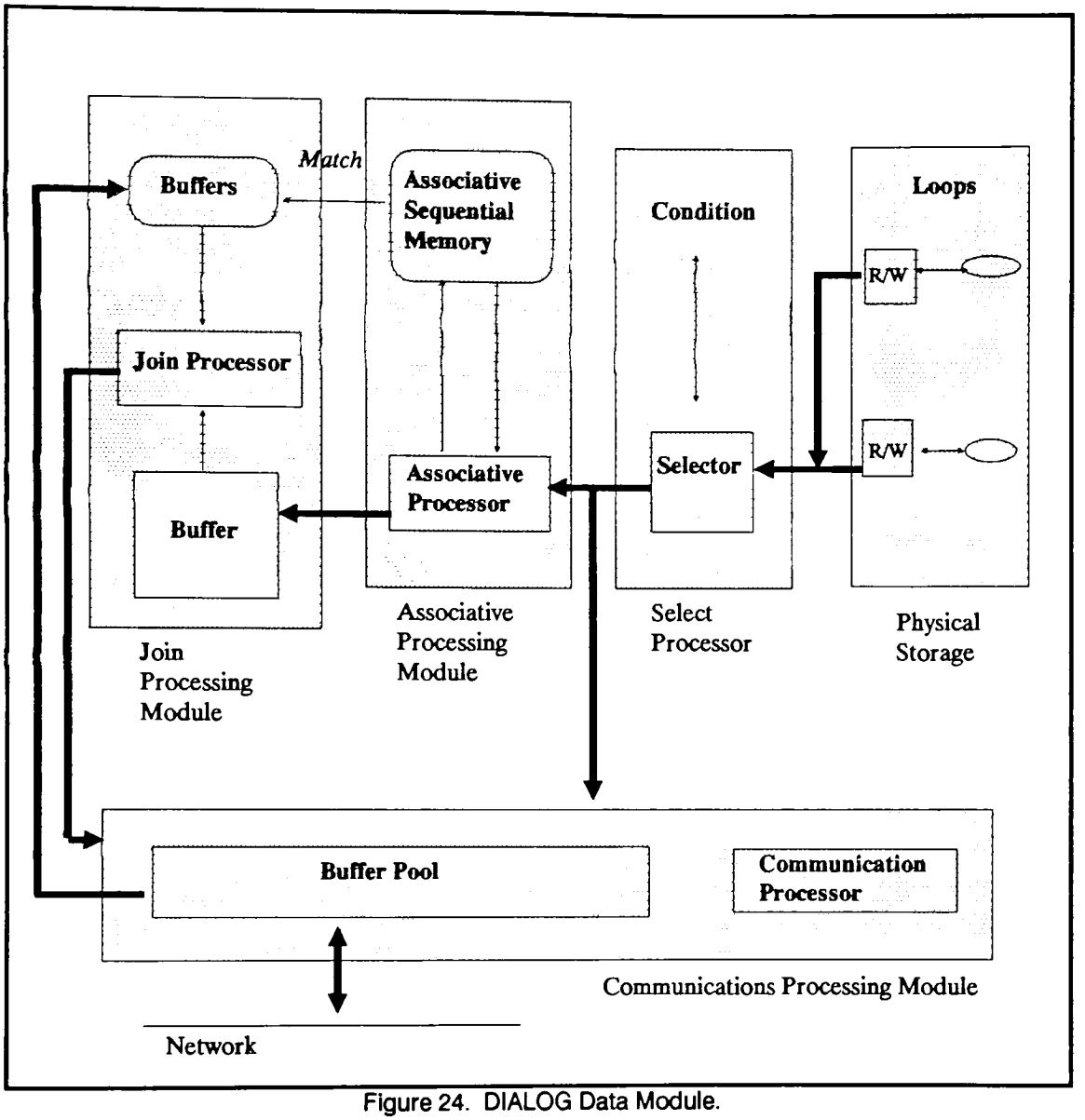

Figure 24. DIALOG Data Module.

The backend controller was responsible for both the control of data and for communications. Queries generated at the host were sent to the controller; responses from the processing units were sent back through the controller to the host. The controller preprocessed the queries it received, making them executable by the data modules. It also scheduled and initiated the operations of each data module.

Simple database operations executed against one relation[22] could generally be processed in one data module. This comprised the on-disk level of the hybrid processing place classification. More

22 For example, selection or projection.

complex operations involving more than one relation[23] required transferring all of the selected records to one data module for processing; this represented the off-disk level. If two relations were involved in the processing and they resided on the same data module, one relation could be transferred to a buffer; then the processing as described above could continue.

The data module was the most significant component of the DIALOG system. It was comprised of five submodules: the physical storage device, a selection processing module, an associative processing module, a join processing module and a communications processing module. As mentioned earlier, the physical storage device held the database and was implemented using a fixed-head-disk device with a single read/write head per cylinder (see Figure 24).

The selection processor connected directly to the read/write heads, processed data on-the-fly. For simple operations, selected tuples were sent to the buffer pool for output onto the communications network. In the case of more complex operations, the selection processor performed a sort of "context" search, as in CAFS, selecting those tuples suitable for further processing. These tuples were sent to the associative processor for a more detailed "content" search, comparing the tuples against search information stored in this module's associative sequential processor. Tuples meeting the associative processing module's selection criteria were then sent on to the join processor where the actual joining of the tuples from the two relations took place. The communication processing module received instructions from the backend controller. It also handled data in the buffer pool and communications with other data modules.

The only significant disadvantages inherent in the design of DIALOG were the slower fixed-head-disk and the overhead involved in implementing all of the various processing subunits. In general however, the design was sound and made full use of the concept of providing modularity that would support database growth.

K. RAP.3

The final design to be reviewed in this chapter is the third version of the RAP machine, RAP.3 [OZKA 85]. It is not important here to reiterate the basic RAP design, but rather to discuss the changes implemented in RAP.3 and the reasons why those changes were made over the previous two models.

Two changes of a general nature made in RAP.3 were the use of microprocessors and random access memories. The hardwired cell processors of the earlier models were replaced by microprocessors. Advances in microprocessor technology by this time made them very fast and relatively inexpensive. A parallel combination of these microprocessors was used in redesigning the cell processor (see below). RAM was chosen over magnetic bubble or CCD memories because it was more controllable. If extra time were required to process a particular tuple, the RAM could be stopped and restarted, unlike the other types of memory which could only be slowed down.

---

23 For example, a join.

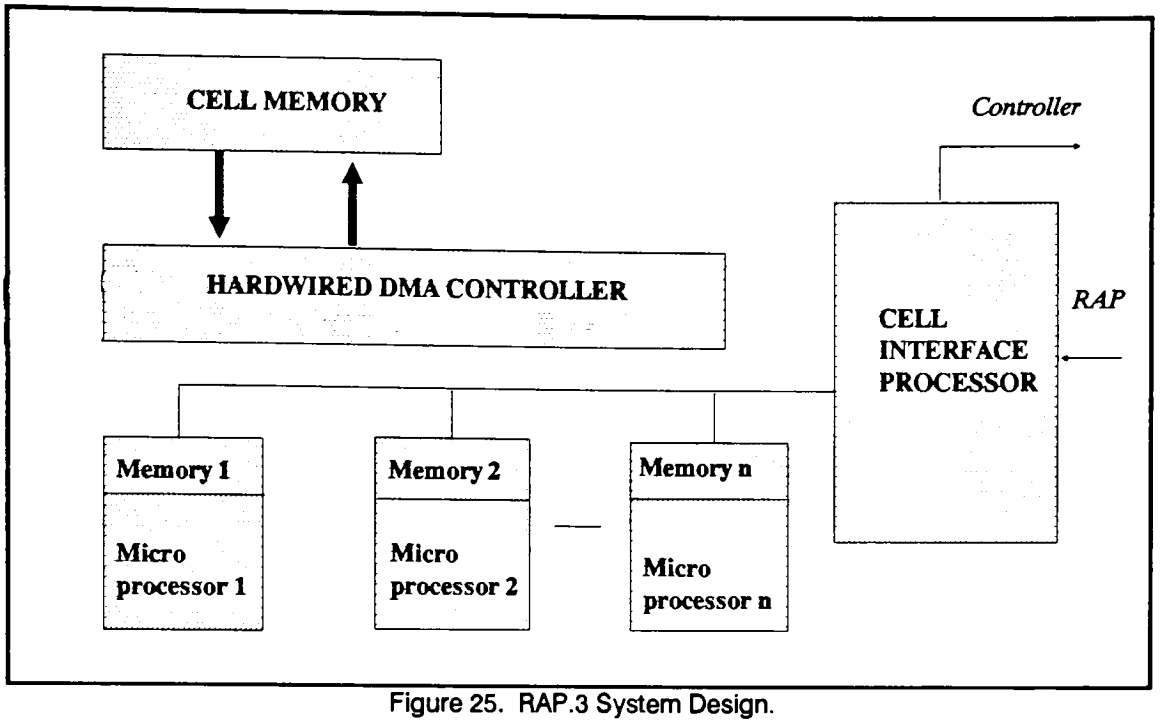

Figure 25. RAP.3 System Design.

The RAP.3 system architecture is shown in Figure 25. Cells in this design consisted of four components: microprocessor subcells, direct memory access unit, cell interface processor and the cell memory.

The microprocessor subcell component was comprised of a number of Intel 8086 microprocessors each capable of executing RAP instructions in parallel with other subcells. Every subcell was locked in a "wait" loop until activated by the DMA which fetched a tuple from cell memory[24] and placed that tuple into the subcell's RAM. The subcell repeated the given RAP instruction on every tuple placed in its memory. The subcells could be partitioned so that more than one query could be executed in parallel. The activities of the DMA[25] were controlled by the cell interface processor. Information such as record delimiters and header blocks which were previously stored in cell memory were stored in the cell interface processor of RAP.3. Also, this processor interfaced with the RAP controller, which had not changed significantly from RAP.2

III. Conclusion

This concludes the historical overview of database machine designs. Figure 26 shows the database machines discussed in this chapter, grouped according to index level, query processing place and processing multiplicity. Two major groups emerge; CASSM, RAP.1 and RARES, early models with database indexing levels, on-disk query processing and single operation stream-multiple data stream processing multiplicity; and the later models RAP.2, DIRECT and RAP.3, having relation indexing

---

24 A one to two megabyte RAM.

25 That is, fetching from cell memory, loading the subcell's RAM and writing tuples back to cell memory.

levels, off-disk processing and multiple operation stream-multiple data stream processing multiplicity. The next chapter will focus on the Teradata DBC/1012, the only successfully marketed database machine today capable of supporting very large databases.

Machine Group	Indexing Level	Query Processing Place	Processing Multiplicity
A	database	off-disk	SOMD
B	database	on-disk	SOMD
C	database	on-disk	MOMD
D	database	hybrid	SOSD
E	relation	off-disk	MOMD
F	page	hybrid	SOMD
G	relation	hybrid	MOMD

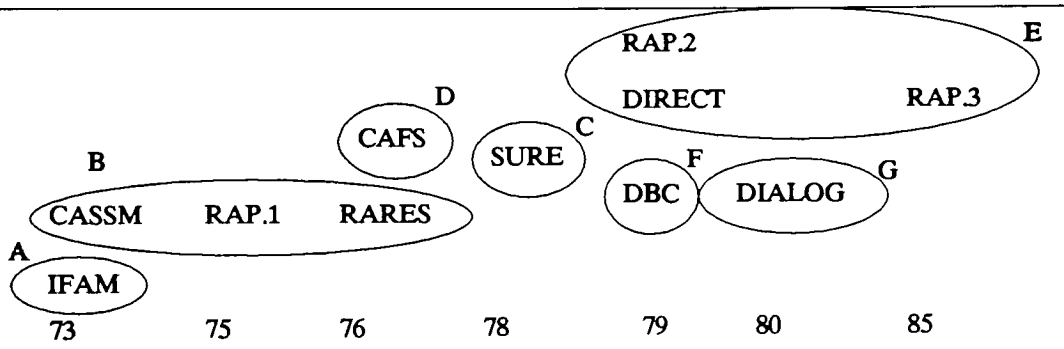

Figure 26. Major Database Machine Groups.

Chapter 3

The Teradata DBC/1012

I. Introduction

The research of Phillp M. Neches [NECH 84], a co-founder of Teradata Corporation (Teradata), provides Insight Into the reasons why that company desIgned the DBC/1012 as a dIstributed parallel-processIng machine, departIng from the apparent trend toward content-addressable memory techniques. Neches found that distributing the database access workload across a number of processors working In parallel offered higher economIes of scale[1] than did conventional von Neumann architectures or logIc-per-trak (LPT) devices. Before examining Neches' work however, it is Important to dIscuss some basic concepts regardIng the evaluation of performance for computer systems In general.

II. Performance Evaluation of Computer Systems

Ferrari [FERR 78] describes a computer system as an aggregation of both hardware components and software components. These elements are sometImes referred to as a system's "resources"; the set of hardware components specifically belng termed a "configuration". The attributes of each component[2] are known as "system parameters". Into a given computer system may be Introduced a "workload", a set of inputs[3] used in evaluating performance of the system in processing those Inputs. An evaluation study then, is one designed to gather information about a system's performance under a specified workload, with a given set of system parameter values.

A. Performance Indices

Performance In general is a measure of a computer system's efficiency and speed In processing. A performance Index Is that whIch represents some aspects of a system's overall performance. There are three major classifications of performance Indices: productivity, responsiveness and utilization. A performance index in the productivity category would measure how much information a system Is capable of processing within a given unit of time. Measurements of a system's throughput rate[4] Is an example of such an index. An index measuring responsiveness would be concerned with the amount of time between Introducing some Input Into the system, for example a query, and receiving the corresponding output, selected records for example. Elapsed time measurements are examples of a responsiveness performance Index. Finally, a performance index In the utilization category Is a rneasure of the ratio of time that a given system resource Is used during an interval of time, over the length of that interval. An example might be the amount of time a memory buffer is actually used during the processing of a transaction that takes five seconds to complete.

B. Evaluation Techniques

The means by which performance Information Is gathered Is referred to as an evaluation technique. Two techniques are widely accepted - measurement and modeling. Information collected by measurement Is obtained from an actual existing system; modeling on the other hand, simulates a system from which performance information Is gathered. The former is most often used to study the change in a system's performance after the introduction of some new component. Several measure-

---

1 i.e. the processing power of such a system would increase more than linearly with its cost [FERR 78].

2 For example, track size in bytes on a disk device.

3 Programs or queries, for example.

4 The number of transactions processed per second.

ment evaluation studies will be presented at the end of this chapter. Modeling is more suited to the development of new system configurations. Modifications to the parameters of the hardware resources can be controlled in an heuristic fashion to determine which configurations meet certain designated performance criteria at the lowest cost. Other modeling studies like the Bank of America evaluation discussed at the end of the chapter, use modeling techniques to simulate different processing environments for an existing computer system.

III. The Neches Study

Neches' research [NECH 84] was based on the modeling evaluation technique. His goal was to derive a configuration suited for database functions meeting a responsiveness performance index criterion of fifteen seconds and delivering the highest economies of scale.[5] Thirty configurations were modeled, each subjected to the same set of workloads designed to simulate other requirements. Out of the thirty architectures, three were found capable of satisfying the performance criterion: von Neumann architecture, LPT devices and distributed processing designs.

The von Neumann architectures, when used in database processing, store the database in pages in secondary memory, usually on disk. Data is transferred, "paged in", as required from this secondary storage to main memory for processing. These designs, Neches determined, were not capable of maintaining acceptable response times for processing relations of more than five thousand rows due to bottlenecks caused by the paging of information from disk memory.[6]

LPT devices on the other hand, had a more effective rate of transferring information from disk because of the processing logic located at each head. However, there were distinct disadvantages in performing projections or joins on large relations, those disadvantages resulting from constraints placed on such processing by the size of the content-addressable memory stores.[7]

The distributed processing architecture was that which divided the work of processing relations among many elements, each acting in parallel on one record or block of records. This method proved more economical for joins and projections of large relations than did the LPT devices. Neches considered this capability important enough to single out the distributed processing architecture as the one best suited of the three for database functions.

---

5 One would ideally look for the architecture that proved to be most economical under any combination of processing parameters. Neches found however, that the model's results were not so definitive.

6 In the next chapter, a discussion of IBM's new MVS/ESA operating system will show this company's solution to this bottleneck problem.

7 A fact demonstrated in several examples in the previous chapter.

IV. DBC/1012 Concepts

It was the distributed processing approach that formed one of the fundamental premises on which the DBC/1012 was designed. The other was that this machine would support the relational database model [DATE 86]. A combination of technological advances in the late 1970's and an increasing acceptance of the concept of relational databases helped create an environment conducive to Teradata's novel and somewhat speculative undertaking.

In 1978, Intel Corporation introduced its 8086 microprocessor with an internal data path width[8] of 16 bits and a basic clock cycle[9] that was two to three times faster than that of the earlier 8080 processor [BAER 80]. The 8086's instruction set processor[10] was equal in complexity to that of IBM's System/370. This was viewed by the founders of Teradata as an opportunity for providing, fairly inexpensively, the power required for distributed data processing in a centralized backend database machine.

The basic concept was to evenly distribute the processing of data among a number of 8086 processors so that each would be responsible for an approximately equal number of rows for every database table. Having the data distributed in this fashion would allow two types of processing to take place. First, all of the processors could perform equal amounts of work to satisfy one complex query for a single user. This is similar to DIRECT's concept of intra-query concurrency. Second, a number of different, simple queries could be processed by individual units for multiple users; this being the inter-query concurrency approach.

A. Interconnect Network Requirements.

The biggest problem that the DBC/1012 designers faced [COMP 84] was maintaining control over the work being performed by multiple processors. A control mechanism - an interconnect network- had to be devised which would provide four vital functions for the database machine. The first function was communication with all of the processors. Queries against relational databases are decomposed when processed into a series of relational algebraic expressions, "instruction steps" in Teradata jargon. If all of the modules were to process the same query, a method of broadcasting the steps to all of the processors had to be devised. Moreover, if only one or some of the modules were to process a particular query, that information would likewise have to be imparted. An interconnect with a broadcast mode of operations was needed.

The second function required of the interconnect was to ensure that all of the processors that had been sent instructions, received them. Since activity against a database can change the actual data contained therein, it is critical that processing be carried out in toto; partial processing is not acceptable. An acknowledgement of instructions had to be received from every processor to which those instructions were sent.

---

8 The amount of data transferred between the various components within the processor itself.

9 A digital timing device that coordinates activity of the actual circuits found in the microprocessor components.

10 That component of the processor which interprets and executes the microcode instructions.

Third, the interconnect would have to coordinate the status of the instructions sent to the processors; subsequent steps would have to wait until the first was completed by every processor involved. Finally, since each processor would deliver only a portion of the entire response to an instruction,[11] the interconnect would have to merge the partial results from all of the processors into one stream. Most likely this stream would have to be sorted into the order requested by the user submitting the instructions.

B. Implementation of the Interconnect

The DBC/1012 designers contrived an interconnect mechanism which was an implementation of the "tournament sort" algorithm [COMP 84]. This algorithm is best described by an example. In a sports tournament there are for example, eight contestants each of whom compete against one of the other contestants. If the contestants were numbered one through eight, contestant number one would compete against number two, number three against number four, etc. The winning contestants, one from each of the original four pairs, would move onto the quarter-finals. Here, two pairs of contestants would compete. The two winners of the quarter-final contests compete in the semi-final, from which one contestant emerges the winner.

Figure 1 shows how this tournament concept can be applied to a sort algorithm. The eight contestants represent "message packets" comprising for example, information from each processor giving the status of its operation - finished or not finished, "+" or "-". The "contest" which the message packets engage in is a comparsion of the information contained in those packets.

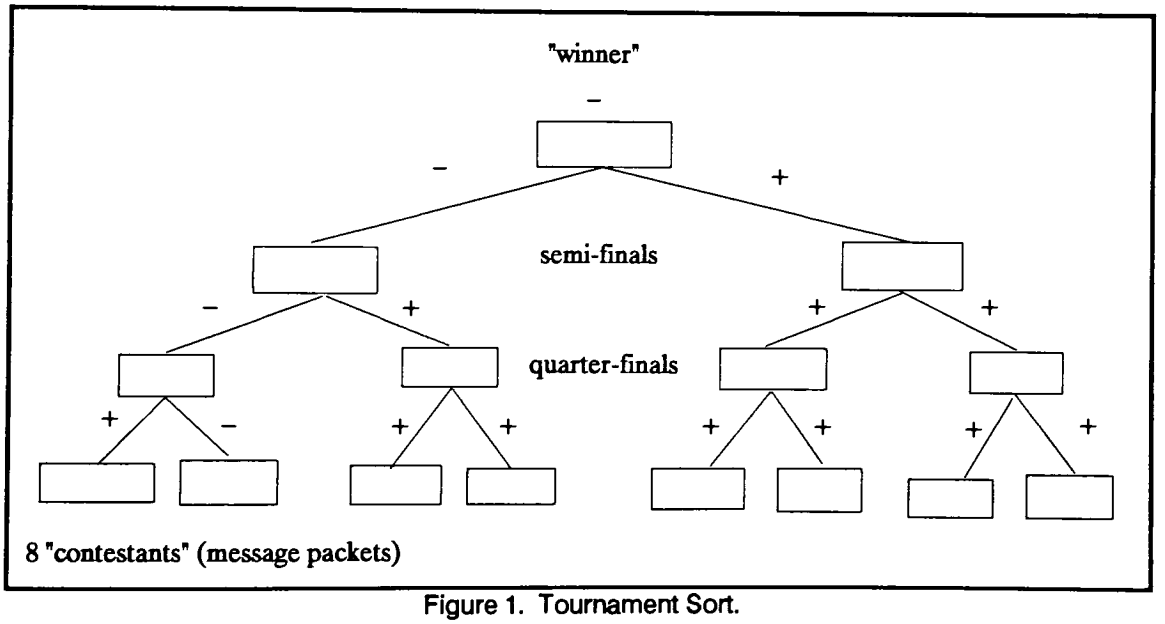

Figure 1. Tournament Sort.

Since, as stated earlier, all of the processors must have finished thier work before the next set of instructions is sent to those processors, a negative response to any processor's completion status would "win" the contest, would sort out. This negative response from even one processor would go

11 i.e. each would select certain records from a table.

all the way to the top, ensuring an "all or none" completion status among the processors. Thus the third function of the interconnect as discussed in the previous section is met.

This same mechanism satisfies the other three functions required of the interconnect, as well. The first function is met because this same design is inherently a broadcast interconnection network (see [BAER 80] for more details), capable of sending information from the top down in a one-to-many mapping. With regard to the second function, lack of a positive acknowledgement by any processor to the instructions given it would sort out at the top, delaying the start of work by all of the other processors. Finally, merging the partial streams of data retrieved by each processor is facilitated by this mechanism, sorting the records into some prespecified order. In fact, it was with this function in mind that the tournament sort approach was adopted in the first place. Later it was determined that it was capable of satisfying the interconnect's other requirements as well.

V. DBC/1012 Subsystems

A. YNET

The actual hardware implementation of the tournament sort algorithm in the DBC/1012 is called the Ynet, a proprietary device described as an "array of active logic" [DBC/ 86] and is named for the shape of the "Y" formed at the nodes of the structure (see Figure 2). The Ynet is a hardware subsystem of the DBC/1012. For clarity of this illustration, only one Ynet is shown. There are in fact two Ynets in the DBC/1012 operating concurrently, sharing the load of communications within the system and acting as a backup for each other in the case of failure. The other hardware components shown in Figure 2 are discussed in greater detail below and are listed, along with the other DBC/1012 subsystems, in the chart in Figure 3.

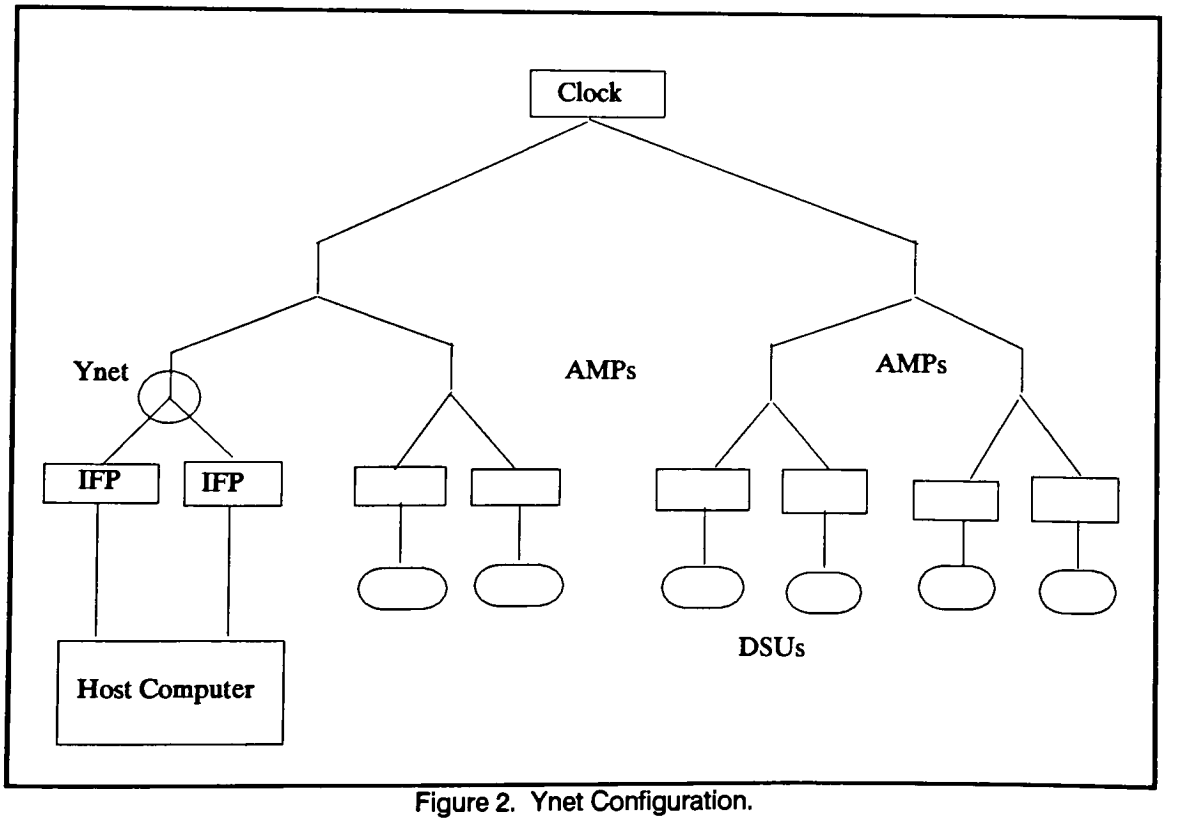

Figure 2. Ynet Configuration.

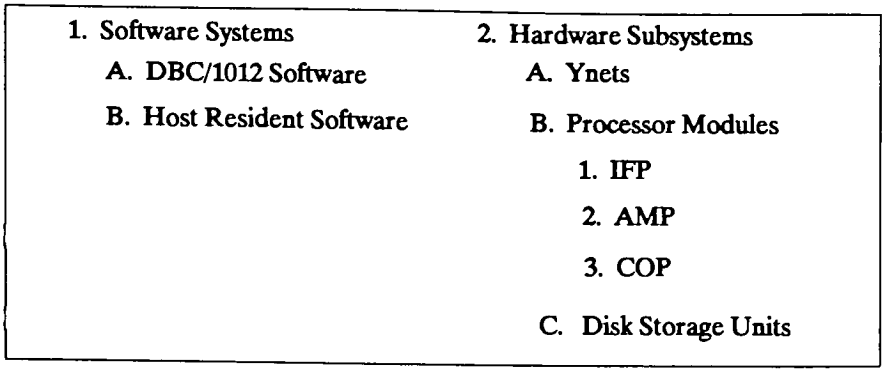

Figure 3. DBC/1012 Systems.

The Ynet supports four basic types of communication between the DBC/1012 system's processors. First, it provides a vehicle for an information exchange between single processors. For example, an instruction step for a simple insert operation that could be processed by a single AMP is generated by one IFP or COP, sorts upward through the "contention logic" of the Ynet to the top. Then the instruction is broadcast down[12] to the appropriate AMP for processing. Conversely, the completion response from the AMP sorts up through the Ynet and is broadcast to the original IFP or COP.

Second, following the same general principles, the Ynet supports communication between a single IFP or COP and all of the AMPs. This promotes the intra-query concurrency of all of the AMPs working to solve one complex query. The resulting communication from this type of processing comprises the third type provided by the Ynet, that of an information exchange from a subset[13] of processors (i.e. AMPs) to a single processor (for example, an IFP). Finally, the Ynet provides a means for communication between various processors in order to synchronize their activities on the data. For example, a primary AMP would resend instructions over the Ynet to a fallback AMP (see below) to maintain a redundant data store.

The basic Ynet configuration is as shown in Figure 2 - seven nodes connecting eight processor modules. This can be expanded by way of a "node expansion module" [DBC/ 86] consisting of three additional nodes connecting either to an original node, or to another node expansion module.

---

12 Acting in its broadcast mode, the Ynet can send information to the specified processor(s) in one clock cycle.

13 Implies possibly all.

B. Host System Communication Interface (HSCI)

Having described how communications within the DBC/1012 is conducted, it is now appropriate to focus attention on the basics of the communication's interface between the database machine and the host computer. The Host System Communication Interface (HSCI), a software subsystem of the DBC/1012 (see Figure 4), enables application software running on the host (user-written programs making CALLS, or the ITEQ or BTEQ facilities) to access the databases located on the DBC/1012. The host must have an established environment of MVS/XA, MVS/SP (Release 1.3 or above), or VM/SP (Release 3 or higher). The flow of information between the two machines is accomplished via a block multiplexor channel[14] connected to an IFP (see Figure 5).

A. DBC/1012 Software

1. Teradata Operating System
   a. Scheduler
   b. Ynet Driver (IFP, AMP)
   c. Host Driver (IFP)
   d. Disk Driver
   e. Message Subsystem
   f. Segment Subsystem
   g. Console Command Interpreter
2. Parser System
3. Data Base System
   a. Session Control (IFP)
   b. Dispatcher (IFP)
   c. Data Base Manager (AMP)

B. Host Resident Software

1. Host System Communication Interface
   a. Call Level Interface library (CLI)
   b. User-to-TDP Communication techniques (UTC)
   c. Teradata Director Program (TDP)
2. ITEQ - Interactive Teradata Query Facility
3. BTEQ - Batch Teradata Query Facility
4. Language Preprocessors
5. Host-Resident Utility Programs

Figure 4. DBC/1012 Software Systems.

The HSCI has three components (refer to Figure 4) all of which reside on the host: The Call Level Interface (CLI), User-to-TDP Communication routines (UTC) and the Teradata Director Program (TDP).

---

14 An IBM-type high speed I/O channel.

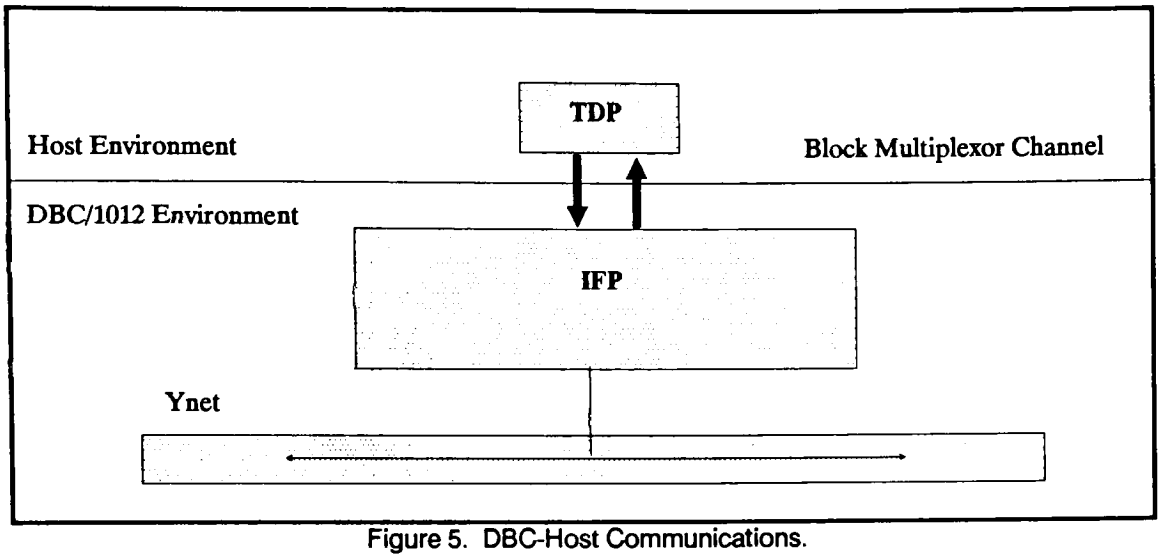

Figure 5. DBC-Host Communications.

The CLI is a collection of routines that can be link-edited[15] with a user program[16] written in any language that has a CALL statement.[17] Application programs access the DBC/1012 databases by means of the CALL statements which are converted into DBC/SQL[18] requests and communicated by the CLI to the TDP. This communication is accomplished by means of a UTC technique.

Several options are available for the implementation of the User-to-TDP-Communcation. In the MVS environment, the UTC can be accomplished by a Supervisor Call (SVC) or by Cross Memory Services (XMS); in VM, the UTC is implemented as an Inter-User Communication Vehicle (IUCV). These are all essentially I/O interrupts [BAER 80] that allow the DBC/SQL instruction to be sent to the TDP whose primary function is to manage communication between the user and the DBC/1012.

The TDP resides in its own region on the host.[19] In MVS, it has its own dedicated address space; in VM, it runs as a virtual machine. As stated above, requests are sent by the CLI to the TDP via the UTC. The TDP creates a message packet (for example, a query instruction step) which is sent over the block multiplexor channel to the attached IFP. From the IFP, the packet makes its journey up the Ynet and is broadcast down to the appropriate AMP(s) for processing. The results from the AMPs are returned up the Ynet, broadcast down to the IFP, sent back over the block multiplexor channel to the TDP which in turn, sends the information via the UTC to the CLI. The CLI returns the information to the user.

---

15 Similar to a User Interface Module in native Datacom programming. See [DATAC 88] for more details.

16 The present discussion focuses on user-written programs. It will be seen that ITEQ and BTEQ interact with the CLI as well.

17 Assembler, PL/1, Pascal, COBOL or FORTRAN. The language C is supported from the COP.

18 Teradata's query language.

19 Similar to ADR's Multi-User Facility. See [DATAC 88] for more details.

C. Processor Modules

The above discusses the basic concepts of intra-DBC/1012 communication, as well as those of the DBC/1012-to-host interface. A more thorough examination of this complex environment requires a detailed look at the remaining components of the DBC/1012. There are three types of processor modules in the DBC/1012 - the Interface Processor (IFP), Access Module Processor (AMP) and Communications Processor (COP).

1. Interface Processor (IFP)

The IFP subsystem handles the dialogue between the host and the DBC/1012. As shown in Figure 6, the IFP has five components: Session Control, Host Interface or Driver, Parser, Dispatcher and the Ynet Interface.

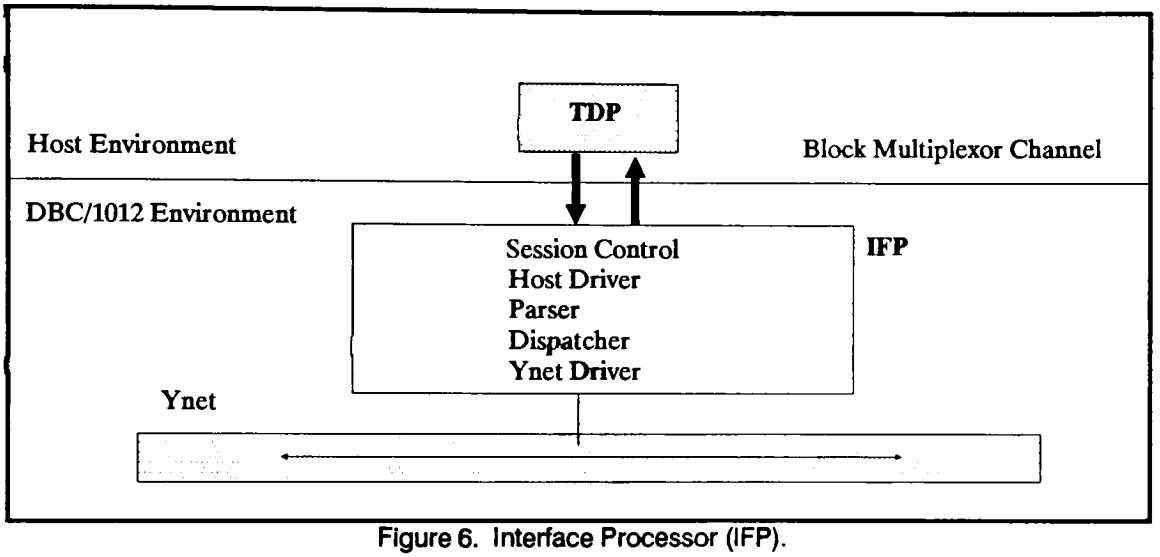

Figure 6. Interface Processor (IFP).

a. *Session Control*

The Session Control module belongs to the Database System, a part of the DBC/1012 Software System (refer to Figure 4). One of these modules resides in each IFP and manages sessions between the DBC/1012 and the users who communicate by means of user-written program or through the ITEQ or BTEQ facilities. A session is the context within which database operations are performed from the host; it is established during the logon process, a major function of the Session Control module. During logon, a user presents textual information (user name, password and accounting information) requesting authorization for a session. The Session Control module verifies this information and either authorizes the establishment of a session, or denies that authorization. Once a session is established, a session number is assigned and is associated with any DBC/SQL statements generated by the user during that session. This session number is also used to identify responses from the DBC/1012.

The other major function of Session Control is the logoff process by means of which a user's session is terminated. Any activity under the session is ended and no new activity is accepted from the user. The session number is released.

b. *Host Driver*

The second component of the IFP subsystem is the interface with the host - the Host Driver, a part of the Teradata Operating System (TOS). This module handles the exchange of message packets between the TDP where they are generated, and the DBC/1012. These packets are transmitted over

the block-multiplexor channel to the I/O processor of the IFP's Processor Board (see Figure 7). Control over this channel is managed by the Host Driver. Message packets returning to the host from the DBC/1012 are generated by the Message Subsystem[20] and are likewise controlled by the Host Driver.

c. *Parser*

The IFP's third component is the Parser System whose primary function is the support of DBC/SQL. The CLI routines convert CALL statements from user-written application programs, or ITEQ or BTEQ commands, into DBC/SQL statements. The Parser System receives these statements in the form of a message packet from the TDP via the Host Driver. Figure 11 depicts the flow of information through the DBC/1012's various subsystems and their components. The Parser checks the message for syntax and evaluates its semantic composition. The Parser System then refers to the Data Dictionary/Directory[21] where it can resolve the symbolic data names found in the message. The message packet is transformed into a series of low-level instruction steps which the Parser sends to the Dispatcher Subsystem. These steps are subsequently used by the Data Base Manager of the Access Module Processor (see below) in its data manipulation operations.

d. Dispatcher

The Dispatcher Subsystem is the IFP's fourth component and belongs to the Data Base System of the DBC/1012 Software category, along with the Session Control and the Data Base Manager (refer to Figure 4). This component controls the execution sequence of the instruction steps. It handles two types of tasks in the IFP - execution control tasks and response control tasks. The first receives the low-level instruction steps generated by the Parser System and transmits them via the Ynet Driver (discussed below) to the appropriate AMP(s) for processing. As mentioned earlier, the processing of each AMP that is associated with the task must be complete before the next set of instructions is sent to it.[22] Responses regarding the completion status of each processor are sorted up over the Ynet with the "winning" response being broadcast down to the originating IFP. Here the Dispatcher Subsystem's execution control task receives the reponse over the Ynet Driver and sends another instruction step if the AMP(s) have finished processing the first set.

Once all of the instruction steps have been processed, the execution control task interfaces with the response control task. This task receives the results from the AMP. This may be for example, a completion indicator from an UPDATE request, or the actual data returned in sorted order over the Ynet. These results are sent to the user by the response control task.

e. *Ynet Driver*

The final IFP component is the Ynet Interface, or Ynet Driver, another part of the TOS. As indicated above, the Ynet Driver manages the flow of instruction steps over the Ynet to and from the AMPs. At a hardware level (see Figure 7), an instruction step is moved from the IFP's main memory, into the Ynet Interface module's high-speed buffer memory. AMP responses are transferred from the buffer to the IFP's main memory.

---

20 Another module of the TOS.

21 A collection of system tables containing current database definitions, as well as general information about each database (eg. table names, authorized users and access rights).

22 Release 3.0 of the Teradata Software however, supports parallel instruction step processing. There are circumstances under which certain steps may be processed simultaneously when the execution of one has no affect on the other. See further discussion below.

Figure 7 represents an IFP's hardware configuration. It contains four circuit boards: a Processor Board, a Memory Board and one Board for each of the Ynets [NECH 86]. The Processor Board is comprised of a central processing unit (an Intel 80286 microprocessor), a numeric processing unit (an Intel 80287 arithmetic co-processor) and an I/O processing unit (and Intel 8089 I/O processor). The latter controls the interface to the block-multiplexor channel.

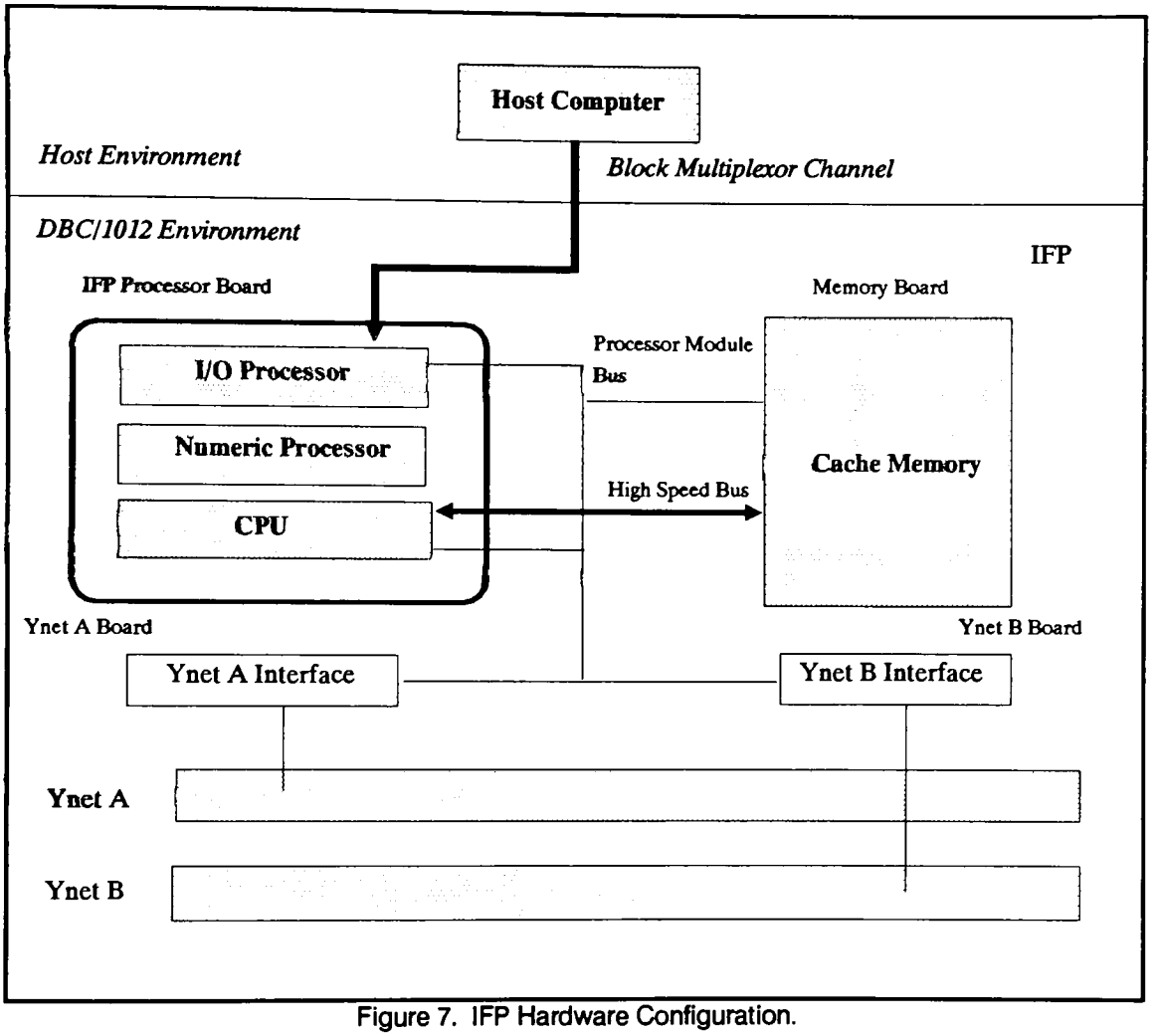

Figure 7. IFP Hardware Configuration.

2. Access Module Processor (AMP)

The next Processor Module in the DBC/1012 is the Access Module Processor (AMP). In response to instruction steps sent to it over the Ynet by an IFP or a COP, the AMP manages the actual data manipulation for the DBC/1012 system. Its primary functions are the retrieval and storage of data on the Disk Storage Unit (DSU) with which it is associated (see below). As data is placed into DBC/1012 databases, it is evenly distributed across all of the AMPs and their DSUs. This distribution is based on a hashing algorithm which uses the primary index field(s) of the rows. AMPs have three major software components: Ynet Interface, Data Base Manager and Disk Interface (refer to Figures 4 and 8).

a. *Ynet Driver*

The Ynet Interface is the Ynet Driver module of the TOS. Note that this same module is a component of the IFP discussed above. The TOS resides in each processor module of the DBC/1012 - IFPs and AMPs alike. The scope of this operating system is limited to the individual processor which it controls. The Ynet Driver, functioning in the AMP, receives instruction steps from the IFP by moving the message packet out of the high-speed buffer memory of the Ynet Interface and over the processor module bus into its main memory (see Figure 10). Responses from the AMP's data manipulation operations are conversely moved from the processors' main memory to the high-speed buffer where they are returned over the Ynet to the IFP.

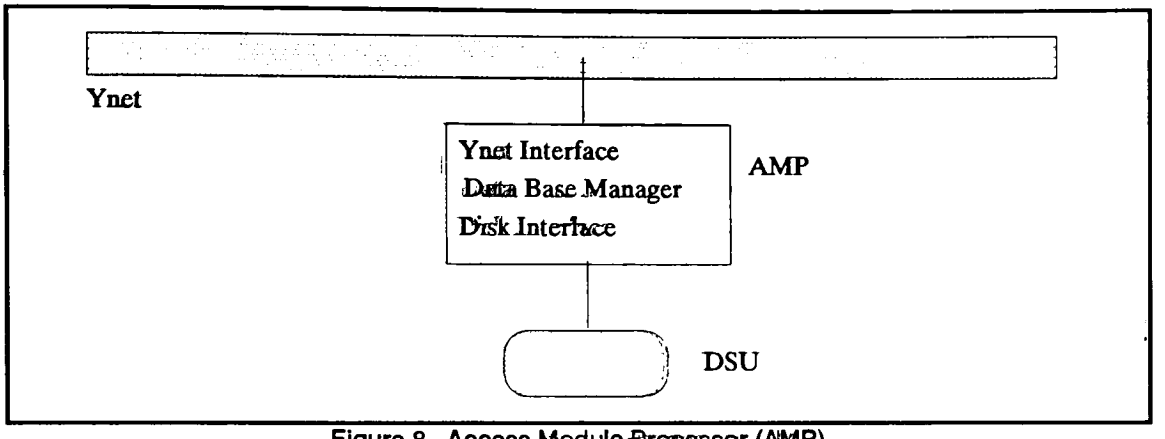

Figure 8. Access Module Processor (AMP).

b. Data Base Manager

The Data Base Manager is the second component of the AMP. It, like the IFP's Session Control and Dispatcher, is a part of the DBC/1012 Software category of Database System software (refer to Figure 4). The Data Base Manager can be considered the work horse of the DBC/1012. Here the critical functions of manipulating the actual data are performed; many steps are involved to ensure the integrity of the databases.

The data store associated with an individual AMP is physically located on that processor's DSU (see below). One of the primary functions of the Data Base Manager is to resolve the disk address of requested data. Rows of data from the same database table are stored in logical blocks, groupings of physical disk sectors. These sectors are read from and written to the disk as a single unit to improve the efficiency of retrieval. Large tables may occupy more than one block. As a row is added to a database table, it is assigned a unique row identifier for that table. The tables themselves also have identifiers. Every data block contains control information about the table whose rows are stored in that block, as well as the identifier for the block's first row.

To accomplish the transition from the data's logical organization on blocks to that of its actual physical location, the Data Base Manager utilizes two levels of indexes (see Figure 9). The first index is the Master Index for the cylinders of the AMP's associated DSU. This index contains a Used Cylinder Descriptor List (UCDList) on which there is an entry for every cylinder containing data. An entry is comprised of the identifier of a cylinder, the identifier of the first row stored on that cylinder and identifier of the table to which that row belongs. The UCDList is sorted first by cylinder, then in ascending order by Table ID, and Row ID. A binary search of this index enables the Data Base Manager to locate the entry for the cylinder on which a row is stored.

Each cylinder in turn has a Cylinder Index which contains a Data Block Descriptor List (DBList). This list contains information about each block located on that cylinder. Recall that a block contains rows

from one table and that one table may be large enough to occupy more than one block. Entries in the DBList include the identifier of the table located on a block, the row identifier for the first row on the block, the starting address of the block on the disk, and the size of the block in bytes. The DBList is sorted in ascending order by table identifier and row identifier of the first row. A binary search of the cylinder index locates the block on which a row is located. That block is read and sequentially searched to locate a particular row.

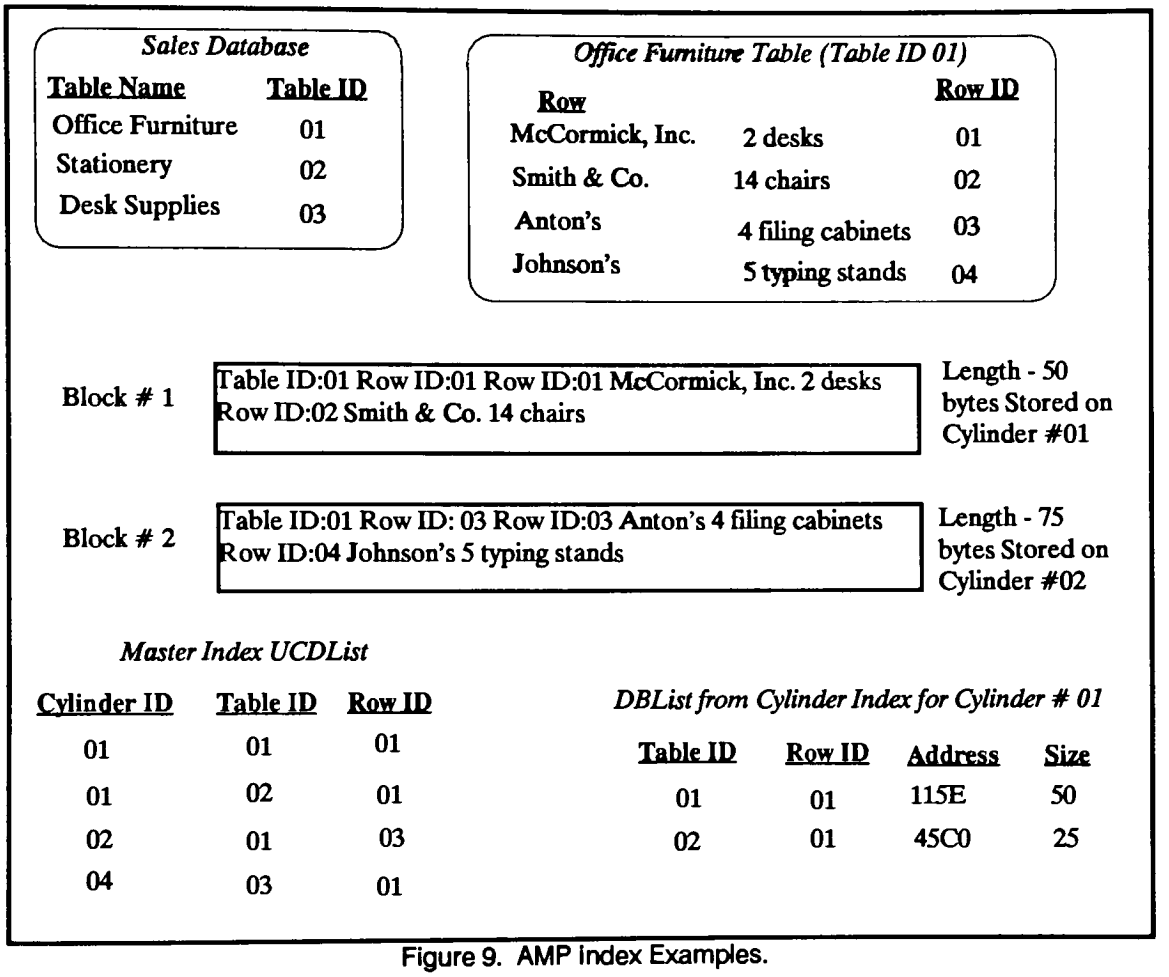

Figure 9. AMP Index Examples.

c. Disk Driver

Once the appropriate data block has been located, the AMP processes the instruction steps which were sent to it. This is done by passing commands to its DSU via the disk interface, the Disk Driver. This is the third component of the AMP and is a module in the TOS. Responses from the DSU are returned to the AMP by the Disk Driver and ultimately sent back to the IFP's Dispatcher (see Figure 11). As seen in Figure 10, the hardware configuration of the AMP is the same as that of the IFP, except that the I/O Processor Board connects to the DSU from the AMP.

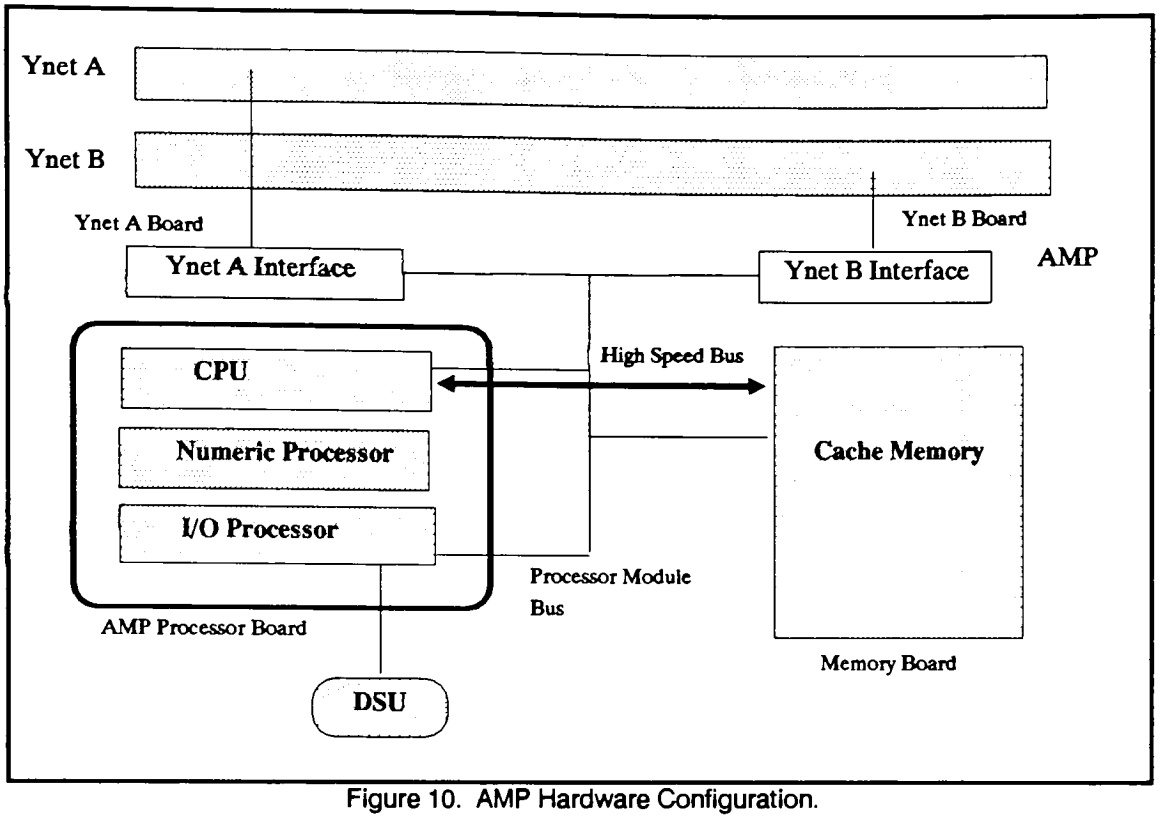

Figure 10. AMP Hardware Configuration.

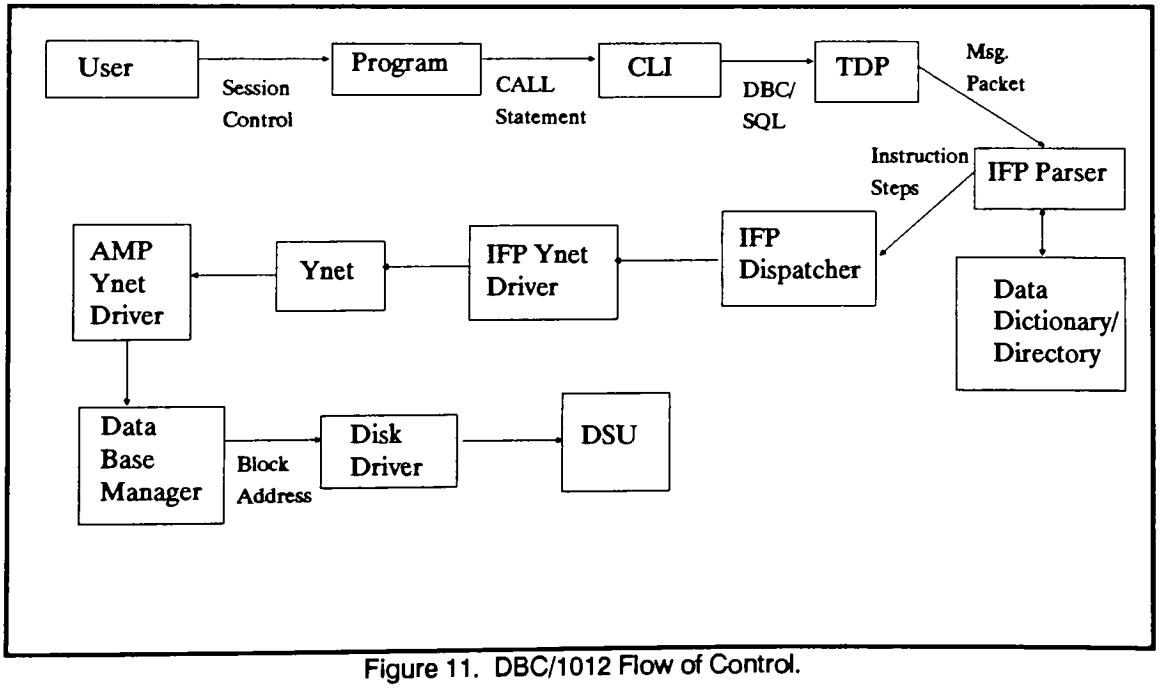

Figure 11. DBC/1012 Flow of Control.

3. Communications Processor (COP).

The most recent addition to the DBC/1012 is the third Processor Module subsystem, the Communications Processor (COP) [TERA 87A]. The COP is a modified IFP which allows user access to the

DBC/1012 from an Ethernet Local Area Network (LAN).[23] The same functions that the IFP performs on behalf of the host, the COP performs for the LAN.[24] These functions include session management, parsing and dispatch control. While the IFP receives requests from the host over the block multiplexor channel, the COP receives requests as messages sent over the LAN according to a given network protocol. Currently the ISO/OSI (International Standards Organization/Open Systems Interconnect) and TCP/IP (Transmission Control Protocol/International Protocol) protocols are supported: more will be available in future releases [TERA 88a]. Figure 12 shows how the COP fits into the DBC/1012 system. One LAN may be connected to more than one DBC/1012; a DBC/1012 may be attached to more than one LAN.

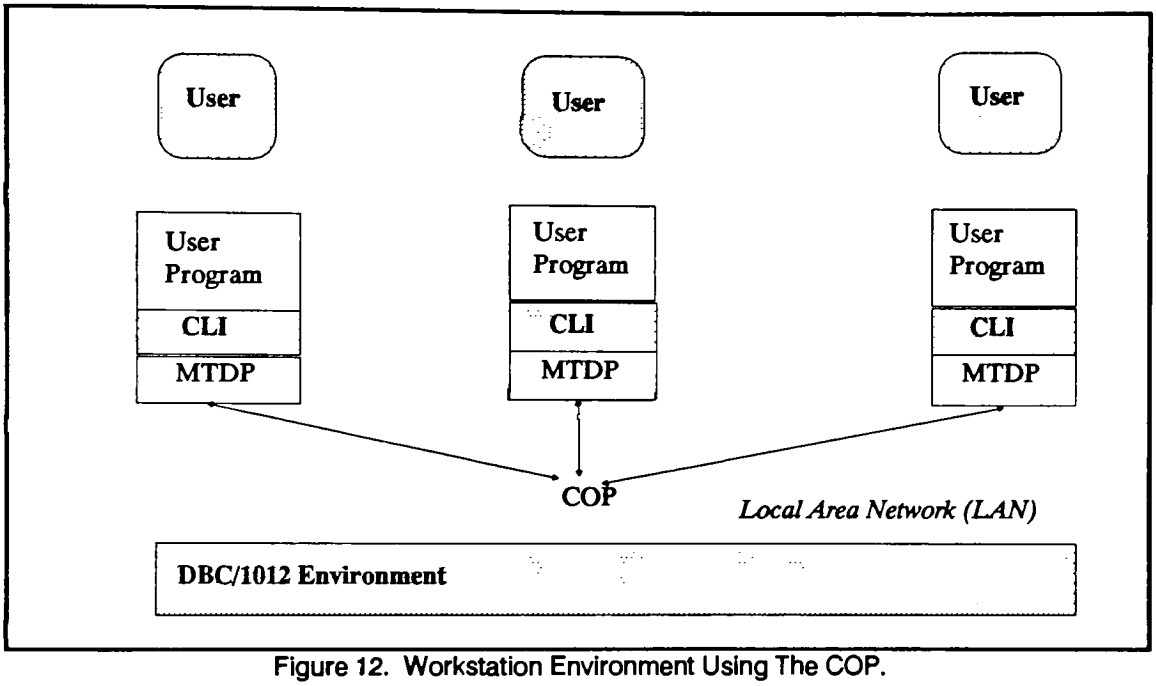

Figure 12. Workstation Environment Using The COP.

Some host-type software is necessary for each LAN accessing a DBC/1012 via a COP. Call Level Interface (CLI) Version 2 is required to manage the formatting of messages sent to the COP over the LAN. The CLI calls the Micro Teradata Director Program (MTDP) to manage communications with the DBC/1012. This is the same sort of interaction seen between the CLI and the TDP in a host-to-DBC/1012 system. Operating system-dependent and communications protocol-dependent service routines are isolated in the MTDP into a library called the Micro Operating System Interface (MOSI). Functions requested of the MTDP that could be handled by the LAN's operating system's communica-

23 More types of networks will be supported in the future.

24 In effect, the LAN is a sort of host to the DBC/1012.

tions protocol are mapped onto calls to those systems by the MOSI. Presently, COP support is provided for LAN environments based on IBM PC's or PC-compatibles running MS-DOS by Microsoft, Inc. or IBM's PC-DOS (version 3.1 or higher), as well as AT&T's 3B2 minicomputers models 300 and 400, running UNIX version 5.2.

4. Disk Storage Unit (DSU)

The Disk Storage Unit (DSU) comprises another major subsystem of the DBC/1012. Support is provided for several disk models, varying in storage capacity and speed. One compatible DSU is the Fujitsu 474 MB[25] Winchester-type random access disk drive. Its average seek time[26] is 18 milliseconds;[27] its average transfer rate[28] is 1.86 MB per second. Another device is a 515 MB Control Data Corporation (CDC) Winchester-type disk with an average seek time of 20 milliseconds and an average data transfer rate of 1.88 MB per second.

Each DSU can be conceptually partioned into three areas: a system area, a primary copy area and a fallback copy area (see Figure 13). The first stores system programs and system tables (for example, the previously mentioned Data Dictionary/Directory).

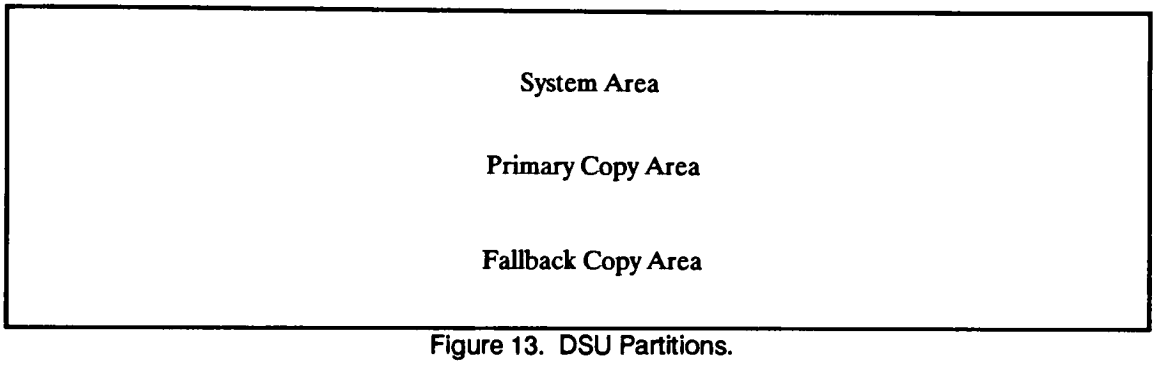

Figure 13. DSU Partitions.

The primary copy area contains that instance of the data which is accessed under normal conditions. The DBC/1012 provides a facility for redundant data storage. Control of the primary copy of a row of data is assigned to a particular AMP; responsibilty for the second or "fallback" copy is given to a different AMP. That way, the copy is always located on a physically separate DSU from the original. If redundant data storage is specified,[29] the primary AMP, after completing a request for the insertion, deletion or modification of data, forwards that request to the fallback AMP. Here the same processing is completed on the copy. If the primary AMP is inoperative for some reason, the request is sent to the fallback AMP directly. As soon as the primary AMP becomes available again, all of the

---

25 Megabyte. One megabyte is one million bytes.

26 Time required to position the read/write head on the appropriate cylinder.

27 A millisecond is one-millionth of a second.

28 Time required to read or write a sector.

29 The DBC/1012 defaults to no creations of a fallback copy. This option must be specified at the time of the database's definition.

update information is collected from the fallback AMP and is processsed in the primary AMP's DSU. That AMP then resumes its primary role.

5. Other Host Resident Components

As previously mentioned, users can communicate with the DBC/1012 by means of programs that include CALL statements. The CLI converts those statements into DBC/SQL format. There are other facilities that allow users to communicate with the DBC/1012 in addition to CALLs from application programs.

a. Language Preprocessor

Another component of the DBC/1012 host resident software are the Language Preprocessors which provide users with the ability to embed DBC/SQL commands directly into their source code. Preprocessors are provided to support PL/1 and COBOL. They run as batch jobs in the host machine.[30] As an application program is preprocessed, it goes through several phases.

In the first phase, the source program is scanned for embedded DBC/SQL statements which are prefixed with a delimiter, usually a question mark ("?"). Next, expansions are made of any macros[31] into their component parts. The DBC/SQL statements are then replaced with CALLs to CLI routines that handle the actual interface between the program and the DBC/1012. The final phase is to compile the preprocessed source code. To form an executable load module, the user can link-edit the compiled, preprocessed source code with the CLI routines. Alternately, the CLI routines may be dynamically loaded at runtime.

b. Interactive Teradata Query (ITEQ)

The Interactive Teradata Query facility (ITEQ) is another software component allowing users to communicate with the DBC/1012. This is very similar to the online Dataquery facility from ADR [DATAQ 88]. The ITEQ software resides in the host, running either under the MVS Time Sharing Option (TSO) or VM's Conversational Monitoring System (CMS). Online users of ITEQ can enter DBC/SQL statements or macros directly to the DBC/1012 from a 3270 terminal or a 3270-compatible display device connected to the host. ITEQ makes calls to the CLI routines which in turn, interact with the DBC/1012 just as they do when CALL statements or embedded DBC/SQL statements are issued from user programs. Users have additional capabilities with ITEQ such as defining program function (PF) keys on their individual keyboards and requesting specific formats for output received from a query. Results from accessing the DBC/1012 can either be sent to the user's screen or output to a printed report.

c. Batch Teradata Query (BTEQ)

The final host resident software component providing special communication to the DBC/1012 is the Batch Teradata Query facility (BTEQ). There is a similarity here to ADR's product, Dataquery Batch [DATAQ 88]. This facility provides users with the ability to execute a series of DBC/SQL statements or macros in a batch or background mode. Productivity for the user is enhanced because multiple sessions with the DBC/1012 can be established for a single user. For example, if a user had two BTEQ jobs running in batch, a third session from ITEQ could be maintained at the same time. BTEQ also calls CLI routines which manage the communications with the DBC/1012.

---

30 Much like a compiler or an assembler does.

31 A DBC/SQL macro is a series of statements executed in sequence. It is defined using the DBC/SQL CREATE MACRO.

There are several advantages to using the BTEQ facility as opposed to ITEQ. First, BTEQ has the added feature of more sophisticated report writing capabilities than does ITEQ. Also, BTEQ is available to workstations connected to a COP. ITEQ is presently not available on workstations. Finally, by means of BTEQ data can be read from and written to non-DBC/1012 datasets that are maintained by the host.

VI. Database Integrity

Of prime importance in database processing is ensuring data integrity. The DBC/1012 has four major facilities for maintaining the integrity of the data stored in its databases (see Figure 14).

DBC/1012 Database Integrity	
1. Transaction Processing	3. Journaling
2. Locking Mechanisms	4. Redundant Data Storage

Figure 14. Data Integrity.

A. Transaction Processing

The first is the utilization of the concept of transaction processing. In a transaction, one or more DBC/SQL statements are executed in a sequence and behave as one single unit. This concept is similar to that of macros, discussed above. Data integrity is maintained by the use of transactions because the DBC/1012 ensures that all parts of a given transaction are performed completely, or else no part is.

DBC/1012 transactions may be implicit or explicit. Macros are examples of implicit transactions. The DBC/SQL statements BEGIN TRANSACTION and END TRANSACTION set apart user-defined, explicit transactions. Any sequence of DBC/SQL statements or macros may be inserted between the BEGIN and END statements. If any part of a transaction, implicit or explicit, cannot be performed to completion, changes made to the database are reversed or "backed out" and any locks placed on the data affected by the transaction are released.

B. Locking Mechanisms

These actions constitute two other data integrity facilities of the DBC/1012, locking mechanisms and journaling. Locking mechanisms are used to control concurrent access to the same data by different users. If one user holds a write lock (explained below) on some data, that user is guaranteed that the data will not be changed by any other user while the lock is in place. Concomitantly, the requests from other users to update that data will wait until the first user's lock is released.

There are three levels of locks; they can be placed on an entire database, on certain tables within a database, or on rows within a given table. Locks are acquired automatically by the DBC/1012 during transaction processing. They are released either upon completion of the transaction, or after a transaction abends and the changes made to the data backed out. Users may specify locks also. This is sometimes done to ensure a more restrictive lock than the one the DBC/1012 would put in place automatically.

Four types of locks exist in DBC/1012 processing: exclusive, write, read and access. The exclusive lock prevents all access to the locked data by users other than the one requesting the lock. This is the most restrictive type of lock and may be acquired only at the database or table levels. Users sel-

dom find the need for exclusive locking mechanisms; they are usually only necessary when structural changes are being made to a database or table by the DBA.

A write lock will give the user placing it the ability to modify the data affected. This type of lock may be placed at any of the three levels - database, table, or row. When a write lock is in effect, only those other users who are reading the data and are not concerned with the consistency of that data are allowed to access it. No one else may make modifications to and no new read locks (described below) may be placed on the data until the write lock is released.

There are times when users might be concerned that the data they are reading not be altered during their read operation. This is especially true, for example, when executing DBC/SQL SELECT statements. A read lock ensures the consistency of the data being read. Several users may each hold read locks on the same data, simultaneously. No modifications may be made to the data (that is, no exclusive or write locks may be acquired) until the read locks are released.

Access locks constitute the least restrictive level. Such a mechansim was designed for users who are not concerned if updates are made to the data they are reading. Prompt access to the data is more likely if the access lock is used in these situations, rather than a read lock which may be forced to wait until an active write lock is released. An access lock would not be made to wait for any other more restrictive lock, except an exclusive lock.

C. Journaling

Journaling is the third DBC/1012 facility for ensuring data integrity. There are two types of journaling - transient and permanent.

1. Transient Journals

Transient journaling is, as its name implies, temporary in nature and plays a major function in transaction processing. It is the transient journal which is used to back out changes made by transactions that did not complete successfully. Entries on this journal are comprised of the transaction identifier and sequential "before images"[32] of the data. If a transaction abends, the transient journal entries are used to restore the data according to the before images.

2. Permanent Journals

Permanent journals are created to assist in forward and backward database recovery. This type of journaling is not automatic, as in transient journaling. A permanent journal must be specifically requested for a particular database at the time it is defined. There may be only one permanent journal per database. Entries on this journal however, may reflect changes made to one or more tables in that database. It is also possible to have the changes made to table(s) in one database be reflected in the permanent journal of another database.

There are several available options for the information recorded as entries on a permanent journal. The images kept of the data may be either before images or images of the data after modifications have been made - "after images". Both before and after images may be journaled. In addition, one copy of the images may be kept - "single logging" - or two copies - "dual logging".

---

32 A copy of each piece of the data before modifications to it were made by the transaction.

During forward recovery, the permanent journal is used to restore modifications made to a database after a certain point in time. If for example, data in a database were destroyed, a previously made backup copy of that data could be used to recover the lost transaction processing. Permanent journal entries would be used to recreate, on the copy of the database, the modifications made to it up to the point of the data's destruction.

Likewise, in backward recovery, permanent journal entries would be used to reverse the modifications made against a database, returning it to some previous state. To facilitate the recovery process, a function called "checkpointing" may be specified. Using this feature, a user acknowledges the processing of data completed up to a certain point, a checkpoint. If recovery were necessary, it could be performed forward from, or backward to, a particular checkpoint, instead of making use of the entire permanent journal. The user is theoretically aware of the state of the database at the time the checkpoint is issued and can function with it, accordingly.

D. Redundant Data Storage

The fourth facility for maintaining data integrity on the DBC/1012 is the redundant data storage managed by the AMPs and discussed in detail above. After a primary AMP has been restored from failure, it makes use of a type of transient journal kept by its fallback AMP. This journal allows the primary AMP to make the updates necessary to conform its data to that of the secondary AMP before resuming its role as primary AMP.

VII. Benchmarks

Benchmark studies have recently been conducted by four major Teradata customers: Citibank, N.A. [TERA 86], Liberty Mutual [BAZE 86], Bank of America [TO 86] and The Chicago Board Options Exchange [REIN 87]. Testing of the DBC/1012 in the first three evaluations was performed at Teradata with specific workload instructions provided by the companies involved. The fourth was conducted on site by Chicago Board Options Exchange (CBOE) personnel. The quality of information presented in these studies varies from vague to explicit, the Citibank, N.A. (Citibank) benchmark proving the most complete. Bank of America (B of A) would supply no raw data, and yet the descriptions of their testing process and their analysis of the results are extremely useful. Liberty Mutual (Liberty) presented a series of graphs and charts which proved lacking in some detail; again, no data per se was supplied, just the analytical results. It is certain that other benchmarking activities against the DBC/1012 have been made by companies who, for internal security considerations, are unwilling to release their findings to the public; the four studies discussed here are the only ones made readily available.

A. Liberty Mutual

Liberty had traditionally been an IMS shop. It quickly realized the benefits of relational database strategies and sought alternatives to its somewhat rigid hierarchical database schemes. As a short term solution, Liberty was willing to enhance the accessiblity of its data resources by means of a dual database approach, maintaining data in both IMS and DB2 databases.[33] DB2 would provide a more flexible vehicle for supporting the ad hoc informational needs of Liberty's management staff. A long

33 See [HESS 84] for a more complete discussion of IBM's dual database strategy.

term solution was required however, one that would involve a single data resource and be readily accessible for business analysts' use.

A performance evaluation study was devised and conducted in 1986, comparing the software relational database management system of DB2 against the DBC/1012. DB2 Release 1 was tested, running under the MVS/XA operating system on an IBM 3081G processor, an 11.4 MIPS machine. A Model 2 DBC/1012[34] utilizing the Intel 80286 processors was tested. Liberty describes this machine only as a 12.0 MIPS system; this was probably a 4 x 8 configuration.

The testing was conducted against three databases containing 880 MB of data in a total of nineteen tables. These tables ranged in size from 150 KB[35] to 350 MB. Tables were categorized as small (less than 15 MB), medium (16-75 MB) and large (over 76 MB). As a workload, Liberty provided fourteen queries described as simple, medium, and complex. The benchmark findings were based on the eight combinations of query category and table size shown in Figure 15.[36]

	QUERY TYPE	
ID	QUERY CATEGORY	TABLE SIZE
SS	Simple	Small
SM	Simple	Medium
SL	Simple	Large
MS	Medium	Small
MM	Medium	Medium
ML	Medium	Large
CS	Complex	Small
CM	Complex	Medium

Figure 15. Query Combinations in Liberty Benchmark.

Liberty's main purpose in this benchmark was to determine how well their workload was processed in either of the two environments. Their major performance indexes fall in the responsiveness category. The first index is response time, measured for the eight test combinations running on both busy and quiet hosts. The second index measured response time based on the execution of multiple identical queries processing against the same database. These were measured for DB2 and for the DBC/1012.

Figure 16 shows the results of measuring the first performance index. The response times shown are estimates based on graphic information supplied in [BAZE 86]. Actual measurements were not made available. From this chart we can see that average response times for the DB2 transactions were adversely affected by other activity on the host. This is significant since virtually no host in a

---

34 i.e. running Teradata software Release 2.0.

35 Kilobytes. A kilobyte is one thousand bytes.

36 It is unknown why the combination of complex query and large table was not reported.

production environment can be considered quiet. The medium table/simple complexity query (SM) test combination was most severely inhibited, showing a response time nearly six times longer on a busy host than on a quiet one. On the average, DB2 response times were over twice as long on the busy host.

DB2			DBC/1012	
Quiet	Busy	Query ID	Quiet	Busy
5	10	SS	17	17.5
4	23	SM	9	9
1	2	SL	4	4
3	7	MS	4.5	4.5
8	25	MM	16	16.5
14	25	ML	8	8
8	15	CS	3	3
60	115	CM	37	37

Figure 16. Liberty Benchmark Response Time in Minutes.

The response times for the test combinations run on the DBC/1012 on the other hand, demonstrate that activity on the host had no significant affect on their performance. Moreover, the average response time for the DBC/1012 queries submitted from a busy host was slightly less than the average for the DB2 queries run on a quiet host. Is should be noted however, that the overall response time for the SS test combination on the DBC/1012 was nearly twice as high compared to the same combination run with DB2 on a busy host; about 3 1/2 times that run on the quiet host with DB2. Also, the response time for the DBC/1012 test combinations do not show improvement over those for the quiet host DB2 tests until we reach the ML combination.

The DBC/1012 tests do in general show improvement over the busy host combinations beginning with the SM test. This is seen to be more meaningful since a database application would most likely run in an environment where the host could be considered busy. The three most complex combinations on the DBC/1012 - ML, CS, and CM - showed from three to five times faster response times than the same tests run on a busy DB2 host.

From this information, Liberty concluded several things. First, it was shown that both DB2 and the DBC/1012 have the capability to perform queries ranging from simple to complex on database tables ranging in size from small to large. Second, DB2 performed its best with the simpler queries run against small tables. Third, the DBC/1012 was seen to perform best with complex queries against large database tables. Liberty viewed the latter as an important consideration, given their expectations that users, becoming more familiar with query techniques against relational databases, would come to execute more and more complex queries against constantly growing databases.

The second performance index measured by Liberty was that of response time for multiple concurrent executions of identical queries against the same database. In runs of the eight test combinations, submitted by from one to ten concurrent users, both the DBC/1012 and DB2 exhibited response times that were inversely proportional to the number of concurrent users. As the number of concurrent users increased, the response times degraded in a linear fashion. In general however, response

or each category of nu

base management implementa-
achines used to run them. Figure
s on the DBC/1012 and those using
/1012 were much less expensive than
n the CM test where the DB2 query cost
uted on the DBC/1012.

Figure 17. Liberty's Per Query Costs in Dollars.

g the 1986 purchase prices of the computers themselves, Liberty noted that the IBM
ocessor[38] cost nearly $5 million, while the 4 x 8 DBC/1012 configuration on which the
arks were run cost slightly over $.5 million. The 12 x 24 DBC/1012 had a purchase price of
1.5 million. Based on these cost considerations and on the results of the performance evalua-
s, Liberty concluded that the DBC/1012 was the better alternative over DB2.

B. Citibank N. A.

The Citibank evaluation study was far more comprehensive than that conducted by Liberty. It did not compare the performance of the DBC/1012 against a software database management system running on a mainframe, as Liberty's did. Rather, it compared three different DBC/1012 configura-

37 No cost was supplied for the CS test on the DBC/1012.

38 An upgrade option from the 3081G, increasing MIPS from 11.4 to 14.46 [WEIN 87].

tions against each other. The primary objective of this benchmarking was to determine whether system throughput would increase in direct proportion (i.e. linearly with) the size of the system itself.

While Citibank is reticent to discuss the specific details of its data processing requirements, it has been ascertained [KULL 86] that the project which the DBC/1012 is slated to support consists of two phases. The first involves centralized information regarding Citibank customers from several of its banking ventures. The goal of the second phase is to build a network in which tens of thousands of terminals would have access to a single data store on the DBC/1012. IBM's Transaction Processing Facility (TPF)[39] would interface with the DBC/1012. The Citibank production configuration consists of 168 processors (32 x 136) with 136 Winchester DSUs with 515 MB of storage capacity each.[40]

The three configurations tested in the Citibank benchmark were 8 x 32, 16 x 64 and 24 x 104. Each was a Model 2, running Release 2.0 of the Teradata system software. The DBC/1012s were driven in each test by an Amdahl 5840 8 MIPS mainframe[41] with 16 MB of main memory. The data was supplied by several Citibank sites.

Four separate workloads were run on each of the three machines. The first was a set of queries[42] comprised of insurance related requests. Next was a sales history analysis suite, followed by a typical banking application. Finally, a script of file loading utilities was run in an effort to examine the system's behavior under conditions of data movement in high volumes.

1. Insurance Application

The insurance scenario was one designed to subject the DBC/1012 to a highly complex query environment. The query suite consisted of nine complex queries which executed joins, groupings, aggregations and sorts (see [DATE 86]) against large tables. The two tables used in this testing were the CLAIM table and the STATUS table. The first contained 5.4 million rows of 131 bytes and twenty four columns per row.[43] As in every DBC/1012 database table, there was a primary key field [DBC/86]; this table also had four non-unique secondary key fields. The second table was comprised of 15.4 million, fifty-eight byte long rows with fifteen columns per row and no secondary key fields.

Figure 18 shows the results of measuring three separate performance indices. The first was a productivity index measuring the number of queries processed per minute. In this test, three query suites were run simultaneously. The number of queries processed increased virtually linearly with the number of AMPs. The 8 x 32 configuration processed .4 queries per minute; the 16 x 64, .8; and the 24x104, 1.2 queries per minute.

The second index, also a productivity index, measured the number of rows processed per second in a full file sequential scan of the CLAIM table. The results are shown in the second chart of Figure 18.

---

39 A high speed online transaction processing system evolved from that company's Airline Reservation System.

40 Over 70 gigabytes (GB) or seven hundred billion bytes, total. A gigabyte is 10^9 bytes.

41 Comparable in capacity to an IBM 3083-J.

42 A "query suite".

43 Nearly 1 GB of data.

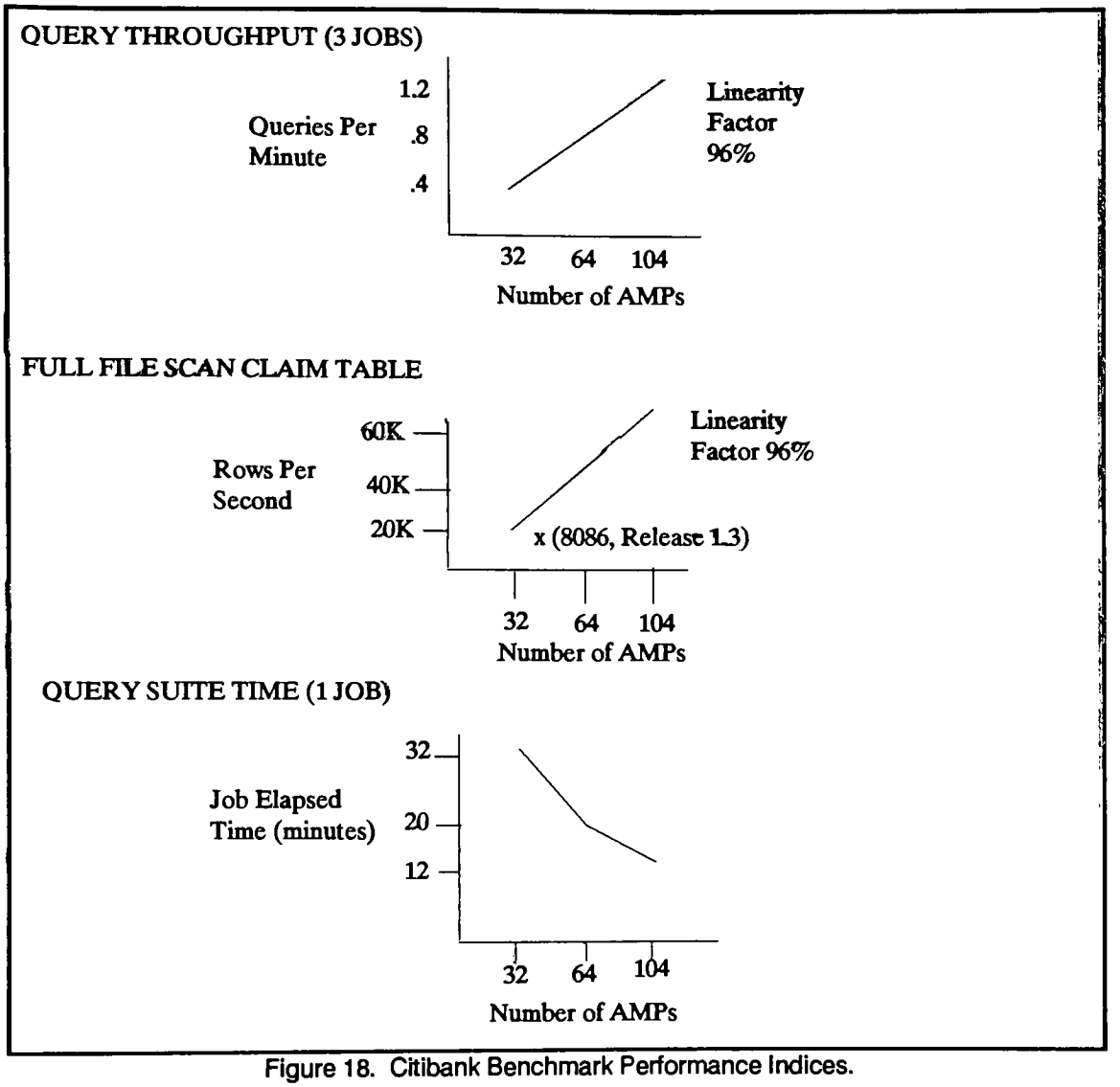

Figure 18. Citibank Benchmark Performance Indices.

Again, the linear throughput increase is verified in this test; increasing the number of processors by a factor of 3.25 resulted in 3.4 times greater throughput. The 24 x 104 machine processed 68,000 rows per second. It was estimated [KULL 86] that Citibank's 168 processor configuration could process 100,000 rows per second. Noted on this second chart is a previous benchmark of the same operation run on a Model 1 DBC/1012[44] running Release 1.3 of the Teradata system software. Significant performance improvements were realized with the combination of the 80286 processors and the revised Teradata software.

The final performance index in the insurance application was a responsiveness index which measured elapsed time of the query suite processing. The third chart in Figure 18 shows that response time

44 Model 2 uses the Intel 80286 processors; Model 1, the 8086.

improved as the number of AMPs increased, but that this improvement did not demonstrate the same linearity factor as did the previous two measurements. Teradata attributes this to the presence of certain serial operations, for example IFP Parser activities and overhead functions whose inherent elapsed time is of a fixed nature. As response time for query processing improves with increases in system size, these fixed times comprise a greater percentage of the total.

2. Sales History Analysis

The second test scenario involved a sales history analysis application which was designed to examine the system as it processed against a large number of tables. The two primary tables were the CUSTOMER/PRODUCT table comprised of 467,000 rows of seventy-eight bytes and twenty-two columns each; and the CUSTOMER/GEOGRAPHY table of 50,000 rows, 111 bytes long with fourteen columns. Fifteen other, smaller tables were also included.

There were eight queries in the suite ranging from simple to complex. Of these eight, the three most complex queries were chosen to stress the DBC/1012 and to measure a responsiveness performance index of elapsed time. Figure 19 shows the degree of comlexity of these queries. The first and third accessed that data through three views; query B used two. Processing of queries A, B and C involved joins of four, three and two tables, respectively. The third major contributor to the complexity of these queries was the combination of Boolean factors in the "WHERE" clauses. The selection criteria involved seven such factors in queries A and C, eight in query B.

QUERY	VIEWS INVOKED	TABLES JOINED	COLUMNS SUMMARIZED	BOOLEAN FACTORS	COLUMNS IN OUTPUT SORTING
A	3	4	0	7	2
B	2	3	4	8	10
C	3	2	2	7	1

Figure 19. Query Complexity.

Figure 20 gives the results of the query suite run on the configurations in this benchmark compared with the results from the same tests run on three Model 1 configurations of 4, 20 and 40 AMPs. Figure 21 depicts a graphic representation of the data. Query A, as processed on the 104 AMP Model 2, ran 229 times faster than on the 4 AMP Model 1. A fairer comparison is made however, between the 32 AMP Model 2 and the 40 AMP Model 1. Here, query A ran 20 times faster on the Model 2 than on the older model; query B, almost twice as fast; and query C, over three times faster.

	MODEL 1, RELEASE 1.3			MODEL 2, RELEASE 2.0		
QUERY	4 AMPS	20 AMPS	40 AMPS	32 AMPS	64 AMPS	104 AMPS
A	41:16	7:59	4:19	:12	N/A	:11
B	28:23	5:53	3:12	1:42	1:07	:40
C	12:46	2:50	1:30	:27	:33	:27

Figure 20. Response Times (minutes).

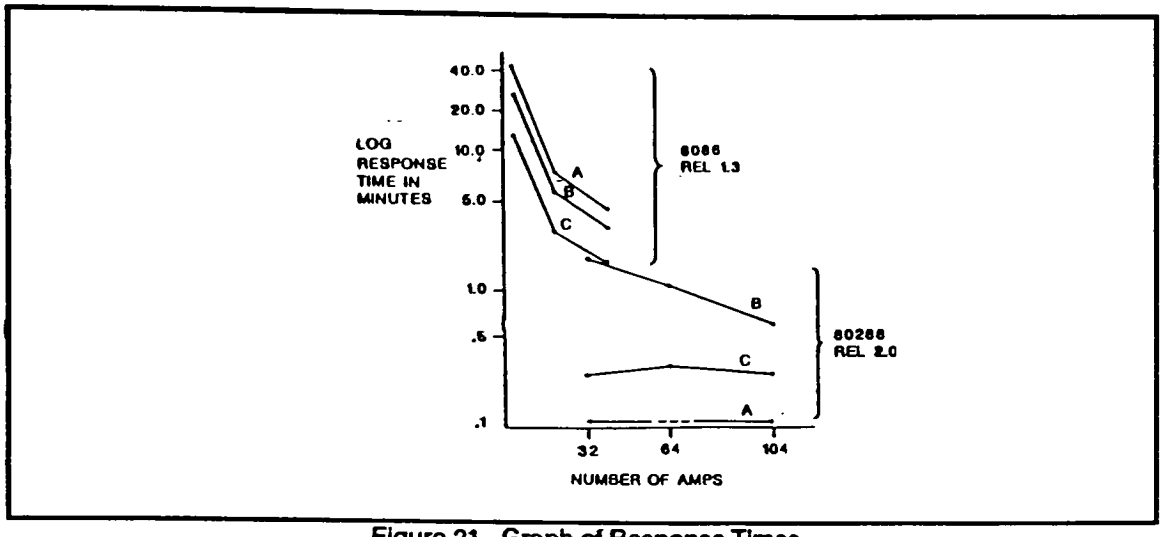

Figure 21. Graph of Response Times.

Drawing comparisons between the three Model 2 configurations, it can be seen that queries A and C reached a point beyond which their response time did not improve. Teradata explains that "maximum parallelism" (distributing the processing evenly among the AMPs) achieved its highest efficiency at the 32 AMP level. Increasing the number of AMPs did not contribute to improving this distribution scheme. Query B however, was able to expand its parallelism as the number of AMPs increased. Response times for query B on the largest configuration were 2 1/2 times faster than on the 32 AMP machine.The third workload tested the DBC/1012 in a transaction processing environment, rather than one of processing queries.

3. Banking Transaction Application

Run as a single unit of work, or "atomic operation", the transaction performed four functions of typical banking activities: debiting the customer's account; updating the totals for both the branch and the individual teller; and finally, recording the occurrence of the transaction itself. The DBC/1012's transient journaling capabilities were also tested with this application. Journal entries had to be kept in the event transaction backout occurred.

While these operations are related to each other within the banking transaction, none is dependent per se on any of the others. That is, deleting a customer account could be done at the same time as, for example, updating the branch total. It is transactions of this nature that are suited for a new feature in Release 3.0 of the Teradata software - parallel step processing [DBC/ 87]. By taking advantage of executing instruction steps simultaneously, response times improve dramatically. Had the following test been run using Release 3.0, Teradata estimated improvements of a factor of three in performance.

Four tables were involved in processing the banking transactions. The CUSTOMER ACCOUNT table held 1 million rows with three columns and 112 bytes per row. The TELLER table was comprised of 10,000 212-byte rows, also with three columns. Third, the BRANCH table had the same row format as the previous table, but held only 1,000 rows. Finally, the TRANSACTION HISTORY table had 100,000 rows of seven columns and 108 bytes each.

The performance index for this evaluation was a productivity index of throughput rate, the number of banking transactions processed per second. The first chart in Figure 22 shows the results for each of the three configurations tested. The 8 x 32 machine achieved a throughput of 46 transaction per

second (tps) and the 80 processor machine, 82 tps. This reveals an 89% linearity factor.[45] Between the 80 and the 128 processor machines, a 100% linearity increase would be 131.2 tps for the larger configuration; the first chart in this figure shows that the 24 x 104 DBC/1012 actually processed 133 tps. Teradata reports that at this, the host computer driving the 128 unit processor was not capable of introducing sufficient work to that machine to take it to its maximum capacity; it could have processed still more.

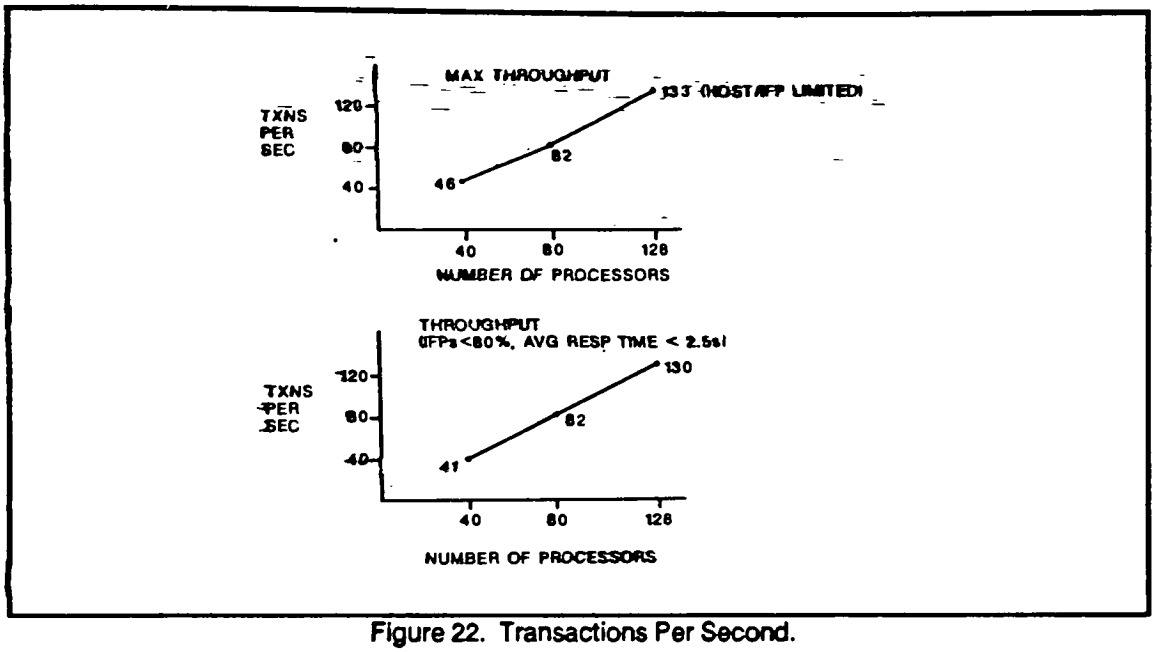

Figure 22. Transactions Per Second.

Figure 22's second chart depicts the results of the test transaction when two processing restrictions were put into place. First, the load sent to each IFP was limited to 80% of that subsystem's capacity. Second, only those transactions with a response time of under 2.5 seconds were reported. Under these constraints, the 40 processor machine processed 41 tps; the 16 x 64, 82 tps; and the 24 x 104, 130 tps.

Figure 23's first column shows actual measurements of a utilization performance index for the various components of the 128 processor machine running 360 sessions[46] of the transaction test. Processing 133 tps, this configuration ran its IFPs at 83% capacity, the AMPs at 49% and the DSU's at 53%. From this, Teradata estimates (see column 2 in Figure 23) that the number of AMPs and their associated DSUs could be reduced to 60 and still run at 85% and 92% capacity, respectively.

The performance index measurements of response time in seconds as a function of the number of transactions processed per second is shown in Figure 24. The average measured response time from the benchmark for a processing rate of 100 tps is 1.7 seconds per transaction. Also shown here are Teradata's estimates for response times under Release 3.0. The response time is predicted to

45 A 100% linearity would have the 16 x 64 configuration processing 92 tps.

46 Concurrent runs.

be lowered to less than one second per transaction, the same as the response time that Release 2.0 achieves processing less than half that number of transactions per second.

	24 x 104 (ACTUAL)	24 x 60 (ESTIMATED)
Transaction Rate	133	133
# Sessions	360	
Host Total	88+%	88+%
IFP	83%	83%
AMP	49%	85%
DSU	53%	92%

Figure 23. Maximum Throughput Utilization.

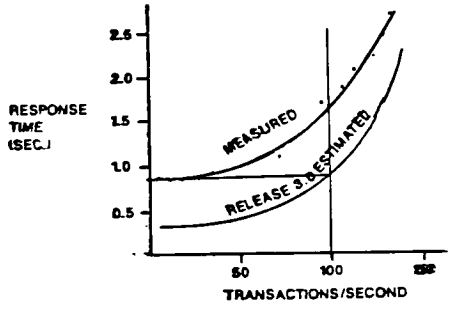

Figure 24. Response Time (seconds).

4. Data Loading Scenario

The fourth Citibank workload benchmarked was that which tested the performance of the DBC/1012 24 x 104 machine in a data loading scenario (see Figure 25). Unfortunately, no data is supplied against which to compare these results. These statistics would have been more meaningful if compared to loading tables of similar structure using Datacom's DBUTLTY LOAD function, for example [DATAC 88]. In any event, the CLAIM table was loaded at the rate of 2,105 rows per second, or approximately 20 rows per second per AMP. The STATUS table's rows were loaded at the rate of 3,387 per second; 33 rows per AMP per second. The benchmark does not specify which of the three load utilities (see the next section) was used in this test. It seems the results of this test are somewhat inconclusive because of this and because no comparisions were made either to other DBC/1012 configurations nor to the load capabilities of other database managers.

TABLE	TABLE SIZE (rows)	ROW SIZE (bytes)	TOTAL DATA	ELAPSED TIME (minutes)
Claim	5.4M	131	.7GB	42:45
Status	15.4M	58	.9GB	1:15:47

Figure 25. Data Loading Measurements.

C. Chicago Board Options Exchange

The Chicago Board Options Exchange (CBOE) Utility Benchmark [REIN 87], like the fourth Citibank evaluation, tested the data loading capabilities of the DBC/1012. Unlike the previous study, this benchmark gives specific information concerning which utilities were used for what purposes. CBOE's second study, the Application Benchmark, examined both batch and online query processing.

1. Utility Benchmark

CBOE upgraded their DBC/1012 system from a 2 x 8, 8086 processor configuration running Release 1.3 of the Teradata software, to a 6 x 20, 80286 machine running Release 2.2. The purpose of the benchmarking was to measure increases in performance resulting from the three environment upgrades: the software upgrade from Release 1.3 to 2.2, the upgrade in hardware from the 8086 processors to the 80286's and finally, the configuration upgrade from 10 to 26 processors.

a. Bulk Data Load

Three general purpose utility programs are included in the Teradata software. Each of these was tested in the Utility Benchmark. Bulk Data Load (BDL) provides the ability to insert, delete and update rows of a database table in a batch or background mode. By means of this utility, large amounts of data stored in host-maintained files can readily be moved into a database on the DBC/1012. BDL sends messages to the DBC/1012 containing INSERT or UPDATE instructions, along with the appropriate row to be inserted or data for an update. In the case of a DELETE, the message contains that instruction and the identification of the row(s) to be deleted. The DBC/1012 responds to each message it receives with either an indication of successful completion of the instruction, or failure of the same.

This same utility can be used to unload DBC/1012 database tables to user-defined files. It is this latter function of BDL that the CBOE Utility Benchmark tested. Figure 26 shows the results of running BDL to unload three different sized tables onto tape datasets. The entries demonstrate the upgrade to the new software release/processor combination, as well as the upgrade to the new configuration. In moving from A (the 2 x 8, Release 1.3, 8086 machine), to B (2 x 8, Release 2.2, 80286 processors), the percentage decrease in CPU seconds was 6% on Table 1 and 4% on Table 2. On Table 3, the largest table however, CPU seconds actually increased by more that 13%. Upgrading from B to C (the 6 x 20 version of B), CPU remained the same for the two smaller tables, but increased by 9% in Table 3.

Since Table 3 is probably more representative of the size of production database tables, one might conclude that upgrading to Release 2.2 of the software and to the 80286 processors results in more CPU intensive processing. Even so, the overall increase in CPU is minimal and is far outweighed by the benefit seen in elapsed time. In loading and unloading data, elapsed time is generally a more critcal consideration than CPU. Maintenance of this type is most often run during an off-peak-hours "window" that should not be overrun.

The elapsed time for Table 1 between machines A and B decreases slightly and increased a little on Table 2. There was a significant improvement (68%) on Table 3 however, indicating that the upgrade to the new software/processor release alone shortened elapsed time for BDL. Improvements in elapsed time are shown for all three tables with the configuration upgrade. For Tables 1 and 2, the improvement between machines B and C was 78% and 79%, respectively. Looking at Table 3, a 22% improvement is shown going from B to C; the difference between A and C is even higher, 75%.

From the above can be concluded that for BDL, upgrading software and processors alone had minimal impact on both CPU and elapsed time for the smaller tables. The same upgrade adversely affected CPU time for the larger table, but resulted in significant improvements in elapsed time. The

combination of all three upgrades resulted in little change in overall CPU consumption, but dramatic improvements in elapsed time for all three tables.

TABLE #	# ROWS
1	190
2	4,575
3	218,322

DBC/1012	TERADATA SOFTWARE RELEASE	CONFIGURATION	MODEL #
A	1.3	2 x 8	1
B	2.2	2 x 8	2
C	2.2	6 x 20	2

DBC/1012	TABLE	CPU SECONDS	ELAPSED TIME (seconds)	# SESSIONS
A	1	.32	123	24
B	1	.30	122.4	24
C	1	.30	27.6	24
A	2	.55	127.2	24
B	2	.53	129	24
C	2	.53	27.6	24
A	3	14.24	3,196.2	24
B	3	16.21	1,035.6	24
C	3	17.66	809.4	24

Figure 26. Bulk Data Load.

b. Dump and Restore

The next Teradata utility evaluated was the Dump and Restore (DAR). This program serves numerous functions, one of which is creating an archive copy (i.e. "dumping") of a database, table or permanent journal into a host dataset. Additional functions include placing checkpoints in journal tables and performing forward or backward recovery on a database to a particular checkpoint. Finally, journal table change image rows can be deleted using DAR.

CBOE evaluated the process of dumping a table to tape and restoring that same table to the DBC/1012. This table was larger than the combination of tables used to evaluate BDL. Figure 27 reveals that unloading requires less CPU utilization than loading and has shorter average elapsed time. The CPU utilization for both the unload and the load processes improved by approximately 54% with the software/processor updgrade. The hardware upgrade however, resulted in an 11% increase in CPU consumption on the unload and a 23% increase on the load.

As with BDL, the improvements in elapsed time are far more expedient than the increases in CPU are detrimental. Software/processor upgrades improved unload elapsed times by 56%; hardware upgrades resulted in an additional 39% shorter elapsed time. For the load, changes in elapsed time from improvements on software and processors is minimal. The hardware upgrade on the other hand, made a 32% improvement.

UNLOAD TO TAPE (16 sessions)	DBC/1012	CPU (seconds)	Elapsed time (seconds)
	A	80	3,162
	B	37	1,395
	C	41	850.2
LOAD FROM TAPE (16 sessions)	A	130.47	2,538.7
	B	61.09	2,580
	C	75	1,753.2

Figure 27. Dump and Restore for 1,776,396 Records.

c. Fast Load

The third and final program evaluated in the Utility Benchmark was the Fast Load. Similar to BDL, this utility loads newly created or modified database tables with data from host-resident files. Fast Load loads data much more quickly than BDL, but does not have the ability to update or delete rows. CBOE reloaded, using Fast Load, the tables that were unloaded to tape using BDL.

As shown in Figure 28, the Fast Load utility did not benefit from any of the upgrades with regard to CPU consumption. Moreover, for the smaller tables, elapsed time actually increased. On Table 3 however, elapsed time improved by 70% with the software/processor upgrades and by an additional 56% from the hardware upgrade. As with BDL, improved elapsed time is the more important consideration; and the large table is more representative of one from a production database environment.

DBC/1012	TABLE	CPU (seconds)	Elapsed Time (seconds)
A	1	.65	75.60
B	1	.77	67.20
C	1	.85	129.00
A	2	.59	132.60
B	2	.53	148.20
C	2	.59	150.00
A	3	18.62	3,393.00
B	3	21.43	1,025.40
C	3	23.70	454.20

Figure 28. Fast Load From Tape.

2. Application Benchmark.

The second portion of the CBOE study was the Application Benchmark. Here an online query was processed against Table 3 and batch queries were executed against both Tables 2 and 3. Unfortunately, no query profile information was supplied with this benchmark and no CPU seconds were provided. The complexity of the queries is unknown; it is only known that one of the online queries used a primary key and the other a non-key selection criterion. Both batch queries used primary keys.

Elapsed time increased by 135% with the software/processor upgrade for the prime-key online query. The hardware upgrade returned elapsed time to roughly its pre-upgrade level. For the non-key online query, elapsed time improved by 45% from the first upgrade, but experienced virtually no improvement from the hardware upgrade.

In a batch mode, the affects of the software/processor upgrade were not measured; the change between machines A and C is examined. Increases in elapsed time for Table 2 were 173% from the hardware upgrade; Table 3 processing saw a 21% increase. CBOE attributes these increases to the larger configuration. The data in this benchmark seems too lacking however, to draw meaningful conclusions regarding query processing in the CBOE environment.

D. Bank of America

The final evaluation to be examined is Bank of America's Relational Product Performance Assessment (RPPA) [TO 86]. In 1983, B of A adopted a Fourth Generation Products (4GP) strategy with a goal of applying relational database technology and fourth generation languages to meet their information requirements. B of A determined that by 1990, they would have a need for a store of 500 billion bytes of data; access to this data would be required by some 25 to 50 thousand users.

RPPA tested the DBC/1012 and DB2. Specific details regarding configuration, release levels or host systems was not supplied; nor were any actual measurements provided. Nonetheless, RPPA represents an important contribution to published benchmarking studies. The methodology is clear and precisely detailed. The conclusions drawn are also of major value.

For this assessment, B of A made use of IBM's Teleprocessing Network Simulator (TPNS) to model a multi-user environment functioning in a relational database management system; TPNS simulated the online activities of a specified number of users. These activities included signon, query execution, and signoff. The number of users was parameterized in the study. The interactions of up to 200 concurrent users were evaluated.

Processing requests were generated by TPNS on behalf of the simulated users and these requests were then sent to the DBC/1012 or to DB2 for execution. The performance index of response time and the validity of the output were the two major factors to be weighed in the final evaluation of the systems via TPNS.

The implementation of RPPA was the culmination of four levels of assessment of relational database management systems. The initial level was that of standard product assessment, where literature was reviewed and vendors consulted. Next came the assessment of the products in a single-user environment, measuring the systems' functionality and volume handling capabilities. The third level was a repetition of level two, only in a multi-user environment. Finally, factors such as backup and recovery, security and database load functions were evaluated.

The testing process at the second and third levels of the RPPA involved separate phases. First was the computer generation of test code. A 1,000 line PL/I program with embedded SQL statements was created for every such statement to be tested. Next, the scripts for the TPNS were generated,

simulating the various combinations of users to be tested (from one to 200 concurrent users). Third, a script was formulated for the user interactions to be simulated in each of these environments. SQL statements were transformed into TPNS-compatible table entries which TPNS, in the fifth phase, simulated as user requests. Performance statistics were collected in the sixth phase and stored on a separate database in the seventh. Reports were generated from the data in trends analysed in the next two phases. In the final phase of RPPA, the analyses and recommendations of the research group were presented to B of A management.

The DBC/1012 was assessed at the single-user level in late 1984 at Teradata under the instructions specified by B of A. The RPPA team concluded that the DBC/1012 functioned satisfactorily as a relational database management system. They reported good overall performance in executing relational operations such as projections and joins. Also, when the databases were created with indices [DBC/ 86], RPPA measured performance rates four times faster than that of the same database without an index. It was noted that elapsed time for data retrieval was faster than for insert, update and delete operations. This is to be expected since less processing would be involved in simple data retrievals.

Second, RPPA concluded that the DBC/1012 performed satisfactorily as a database machine. The distribution of work among the AMPs was determined to be effective. It was found that the "per row" processing time required decreased as the number of rows increased. Also, a 100% throughput linearity increase was noted; a 12 AMP configuration was found to execute a set of queries three times faster than with 4 AMPs; a similar improvement was demonstrated bewteen a 12 AMP and a 24 AMP machine.

Finally, RPPA reported that response times for queries run numerous times were consistent 90% of the time. This indicated that the DBC/1012 provided stability and predictability in a production environment.

In the evaluation of DB2 at the single-user level, RPPA noted that the processing times for simple queries was comparable to that of IMS. With a moderate workload against tables containing less than 100 rows of data, DB2 was found to exhibit consistently good performance. However, when complex queries with subqueries and large-scale joins were executed, response times increased geometrically. For the information management and high volume data processing needs of B of A, RPPA determined that the DBC/1012 was the better choice of database managers.

VIII. Conclusion

It is evident from the Liberty and B of A studies which compared the DBC/1012 to DB2, that the former is the better alternative for installations with high volumes of transactions and data, as well as complex data operations. In the information management world, such characteristics are the norm for most large, production environments. The Citibank and B of A studies demonstrate that there is indeed a linearity of processing capacity as the size of the DBC/1012 systems increases. This too is an important consideration in growing information management centers; installing a system that can be expanded as the processing needs of the company grow requires that the processing capabilities of that system grow at a comparable rate. Conversely, the DBC/1012 may not be the best choice for small shops with simple data processing requirements. Finally, it will be interesting to compare Release 3.0 of the Teradata software to the processing capabilities of Release 2 of DB2. Such comparisons are not yet available.

Chapter 4

Database Processing Solutions by IBM

I. Introduction

The Sierra is presently IBM's state-of-the-art mainframe computer. It is with this machine that IBM has maintained its decision not to develop a database machine per se. The Sierra conforms to the 370-XA[1] architectural scheme [IBM 7085]; the primary component is the 3090 processor [IBM 7120], [IBM 7121]. It is the revised Channel Subsystem of the 370-XA architecture, coupled with the functional capabilities of the 3090 processor that make the Sierra the primary competitor of Teradata's DBC/1012 for fast data access. The two machines are altogether different in design, yet each has been the choice of various data processing centers to meet their needs.

To examine the increasing dependence on databases and the growing requirements for fast data storage and retrieval, one must not only look at the DBC/1012, but must also view IBM's response to the situation. That response has been to dedicate more and more hardware to the support of data I/O, gradually reducing the responsibility on the part of the software operating system and of the Central Processing Unit (CPU) for those functions.[2]

II. 370-XA Architecture

The 370-XA architecture evolved from the System/370 [IBM 7000]. The four main components of a 370-XA system are: main storage, one or more CPUs, the channel subsystem and the storage subsystem (see Figure 1).[3] The most significant difference in the 370-XA architecture is the handling of I/O functions by the channel subsystem.

A. Main Storage

Main storage is that which is available to and directly addressable by the CPUs and the channel subsystem. The operating system, MVS/XA [IBM 1348] resides in main storage. Application programs and their corresponding data must be loaded into main storage before they can be processed. Different CPU models are equipped with different amounts of storage (see the section on the 3090 Processor, below). Some models include a faster-access buffer or "cache" storage to improve performance. Main storage is available in blocks of 4,096 (4K) bytes.

B. CPU

The most important component of a 370-XA system is the CPU. Here, control over the entire system takes place. Despite the different models of processors that may be implemented in this system, the logical functions of the CPU are the same.

1. Instruction Execution

The primary functions of the CPU include execution of instructions and interrupt handling.

---

1 In February 1988 [IBM 288-059], IBM announced its Enterprise Systems Architecture/370 (ESA/370) which will be available sometime in 1988 and will eventually replace the 370-XA. See the discussion later in this chapter.

2 The combination of ESA/370, MVS/SP 3.0 [IBM 288-059] and DB2 2.0 [IBM 288-196] represent the leading edge of this trend.

3 The I/O devices, along with the hardware Storage Control units comprise the Storage Subsystem.

There are five different classifications of instructions executed by the CPU: general, decimal, floating-point, control, and I/O. General instructions are those used to perform binary integer arithmetic, logical, and branching operations. Decimal instructions perform operations on data in decimal format; floating-point instructions in that format. Control instructions are used by the CPU to coordinate its internal activities. Finally, I/O instructions are those that effect the transfer of data to and from main storage.

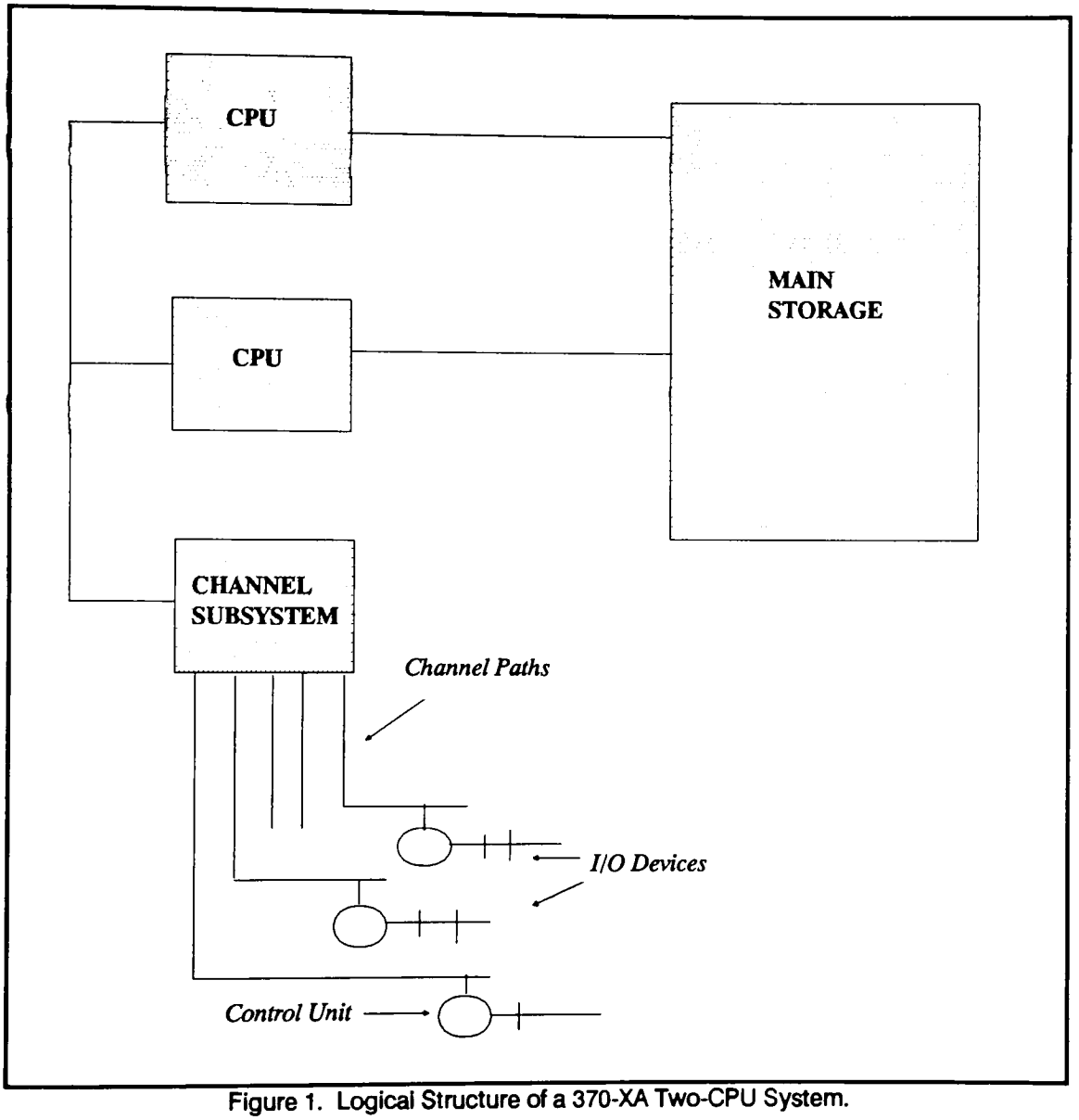

Figure 1. Logical Structure of a 370-XA Two-CPU System.

To coordinate instruction execution sequencing, the CPU makes use of a program-status-word (PSW). The PSW is actually a sixty-four bit register that contains information about the current set of instructions being processed by the CPU. When an interruption is received by the CPU the "current PSW", that which contains information about the CPU's current activity, is stored in a particular location in storage and a new PSW is fetched, beginning the interrupt handling process. After the interrupt is processed, the CPU may retrieve the PSW it previously stored and continue processing the interrupted set of instructions.

2. Interrupt Handling

There are six types of interrupts that may occur in a 370-XA system: external, I/O, machine check, program, restart and supervisor call. Each has its own "old PSW" and "new PSW" location in main storage. I/O interruptions are the ones most relevant to this study and will be discussed in more detail later in this chapter.

C. Channel Subsystem (CSS)

The third component of the 370-XA architecture is the Channel Subsystem (CSS). The transfer of data between the I/O devices and main storage is controlled here as the CSS executes a channel program.[4] In the 370-XA architecture, the CPU is relieved of the task of communicating directly with the I/O devices. Because the CSS coordinates this activity, I/O processing and the other processing that takes place at the CPU can be concurrent.

The CSS performs certain functions that had previously been handled by the operating system in architectures before 370-XA. Specifically, the CSS tests for the availability of a communication path between it and the I/O devices, selects an available path and initiates the execution of the I/O operation with the Storage Subsystem. The CSS is comprised of two components, the channel control element (CCE) and channel paths or channels.

1. Channel Control Element (CCE).

At the CCE, interactions take place between the CPUs, main storage and the channel paths. The CCE receives requests from the CPU for I/O operations. These operations are prioritized by the CCE which then initiates them with the Storage Subsystem. Coordination of status responses upon completion of the I/O operations also takes place at the CCE. Finally, I/O interruptions are sent to the CPU by the CCE.

2. Channel Paths

I/O devices in the Storage Subsystem are attached to storage control units, which are in turn connected to the CSS by way of channel paths, or channels. The channels provide the communications line necessary between the CSS and the I/O devices. Storage control units may be attached to more than one channel path; I/O devices may be connected to more than one storage control unit. There may be as many as eight different channel paths accessing a single I/O device. An I/O Configuration Program (IOCP) [IBM 7120] creates a dataset called the I/O Configuration Dataset (IOCDS) which, for a particular installation, defines the channel paths that will access the I/O devices, the storage control units attached to those channel paths, and the I/O devices that will be associated with the storage control units.

---

4 A series of channel-control words (CCWs) that specify the operation to be performed and the data to be manipulated.

3. Subchannels

A logical component of the CSS is the subchannel which facilitates the performance of I/O operations at the device level. One subchannel is defined for every I/O device accessible to a CSS and is dedicated solely to that device. The Unit Control Word (UCW) is a storage location in the hardware system area (HSA) of main storage[5] devoted to each individual subchannel and its associated device. The UCW for a subchannel contains information about the device and the operations of which it is capable. The UCW acts as a means by which the CPU and the CSS can obtain information about the attached device and its operations.

D. Storage Subsystem

The fourth 370-XA component is the Storage Subsystem. It has two types of components, the Storage Control Units and the Direct Access Storage Device (DASD) system (see Figure 2).[6]

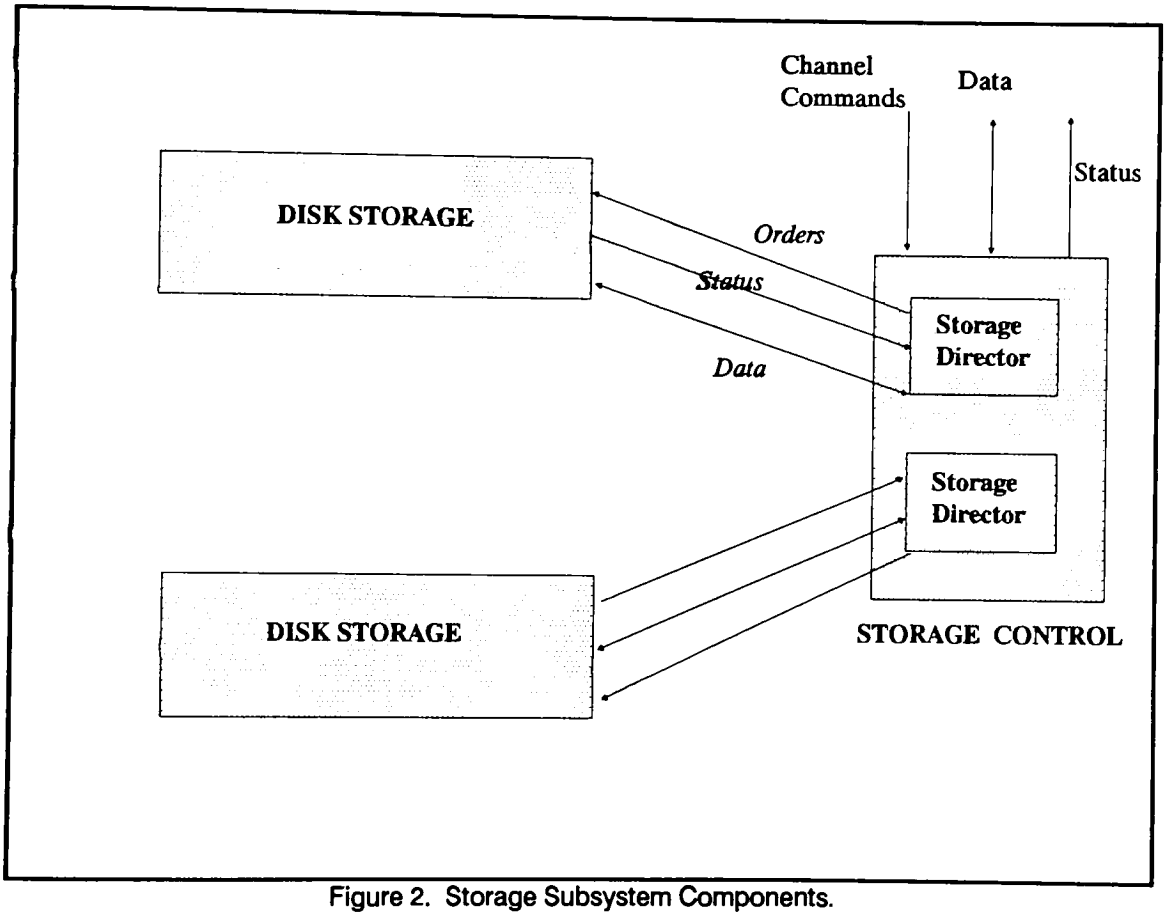

Figure 2. Storage Subsystem Components.

---

5 HSA is a section of main storage not less than 288 KB in size that is made unavailable for use by programs running on the CPU.

6 For purposes of this report which focuses on fast access of data, only the DASD system and not the tape subsystem is reviewed here.

1. Storage Control Unit

The Storage Control Unit, a component of the 370-XA Storage Subsystem, is hardware which interprets I/O commands sent to it by the CSS over a channel path, sending those commands on to disk storage for actual data access. Each Storage Control Unit has one or more Storage Directors that control the flow of data from the disk, to the CSS, to main storage; or from main storage, through the CSS, to the disk. The Storage Director also supplies information to the CSS regarding the status of the chosen device. Certain data error checking and correcting functions are performed by the storage control units.

2. Direct Access Storage Device (DASD) System

The second Storage Subsystem component is the DASD system. Disk storage, as introduced in Chapter 2, is comprised of a disk drive, the disk platters, the spindle on which the platters are mounted and one or more access mechanisms, or "arms". The drive controls the rotation of the disks so that the proper section of a disk surface becomes positioned under the read/write heads that are located on the arm. The time required to position the disk on the correct position is sometimes called "rotational delay" or "latency". Disk surfaces are sectioned into units called cylinders which are further broken down into tracks. The number of bytes that can be stored on a track or cylinder is dependent on the disk model. "Seek time" is the amount of time required to move the access arm between cylinders. The "transfer rate" of data is the speed with which it is moved between the disk and the storage control unit; this rate is measured in bytes per second.

Multiple disk storage devices may be connected in a "string". Within each string of devices is one or more internal "controllers".[7] This controller coordinates the inter-connection of the devices in the string. The disk storage strings, along with their controllers are connected to the Storage Control Unit (see Figure 3). The number of strings that may be attached depends on the models of both the Storage Control Unit and the DASD system.

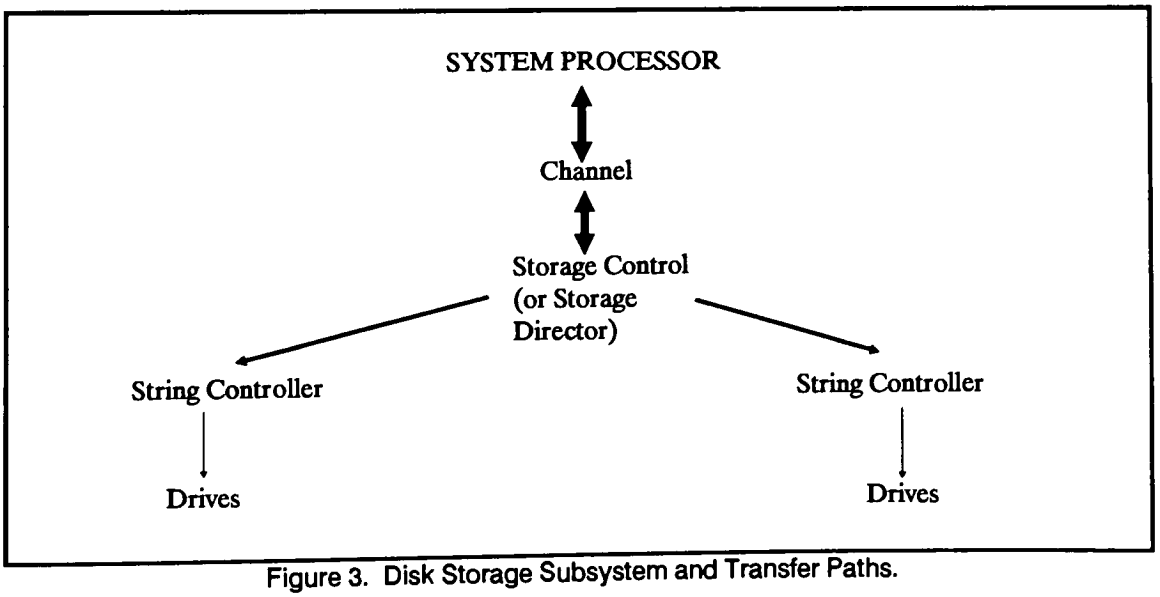

Figure 3. Disk Storage Subsystem and Transfer Paths.

7 A hardware component of the disk drive, not the same as the Storage Control Unit or its Storage Directors.

III. Sierra

As mentioned earlier, the Sierra is a machine based on the concepts of 370-XA architecture. It is appropriate to discuss the specific models of the primary 370-XA components that together function as the Sierra. The specific components are the 3090 Processor Complex [IBM 7120], the 3880 Storage Controller [IBM 1661] and the 3880 DASD subsystem [IBM 4491].

A. 3090 Processor Complex

The 3090 Processor Complex embodies three of the architectural components of 370/XA, discussed above: main storage, CPU and channel subsystem. To date there are twenty three different 3090 Model Configurations [IBM 3090], [IBM 188-037], [IBM 188-123]: 120E, 150, 150E, 180, 180E, 200, 200E, 280E, 300E,[8] 400, 400E, 500E, 600E and the newest models announced in July 1988, 120S, 150S, 170S, 180S, 280S, 200S, 300S, 400S, 500S and 600S. The models denoted by "E" are the "enhanced" versions; both these and IBM's newest releases, the S series, will support the ESA/370 architecture.

The components that most significantly distinguish these models from each other are the channel subsystems and the number of CPUs. Models 120E, 150 and 150E each contain sixteen channel paths with an optional eight additional channel paths for a maximum of twenty-four. The 180, 180E, 120S, 150S, 170S and 180S have sixteen standard and optional channel paths for a maximum of thirty-two. Forty-eight channel paths are possible on the Model 200; sixty-four on the 200E, 280E, 300E, 200S, 280S and 300S Models; ninety-six on the Model 400; and 128 on the Model 400E, 400S, 500E, 500S, 600E and 600S. There is one CPU in the 120E, 120S, 150, 150E, 150S, 170S, 180, 180E and 180S configurations. Models 200, 200E, 200S and 280E have two CPUs. The 300E and 300S have three CPUs; the 400 , 400E and 400S have four; the 500E and 500S have five; and the 600E and 600S have six. Central storage on the configurations with multiple CPUs is shared among them.

The strength of the capabilities of the 3090 Processors is demonstrated in the three high-end models, the 400S, 500S and 600S. Each has 512MB of central storage and two channel subsystems. They may be configured as "single-image" machines in which the activities of the two CSSs are coordinated and appear to the operating system as one single CSS. Functioning in single-image, central storage is shared and instruction execution occurs on all of the CPUs, increasing the processing capacity by four to six times over that of a single CPU. These models may also be physically partitioned into two "sides" to function as two separate mainframes. The 400S may be split to appear as two 200Ss; the 500S split as a 300S and a 200S; and the 600ESto appear as two 300Ss. Each side of the processor so divided would have its own CSS.

The next two Sierra components, the 3880 Storage Control Unit and the 3380 DASD system together comprise the storage subsystem component of the 370/XA style architecture as discussed above.

B. 3880 Storage Control Unit

The Sierra's third major component is the 3880 Storage Control Unit [IBM 0067], [IBM 1661]. These units have two Storage Directors which coordinate the actual transfer of data between disk storage and the CSS. The two Storage Directors are connected to separate channels of the CSS. Each Storage Director of a 3880 Model 2 attaches to as many as four channels; 3880 Models 3, 13 and 23 up to eight.

---

8 Recently, [IBM 188-037] upgrades to this and the 400E models have been announced.

Models 13 and 23 are the high performance models of the 3880, utilizing a high speed electronic storage component called "subsystem storage" [IBM 0067]. This storage is made up of two elements, a cache and a directory. The cache storage is the larger of the two and stores recently accessed data in "track slots", sections that can hold data from one track of a 3380 DASD. By keeping this data in the cache, its access is theoretically faster. The directory contains information about the contents of each track slot, as well as access activity by the processors.

If the channel program requests data to be read from the disk, the directory is consulted to determine if that data is present in the cache. If it is, the data is transferred immediately to the CSS; this is a "read hit". If the data is not in the cache (i.e. a "read miss" occurs), the 3880 Storage Director involved in the operation sends commands to the disk for the data and transfers it to the channel. If possible, a track slot is made available and the data, from the place on the track where the requested record was found to the end of that track, is transferred into the slot. Conversely, during an operation where data is to be written to a disk, if the data is in the cache there is a "write hit" and the cache is updated. The disk data is subsequently updated as well. In the event of a "write miss", the data is written to the disk.

C. 3380 Direct Access Storage

The third Sierra component is the 3380 Direct Access Storage system [IBM 4491]. The term "device" in the context of the 3380 is used to refer to one set of disk platters on a spindle (a "volume"), the two access arms associated with that volume and the electronic circuitry connected to them. Two such devices are enclosed in a Head Disk Assembly (HDA). Two HDAs (i.e. four devices) comprise a 3380 "unit"; a unit consists of two sets of magnetic disks and four sets of access arms (see Figure 4).

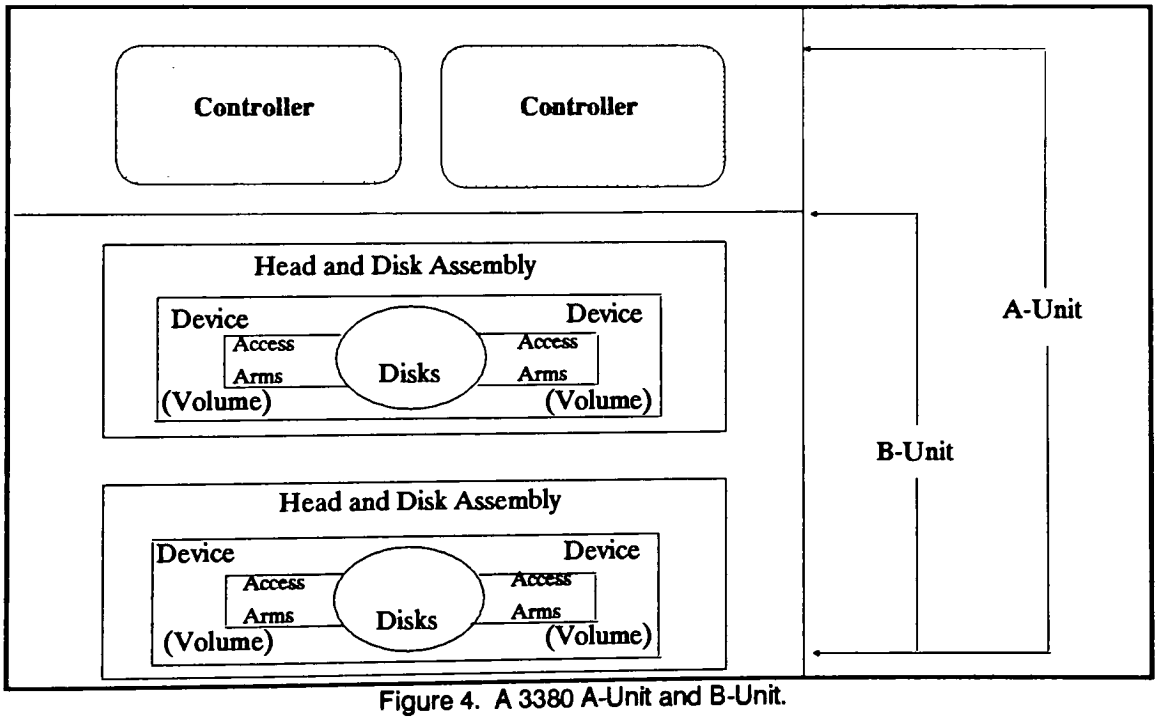

Figure 4. A 3380 A-Unit and B-Unit.

There are two types of 3380 units, A-units and B-units. As many as three B-units may be attached to an A-unit in a "string". One or two 3380 strings may be attached to each of the two Storage Directors of the 3880 Storage Control Unit. As many as thirty-two devices are accessible through each Storage Director for a total of sixty-four devices per 3880.[9]

A-units are equipped, depending on the model, with two or four hardware controllers that provide four primary functions for accessing the data on the disk. First, commands issued by the Storage Control Unit are interpreted and executed by the A-unit controllers. Second, the actual transfer of data to or from the disk surfaces and the Storage Control Unit are coordinated by the controllers. Third, status information about the disks is supplied by the controllers to the Storage Control Unit. Finally, certain error detection and correction is initiated by the controllers.

A Model A04 3380 unit is a "single-path string" unit. It consists of B-units attached to an A-unit with one controller and has one data transfer path between the string and the Storage Control Unit. Such a string can support up to sixteen separate devices; four devices from the A-unit itself, plus four from each of up to three B-units. Data is transferred over that single path to any device on the string. "2-path strings" are comprised of A-units with two controllers (Models AA4, AD4 and AE4). These strings support the same number of devices as the single-path string models, but data can be transferred on two paths to or from two devices on the string concurrently. All of the 3380 models have a data transfer rate of 3.0 MB per second. Figure 5 gives a summary of some of the other characteristics of the different 3380 models.

Physical Characteristics	3380 Model	
	A04 AA4 B04 AD4 BD4	AE4 BE4
Devices per Unit	4	4
Data Cylinders per Device	885	1,770
Data Tracks per Device	13,275	26,550
Bytes per Track	47,476	47,476
MB per Device	630	1,260
MB per Unit	2,520	5,041

Figure 5. 3380 Model Summary.

9 A configuration based on one 3880 attached to each of 128 channels of a 3090 Model 600E would allow access to 8,192 3380 devices.

Each A-unit controller can provide access to any device on the string by means of four separate "internal paths".[10] Depending on the 3380 model, access capabilities to the devices improve with successive models.

1. Standard Models

The AA4 is considered a "standard" model. Each of the four internal paths on either controller may access up to four separate devices. An internal path from one controller may connect with an internal path of the other controller, making the devices on that path accessible to either controller (see Figure 6). Two devices located on the same internal path may not be accessed by both controllers at the same time, but two devices located on separate internal paths may be accessed by each controller simultaneously. For instance, if device 0 in Figure 6 were being accessed by controller A1, device 1 could not be accessed at the same time by controller A2 because the two devices are on the same internal path. Controller A2 could however, access device 2 at the same time that controller A1 is accessing device 0, since the two devices are on separate internal paths.

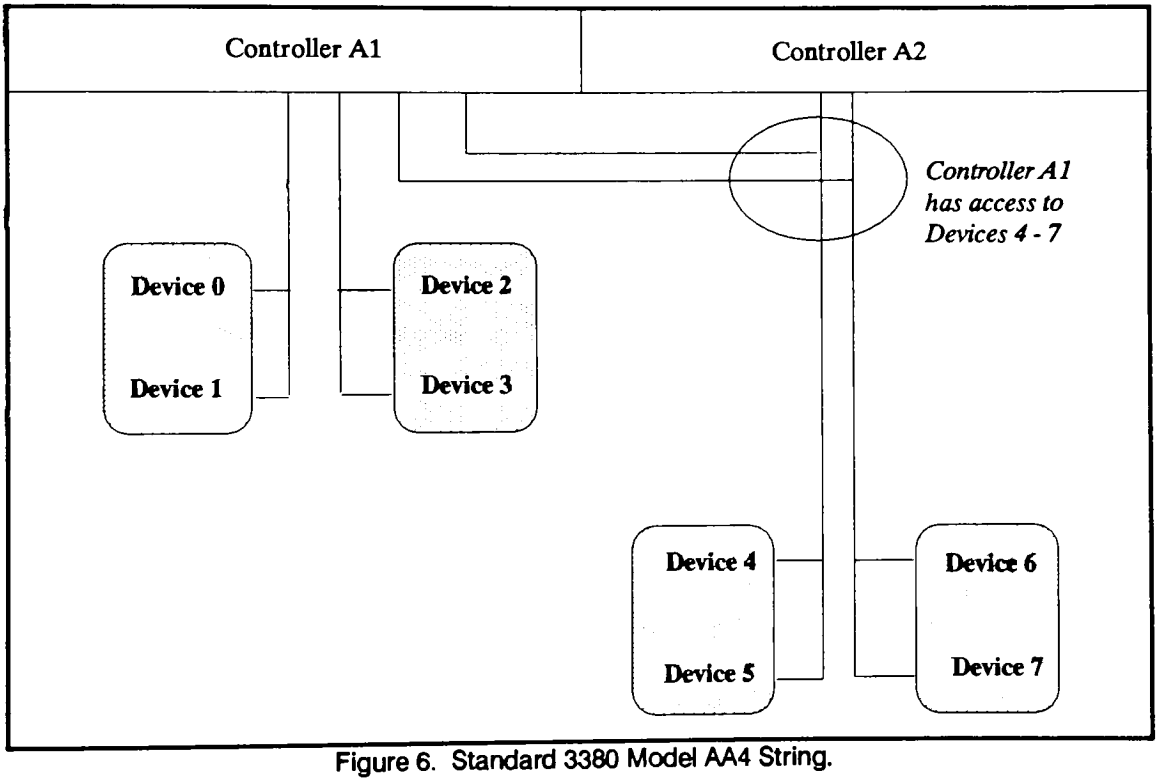

Figure 6. Standard 3380 Model AA4 String.

Information is transferred between the string and the Storage Control Unit over one of the two paths connecting the A-unit controllers with the Storage Control Unit on the 2-path string. For a "standard" 2-path string, a function called "Dynamic Path Selection" (DPS) determines which of the two data transfer paths will be used in transferring data between the Storage Control Unit and the string. If

---

10 "Internal paths" between the controllers and the devices are not to be confused with the two "data transfer paths" between the A-unit controllers and the Storage Control Unit on a "2-path string".

one path is busy, an alternate path to the other controller will be selected. Data may be transferred over both paths simultaneously.

2. Extended Capability Models

3380 Models AD4 and AE4 are considered Extended Capability models. Where in the Standard AA4 model any one device was connected to one internal path, in the Extended Capability models each device is accessible by two internal paths, one from each controller. Since either controller may thus select any device in a string, any two devices may be selected and accessed concurrently. If an internal path on which a device is located is busy, access to that device can be made through the second internal path. Device Level Selection (DLS) is the function that makes it possible for any two devices to be accessed simultaneously. The amount of time that is needed to wait for access to a device is significanlty reduced.

3. Disk Storage Management

The third topic for discussion regarding the 3380 System is Disk Storage Management. 3380 devices are categorized as "Count, Key and Data" (CKD) devices,[11] which describes how the device tracks and the records written to them are formatted. CKD tracks are formatted between two index points (see Figure 7). Immediately following the first index point is the track address or "Home Address" (HA) which contains the cylinder number and head number associated with that track. Information is also contained in a flag byte ("F flag") of the HA, indicating whether or not the entire track is defective.[12] If only a portion of the track is defective, the HA will contain control information causing the defective area to be skipped.

The first record following the HA on the track is Record 0 (R0), which acts as a track descriptor record. The HA and R0 are established when the track is manufactured. If the F flag of the HA indicates that the track is defective, the count area (see below) of R0 will contain the address of the alternate track. Similarly, the R0 of the alternate track contains the address of the defective track.

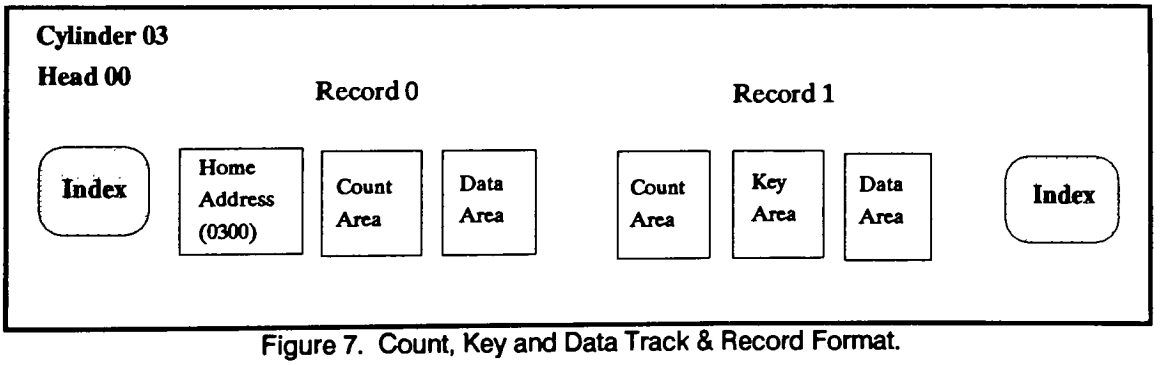

Figure 7. Count, Key and Data Track & Record Format.

---

11 As opposed to Fixed-Block Architecture devices. See [IBM 1675] for more details.

12 If a track is defective, an alternate track is used. The flag byte of the alternate track will indicate that it is being substituted for a defective track.

Following the R0 on a track are stored one or more data records in CKD format. CKD records have three areas: the count area, key area and data area. The key area is optional (e.g. R0 has no key area) and is used to identify the information stored in the data area. The count and data areas are not optional for CKD records. The count area contains information about the address of the record's data area by specifying the cylinder number and head number,[13] in addition to the record number.[14] The length in bytes of the key area, if it exists, and of the data area is also contained in the record's count area. An "end-of-file" record is one that marks the end of a logical group of records. The count area of the end-of-file records contains a data area length equal to zero.

IV. I/O Processing on the Sierra

An examination of the way in which this machine competes for database access performance with the DBC/1012 must focus on the processing of I/Os. While different models of the components discussed above may be combined to dedicate increasing amounts of hardware to the support of these functions thus improving their performance, the concepts of I/O processing in the Sierra remain the same.

A. Initiation of I/O Operations

As mentioned earlier, one of the primary functions of the CPU is the execution of instructions. Application programs are translated into a series of CPU instructions the execution of which accomplishes the tasks the application program was designed to perform. The data that is being acted on by an application program, as well as the instructions that constitute the program itself, must be present in main storage in order for processing to take place. This requires the transfer of information from for example, a 3380 device to main storage; this is called "input". In addition, if the data is modified during that processing, it must be rewritten to the device it was input from; this constitutes "output".

1. Start Subchannel

When it is determined that some certain data from a particular device must be transferred into main storage, the CPU initiates the operations necessary to accomplish this by executing the I/O instruction "START SUBCHANNEL" in which the subchannel associated with the device is designated. The START SUBCHANNEL instruction is a 370-XA I/O instruction that follows the "S" instruction format as shown in Figure 8. See [IBM 7085] for more details on instruction formats. The first two bytes signify the instruction "opcode" which specifies the operation to be performed. In Figure 8, the opcode "B233" means "START SUBCHANNEL". General register 1 contains the "subsystem-identification-word" (SID), a four-byte field the last two bytes of which contain the "subchannel number"[15] of the subchannel that will be used in this operation. Bits 16-31 of the START SUBCHANNEL instruction give the address of the "Operation-Request Block" (ORB).

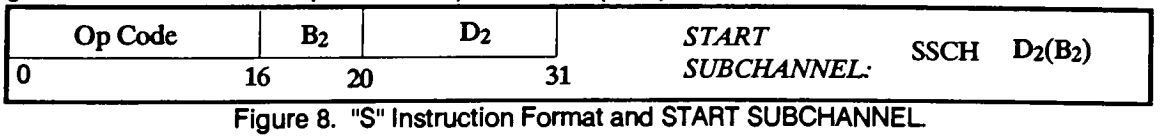

Figure 8. "S" Instruction Format and START SUBCHANNEL

13 The "track address".

14 The sequential number of the record on the track.

15 A number unique within the system, used as an address for a subchannel.

2. Operation-Request Block (ORB)

During the execution of the START SUBCHANNEL Instruction, the contents of the ORB are placed at the subchannel designated by the SID. The ORB is a three word block of storage (see Figure 9) that contains several parameters used by the CSS. The first word (word 0) of the ORB contains information that will be used in the Interruption code that the CSS will send to the CPU when this I/O operation is complete. The third byte of the second word (I.e. word 1) is a "logical-path-mask" (LPM) which indicates which of the eight channel paths used for accessing the device associated with the specified subchannel are available for use. Bits 0-7 of the LPM correspond to the eight channel paths. If any bit is set, the corresponding channel path is available. Word 2 of the ORB specifies the address of the first Channel-Command-Word (CCW) in the channel program to be executed (see below).

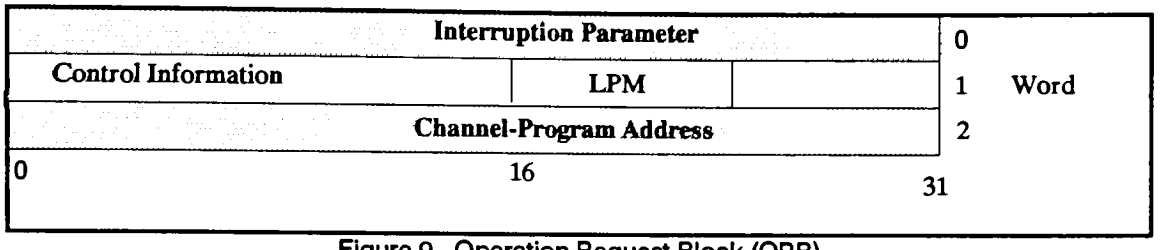

Figure 9. Operation Request Block (ORB).

Once the contents of the ORB have been passed to the subchannel, a condition code of 0 is returned to the CPU's current PSW and the START SUBCHANNEL instruction execution is complete.

3. Channel-Command Word (CCW)

A CCW is an eight byte block of storage containing information about the I/O operation to be performed. Execution of a "channel program" consists of the execution of a CCW or a groups of CCWs linked logically together and executed in sequence, resulting in the performance of the I/O operation. The two possible formats of a CCW are shown in Figure 10. Format-0 CCWs can access addresses anywhere in main storage; format-1 CCWs may only access the first 16 MB. The information contained in either format is the same; the position of the fields within the CCW is different between formats.

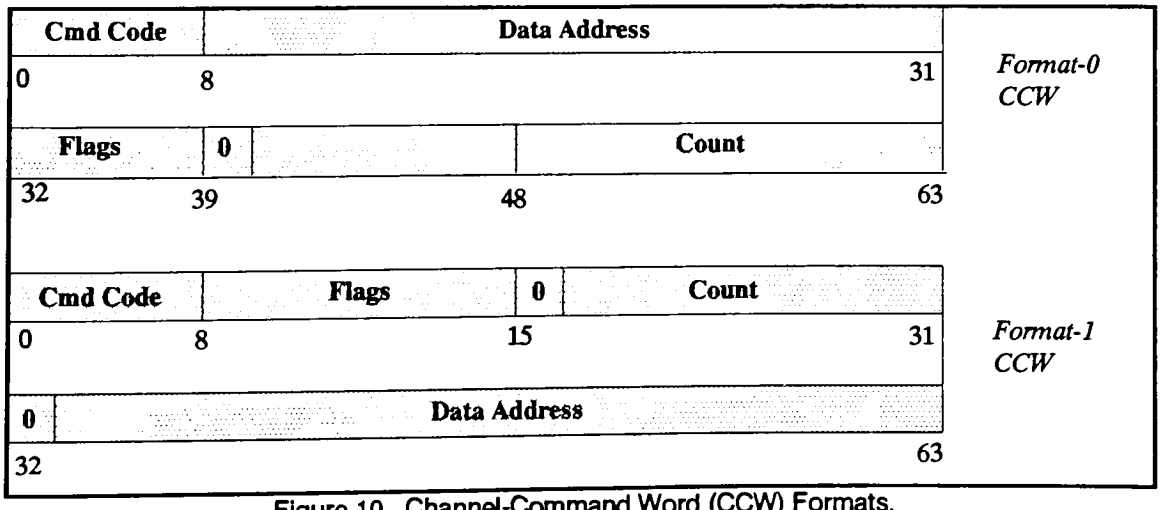

Figure 10. Channel-Command Word (CCW) Formats.

The first byte of the CCW contains a code that signifies the command that will be sent to the Storage Control Unit in order to effect the operation to be performed. The commands most relevant to this discussion are the READ and WRITE commands. For more information on other types of commands, see [IBM 7085] and [IBM 0067]. There are six different WRITE commands and nine READ commands.

The Data Address field of the CCW specifies a location in main storage. In a READ command this field specifies the location in main storage where the data that is read from the device is to be placed. In a WRITE command, this field specifies the location of the data to be written back to disk. The Count field of the CCW specifies the number of bytes of data affected by the command. Finally, there are two one-bit chain flags in the CCW that contain information used in establishing the channel program. The Chain Data (CD) flag, if set, indicates that the data specified by the next CCW is to undergo the same operation as that specified in the current CCW. This is "command-chaining".

The CCWs that make up a channel program are located in contiguous storage. The address of the next CCW is calculated by the CSS by adding eight to the address of the current CCW. The address of the first CCW is contained in the ORB.

B. START Function of the Channel Subsystem

There are certain I/O instructions which specify that the CSS is to perform a particular function in conjunction with that instruction. The START SUBCHANNEL is such an instruction and directs the CSS to perform a START function. Upon completion of the START SUBCHANNEL instruction execution, the CPU is released to execute new instructions. The CSS is signalled to perform a START function with the disk device. Having the CSS interact with the storage subsystem in this manner is the essence of the improvements made in the 370 architecture by the XA version; in previous 370 architectures, the CPU would be involved in these functions.

The START function consists of three parts. First, the CSS engages in path-management operations to test for and select an available channel path to the device. Second, the channel program that effects the requested I/O operation is executed. Finally, the completion of the I/O operation is indicated at the subchannel.

1. Path Management

Path Management by the CSS in the 370-XA architecture includes functions that were performed in previous architectures by software components. Using the information contained in the LPM field of the ORB, the CSS attempts to choose an available channel path to the device. A logical connection is made between the channel path and the Storage Control Unit to which the device is attached.

The CSS sends to the Storage Control Unit the command code of the first CCW. This code is sent by the Storage Control Unit's Storage Director to the device, which in turn sends back a status byte indicating whether or not the command can be executed. If circumstances are right for the transfer of data, the connection between the channel path is maintained and the execution of the channel program may begin.

2. Channel Program Execution

The CSS fetches and decodes the CCWs that comprise the channel program using the control information contained in the CCWs as described above. When the device has completed the operation specified by a CCW, a device-end signal is sent to the CSS indicating that the device is ready to execute another operation. The CSS fetches the next CCW. When the channel program execution is complete, a channel-end condition indicating that the transfer of all of the data is complete and that the CSS's services are no longer required. This constitutes an "interruption condition".

3. Indicating Completion of Operations

When the CSS recognizes the interuption condition, information is stored at the subchannel indicating that the operation has reached successful completion. The subchannel is considered to be "status-pending", a condition which causes the CSS to make a request for an interruption at the CPU.

C. I/O Interruptions

The I/O interruption request remains pending at the CPU until it is acknowledged there. When the CPU accepts the interruption request, a TEST SUBCHANNEL I/O instruction is executed, making certain information concerning the execution of the requested I/O operations available to the CPU. This information is placed in the Interruption-Response Block (IRB). Depending on the nature of the operation, data is either transferred into main memory (in a READ operation), or sent to be written to disk (a WRITE operation) during the interruption handling at the CPU. Assuming a normal end to the operations, the processing of the program which the CPU was executing when the I/O Interruption was processed, can continue.

V. IBM Mainframe Directions

If any portion of IBM's mainframe systems will assume the functionality of a back-end database machine, it will evolve from the Storage Subsystem; the Storage Control Units and DASD Subsystems, managed by components of MVS's Data Facility Product (DFP). Storage Management is perhaps the most critical concern for IBM and its customers today. Methods are being sought by which the Storage Subsystem will coordinate all aspects of data management including strategic storage of database records to promote faster, more efficient access and retrieval.

The first steps in this evolutionary process are seen in the introduction of the Enhanced DASD Subsystem Models AJ4/BJ4 and AK4/BK4 [IBM 4491]. They will be able to be configured in 4-path strings when connected to the new 3990 Storage Control Units [IBM 0098] available during the third quarter of 1988 [IBM 3990]. These new hardware components coupled with the data management facilities of Release 2 of DFP [IBM 4142] mark the beginnings of IBM's System Managed Storage strategy [BRAU 86d].

A. 3990 Storage Control

The 3990 Storage Control is the 3880's successor. It removes data storage related functions from the CPU. IBM has introduced major performance enhancements in the new 3990. Via 4-path strings in Models 2 and 3 of the 3990, connected to 3380 Model AJ4/BJ4 or AK4/BK4 DASD strings (see below), each device is attached to four separate internal paths, twice the number of a 3880 Model 23 Storage Control Unit. There can be four independent I/O operations concurrently active on a configuration with these components.

The features of the 3990 Model 3 are the most advanced of the three available models and so the focus of the remaining discussion will be primarily on this model. The Model 3 contains larger cache sizes than the 3880 Models 13 or 23. 32, 64, 128 or 256 MB cache memories are available. Better cache space utilization is achieved through improved cache slot segmentation. Data transfer rates of the 3990 Storage Control are as high as 4.5 MB per second, compared with the 3.0 MB rate of the 3880.[16] When attached to the Enhanced Subsystem 3380 Models, the Device Level Selection En-

---

16 Transfer rates to and from the DASD string remain however, at 3.0 MB per second.

hanced (DLSE) mode of data retrieval is supported by the 3990 (see below). The most significant features of the Model 3 however, are the Fast Write facilities and Dual Copy.

1. 3990 Components

There are three primary components of the 3990 Storage Control: storage clusters, non-volatile storage (NVS) and cache, each of which is located in its own separate power and service regions (see Figure 11). Each 3990 has two storage clusters. Where the 3880's Storage Directors connected from one to eight channels in the CSS, each of the 3990's storage clusters can be attached to either four or eight channels. Within each storage cluster are two storage paths managed by a multipath Storage Director.

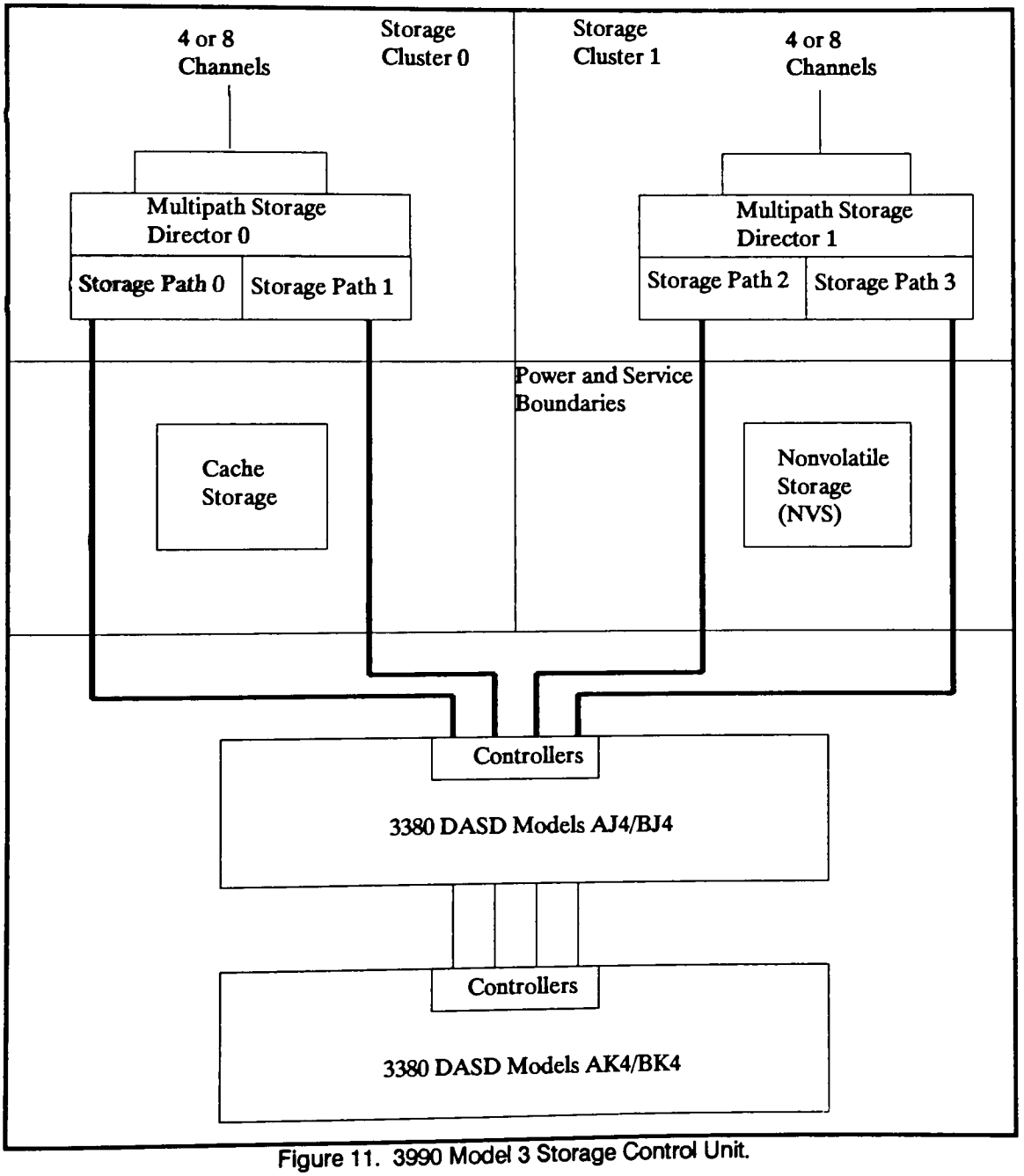

Figure 11. 3990 Model 3 Storage Control Unit.

Each channel from the CSS that is attached to a 3990 storage cluster is connected to both storage paths within that cluster. This ensures either that a single CPU has several paths to the data on the attached DASD string, or that multiple CPUs can each have their own path to the data. The Storage Directors, as in the 3880, interpret commands from the CSS and coordinate the transfer of data and status information back to the CPU. Unlike the 3880's Storage Directors however, the 3990's have the ability to choose either of two storage paths in its storage cluster by which to access the data. This eliminates a large percentage of wait time due to "path busy" conditions occurring in the 3880. Additional functions of the Storage Directors in the 3990 are the control of the cache and the NVS.

The second 3990 component, the NVS, is a 4 MB random access electronic memory device used in the DASD Fast Write and Dual Copy operations (discussed below). Data can be retained in the NVS for up to forty-eight hours without an electrical power supply due to its battery operated characteristics.

Finally, the 3990's cache, like that of the 3880 Models 13 and 23, is a high density electronic storage device which is shared by all of the storage paths. The cache is also located in its own power region, reducing single points of failure. Improved segmentation capabilities in the 3990's cache over the 3880's allow the logical relating of separate 16 KB segments so they can be treated as single units of data. Transfer rates of data between the cache and the CSS are also improved, supporting speeds of up to 4.5 MB per second.

A "dual frame" 3990 configuration can be supported wherein two Model 2s or two Model 3s are physically bolted together and their primary components are interconnected. Each string of DASD is attached to one multipath Storage Director in each of the two 3990s. This configuration is recommended for installations making use of the Dual Copy or DASD Fast Write operations because there is a circuitry that is common between the cahce and NVS. If a failure occurred in this circuitry on a single 3990, access to data in either the cache or NVS could be temporarily lost. A dual frame configuration duplicates this circuitry, eliminating this possibility.

2. Fast Write Capabilities

Available only in Model 3 units, the 3990's Fast Write operations greatly improve system response time by eliminating most immediate writes of data to DASD. Write operations performed in cache proceed at cache speeds since DASD rotation and seek times are eliminated.

a. DASD Fast Write

During operations performed by DASD Fast Write, data involved in cache write hits or format writes[17] do not require acces to DASD immediately. Data is stored simultaneously to the cache and the NVS (see Figure 12). Then, the Storage Director returns both channel- and device-end statuses to the CSS. The channel program can continue processing without having to wait for data to be written to DASD. If space needs to be freed either in cache or NVS, the data stored there under DASD Fast Write operations is written to DASD (the data is "destaged".)

---

17 Write operations that involve creation of new records are called "format writes". There is no need to verify any data on a DASD track before allowing the data to be written to cache.

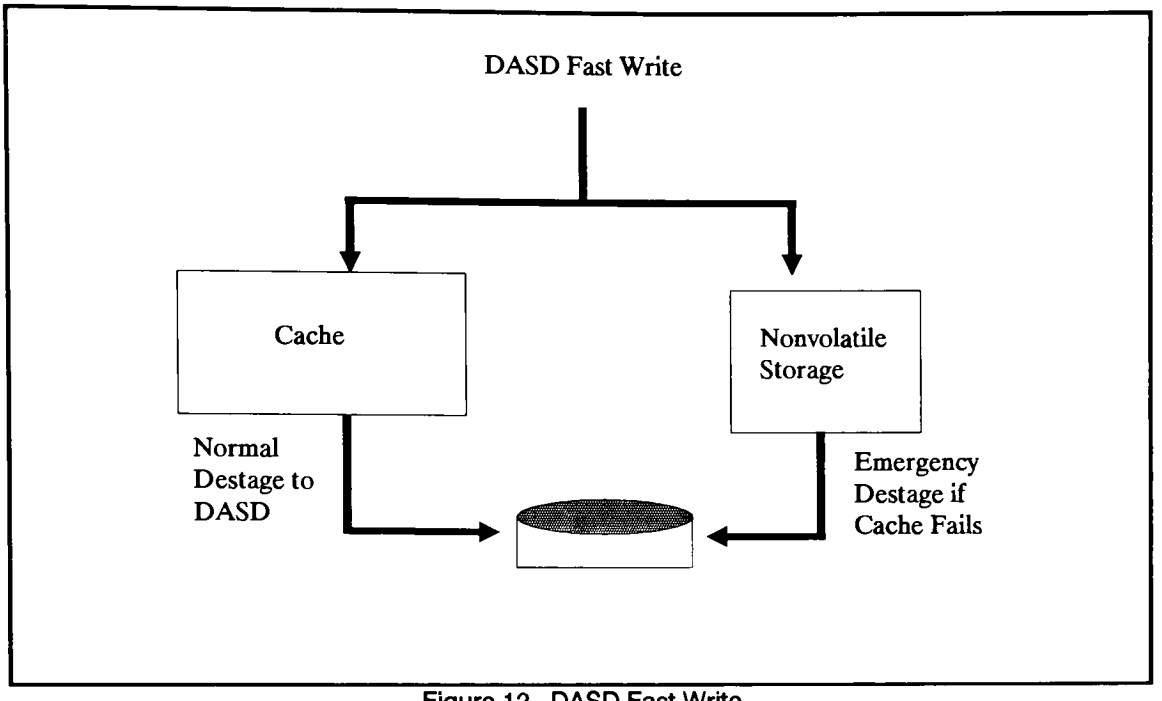

Figure 12. DASD Fast Write.

b. Cache Fast Write

Unlike DASD Fast Write, Cache Fast Write operations do not make use of the NVS. Cache Fast Write is used for certain kinds of data such as temporary datasets or files created from DFSORT operations.[18] Instead of writing the data from these files to DASD, small to medium sized sort operations may be performed exclusively within the cache, at cache speeds.

3. Dual Copy

The 3990's Dual Copy operations, similar in concept to the DBC/1012's fallback AMPs, create duplicates of the data located on those DASD volumes enabled for these functions (see Figure 13). Two devices form a "duplex pair", consisting of a primary and a secondary device. The Dual Copy operations are transparent to the CPU and are managed solely by the Storage Subsystem. All channel commands are directed to the primary device. Data is written to that device and to the cache simultaneously. Channel- and device-end statuses are returned for the primary device and the 3990 later completes the operations from the cache to the secondary device which is not connected to the CSS. If the primary device fails, the secondary device is accessed instead, again similar to the DBC/1012's fallback AMP.

Dual Copy operations provide several important advantages to the Storage Subsystem. First, critical data can be protected from the event of a failure on a single device. Second, maintenance windows previously required for creating backup copies of data are reduced or possibly eliminated, providing near continuous operation. Finally, forward recovery procedures may be eliminated since updates are duplicated virtually concurrently on the duplex pair. Another operation available on the 3990 is a combination of DASD Fast Write and Dual Copy operations, called "Fast Dual Copy".

---

18 A component of DFP supporting sort functions, DFSORT Release 9 supports Cache Fast Write.

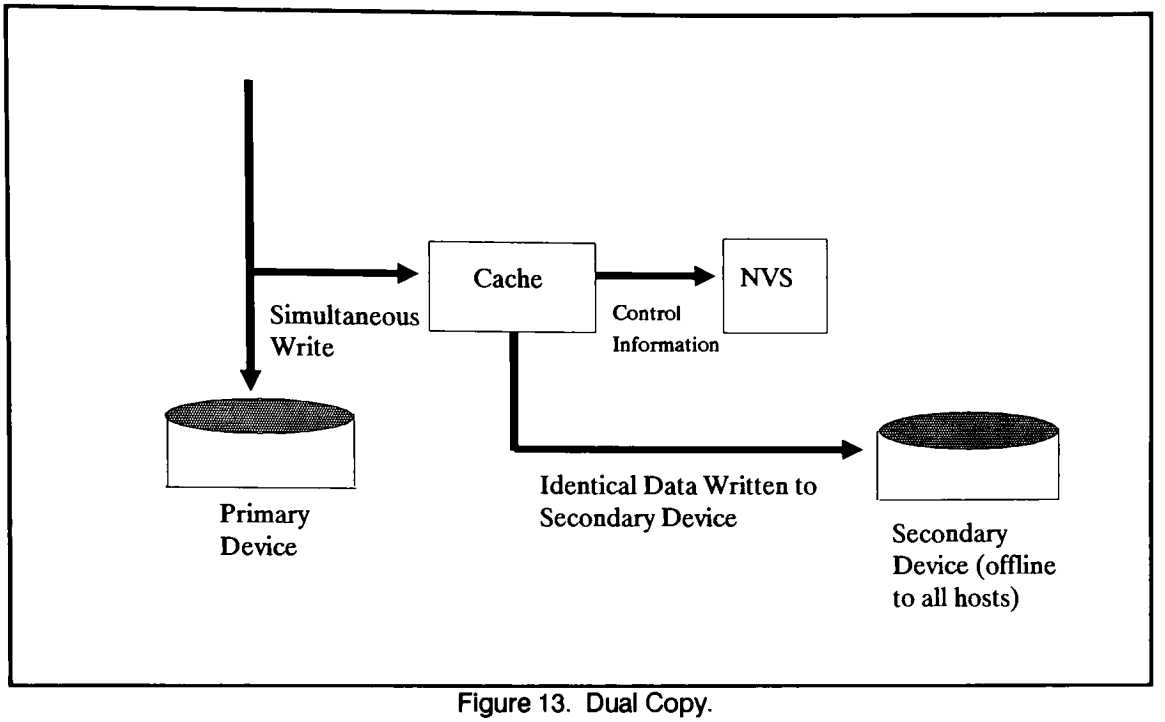

Figure 13. Dual Copy.

B. Enhanced DASD Subsystem Models

Mentioned in the previous sections, there are new 3380 Models AJ4/BJ4 and AK4/BK4 which are the Enhanced Subsystem Models. They may be attached to the 3990 in 4-path strings. Each string has four A-unit controllers, unlike earlier models with only two. Each controller can access any device on the string by way of any one of its four internal data paths, up to thirty-two devices. When these DASD models are connected to a 3990 Model 2 or 3, data may be transferred simultaneously to or from any four devices on the string in what is called Device Level Selection Enhanced (DLSE) mode.

C. Data Facility Product (DFP)

As new advances are made in the hardware components of the Storage Subsystem, new functions of DFP will be developed to support and coordinate them. DFP constitutes the largest percentage of the MVS/XA operating system [IBM 4142] and performs four major functions in systems based on the 370-XA architectural scheme: program management, device management, data management and interactive storage management. How these functions are enhanced and their integration with other products such as IMS and DB2 will in a large part determine the extent to which the System Managed Storage strategy is successful in meeting the data access and retrieval needs of the IBM customer community.

The device management facilities of DFP support the creation of the IOCDS discussed in the 370-XA overview. This defines the I/O devices to the operating system. Although the 3990 is not yet supported by the current Release 3.2 of DFP, future release will include 3990 and DLSE support. Cache management is already provided for the 3880 cache models.

The primary functions of DFP reside however, in its data management components which coordinate data organization and storage. Two DFP components involved in the management of data are the Data Facility Data Set Services (DFDSS) and Data Facility Hierarchical Storage Manager (DFHSM). Functions of DFDSS assist in reclaiming fragmented space on DASD volumes, as well as manipula-

tion of actual datasets to effect copy, move, dump and restore operations. DFHSM also assists in managing DASD space and provides different storage "levels" that provide more efficient placement of datasets for better access. While system analysts today interface with DFDSS and DFHSM by way of a third DFP component, the Interactive Storage Management Facility (ISMF), plans are to isolate storage management decisions to the Storage Subsystem itself.

D. New IBM Announcements

On February 15, 1988 IBM made a series of product announcements introducing the successors to the 370-XA architecture and to MVS/XA. These new products are Enterprise Systems Architecture/370 (ESA/370) and MVS/ESA which is comprised of MVS/SP Version 3 and MVS/DFP Version 3 [IBM 288-059], [IBM 2/18/88]. Processor speeds and DASD capacity have continued to increase, along with the complexity of user transactions. I/O subsystem speeds have lagged however, causing a disparity between the rates at which data can be transferred from disk (3.0 MB per second in the newest 3380 Models) and the rate at which the 3990 Storage Controller can process that data (4.5 MB per second). The use made by ESA/370 of expanded storage in the form of data spaces and hiperspaces (see below) instead of relying on auxiliary storage, was intended to rectify this inconsistency. It is with these new products that performance enhancements for DB2 processing will challenge the capabilities of the DBC/1012. Available in late 1988, ESA/370 and MVS/ESA will usher in a new era for IBM mainframes.

1. Enterprise Systems Architecture (ESA/370)

The Enterprise Systems Architecture (ESA/370) is a new processor architecture for the 3090 enhanced models [IBM 288-059], [IBM 2/18/88]. IBM views its introduction as an evolutionary step beyond the 370-XA architecture. ESA/370 should be available sometime in late 1988. Primarily, ESA/370 establishes the environment in which IBM's new operating system, MVS/ESA functions. New registers and an enriched instruction set provide new hardware support for virtual addressing, permitting access of virtual memory in multiple address spaces and in the new data spaces (see below). This extended addressability will be available to the operating system, various subsystems (see for example, DFSMS below) and to applications.

2. MVS/Enterprise Systems Architecture (MVS/ESA)

The new MVS/ESA is comprised of the products MVS/SP Version 3 and MVS/DFP Version 3 [IBM 288-059], [IBM 288-060]. The former should be available with the ESA/370 architecture; DFP at the end of 1988.[19] Functional enhancements to MVS and DFP under IBM's new direction of Systems Application Architecture (SAA)[20] will be delivered through MVS/ESA.

a. MVS/SP Version 3

Version 3 of MVS/SP was designed to function with the new ESA/370 architecture, to take advantage of the new addressing capablities it offers. The primary features of the new MVS release are Data Spaces and Hiperspaces.

---

19 MVS/SP version 3 can run with DFP version 2.3 [IBM 2/18/88].

20 IBM's plan to give users a single view of the system. The same applications would be able to run on machines in all three tiers of IBM's architectural scheme. See [GART 87a], [GART 87b] and [GART 87c] for further details.

1. Data Spaces

Where in MVS/XA, programs and data shared an extended storage area of 2 GB, in MVS/ESA separate data-only, virtual addressing spaces of 2 GB each are available to applications and subsystems and are isolated from program storage altogether. The emphasis for the use of data spaces is on data that is read heavily. This includes individual datasets, database buffers and tables,[21] work files and load modules. Data spaces may be shared by multiple users so that they may all read the same byte of data, simultaneously. Access to the data spaces is controlled by hardware in the ESA/370 architecture. A data space is a virtual addressing space. Data in such a space can reside anywhere in physical storage, including auxiliary storage. It is therefore subject to normal storage contention and paging activities.

2. Hiperspaces

An acronym for "high-performance-space", hiperspaces are buffers in extended storage designed exclusively for use in reading and writing data in 4 KB blocks. There are two types of hiperspace, one for authorized system program use only and one available to all applications. Because hiperspaces reside only in extended storage, they are not subject to storage contention or paging activities like the data spaces. Hiperspaces provide better response time for data requests. Data in a hiperspace can be referenced by high-level languages using the new "data windowing" system service.

3. System Services

Manipulation of the data spaces and hiperspaces is made possible by a series of three system services: data windowing, authorized service and library service. A "window" is a user-defined area in an application that gives that application a specific view of the data, not unlike a user view in a database application. The data windowing system service of MVS/SP Verion 3 allows data in a window to be manipulated by statements from programs written in Assembler, FORTRAN, PL/I, COBOL, C and Pascal. The data in a window can be either a permanent or temporary data object. As mentioned above, data in a hiperspace can be referenced by data windowing.

The authorized service provides the ability for programs to store objects, like load modules or datasets, in a data space. Finally, the library service automatically identifies the system's most frequently used load modules and places them in data spaces where they will be accessed on subsequent retrievals.

b. MVS/Data Facility Product (MVS/DFP) Version 3

The new version of MVS/DFP is IBM's first step toward making system managed storage a reality. A new subsystem, the Data Facility Storage Management Subsystem (DFSMS), is comprised of MVS/DFP Version 3, DFHSM, DFDSS, DFSORT Release 10 and RACF Release 8 [IBM 2/18/88]. The primary goal of this subsystem is to separate the logical from the physical view of data. Eventually, users will not have to be concerned about dataset placement. Parameters will be established for the installation as a whole by a system administrator; DFSMS will control all other aspects of dataset management.

Conforming to standards set for a particular installation, the system administrator establishes "storage groups" for datasets. Storage groups are defined by three factors: data class, storage class and management class. The data class defines the characteristics of a dataset (i.e. whether it is a partitioned dataset, sequential, etc.). Management class establishes what services the dataset will need

---

21 Enter DB2.

(e.g. how frequently it should be backed up). The storage class places the dataset under system managed storage. Factors like performance preference and hardware availability priorities are established by a dataset's storage class. These three attributes combine to form the dataset's storage group which determines its physical placement in the storage subsystem; previously such decisions were made by users who may or may not have been able to optimally place datasets within the system.

3. DB2 Considerations

DB2 Version 2 will be available sometime in late 1988 [IBM 288-196]. It is designed to utilize the new features offered by ESA/370 and MVS/ESA which should significantly improve its transaction processing rates by making better use of system resources. Utilization of data spaces and hiperspaces will provide fast, efficient and simultaneous access to data by multiple users. Placement of DB2 tables and buffers in data spaces and DB2 system load modules in hiperspaces will significantly increase performance for DB2 applications. In addition, the DFSMS will provide system managed storage for DB2 related datasets. Coupling all of this with the 3990's capabilities for Fast Write and Dual Copy and the 3380 subsystem's enhanced DASD models, makes optimal use of the latest in IBM mainframe system technology.

This synergy between hardware and relational database technology is promising. Its success however, depends on the introduction of user interfaces and tools. Efficient high-level language and query facilities are seen lacking in the line of IBM products. Other vendors, like ADR with Release 2.0 of its IDEAL fourth generation language [IDEA 88] are offering SQL interface capabilities. Within the next year, it will be determined how well the DB2-ESA/370 combination will work.

4. Summit

Announcement of IBM's next generation of mainframes, the Summit is anticipated in the third quarter of 1989, with actual release projected for the 1990's [RADD 87]. A step beyond the ESA/370, there is a great deal of speculation concerning the new machine's architecture. Some [MORA 87] anticipate that it will make a radical change from the current 370-XA and the newly announced design; others [RADD 87] believe that current architectures will be incorporated into a more complex, overall architecture. It is certain that IBM, with the products discussed above, has begun a gradual move toward an entirely new line of mainframes.

Features expected to be included in the Summit are fiber-optic channels accessing pools of DASD, instead of the present configuration of strings attached to the CSS of a particular CPU [CONN 87/88]. From a larger perspective however, the Summit is likely to encompass "cluster processing" [MORA 87], a direction foreshadowed by the 3990's storage clusters. Cluster processing would enable various data processing functions to be performed within a group of processors dedicated to the efficient execution of those operations. The clusters would be linked to each other via high speed buses, and to a common pool of storage devices by the fiber-optic channels. Isolating storage management functions into such a cluster would in effect create a back-end database machine [CONN 87/88].[22] Other cluster processing candidates are general purposes processors and scientific processors. Increments in processing capacity will probably be achieved by adding more processors to a particular cluster, in much the same way that extra processors may be add to the 3090 models today.

---

22 There is currently a project at IBM being referred to as Ole which is expected to produce an off-load-engine (o-l-e) to manage databases [CONN 87/88].

VI. System 38 (S/38)

The System/38 (S/38) [IBM 7728] belongs to the middle tier of IBM's three-tier architectural scheme. Its discussion is relevant to this research since this machine was designed and functions as a multi-user database machine. Where the DBC/1012 began as a back-end processor for mainframe environments and evolved toward the support of workstation environments with its COP, the S/38 was initially developed for wokstation support and has added features providing connectivity to mainframe systems.

The primary purpose of the S/38, like that of the DBC/1012, is the management of databases and their related processing activities. Certain of the database management functions of the S/38 are performed by the Control Program Facility (CPF), the system support software; other functions are managed by the hardware in the system. The S/38 has a unique interface, the High-Level Machine Interface (HLMI), which coordinates the interaction between the CPF and the system hardware.

A. High-Level Machine Interface (HLMI)

The HLMI depicted Figure 14 handles many of the resource management functions, as well as certain supervisory functions that have traditionally been performed by operating subsystems. Basic in the design of the S/38's database management functions is the concept of object-orientation. Objects, when speaking in terms of the S/38, are specific items defined by a set of attributes and comprised of data. Examples of objects in a S/38 are database files, user programs, message queues and device descriptions.

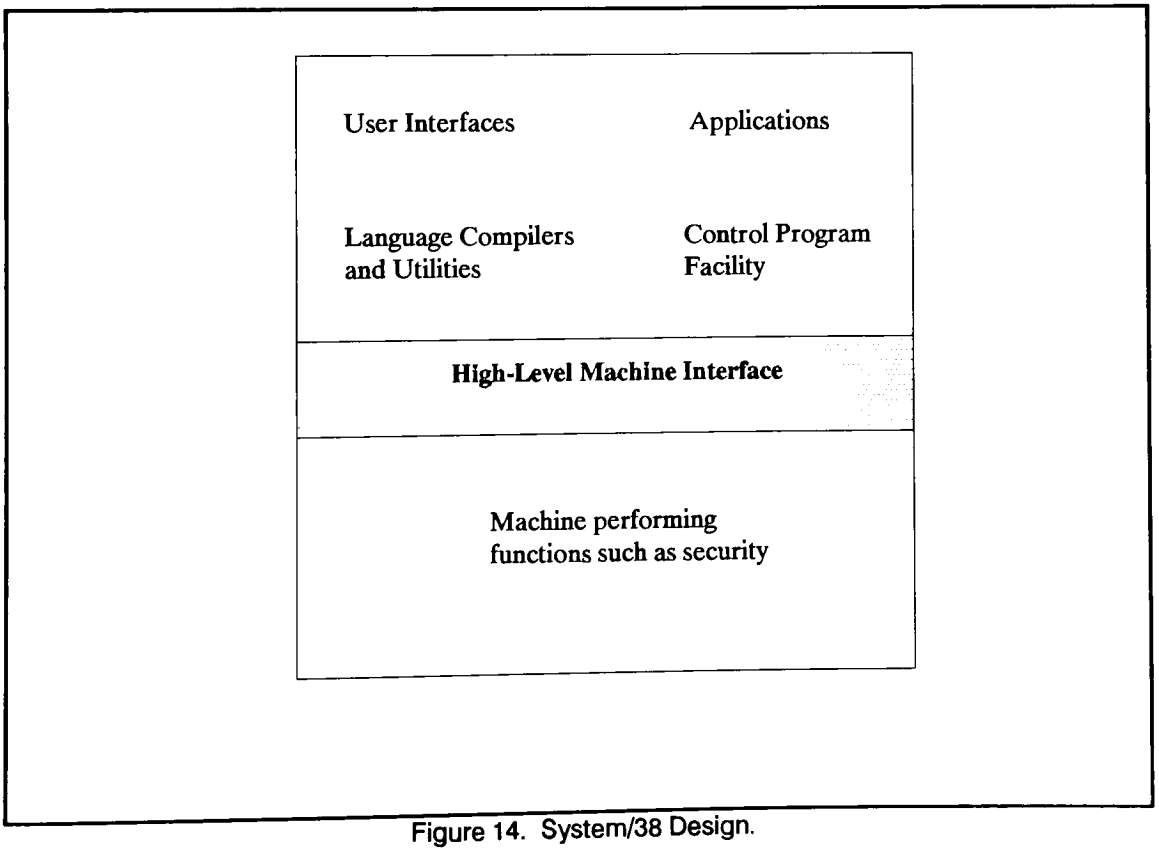

Figure 14. System/38 Design.

When objects are created, "access paths"[23] to those objects are created by the CPF. An object may have more than one access path. Control of these paths is the first major function of the HLMI. Authorization to access specific objects is enforced here. This offers a level of security to the data stored on the S/38. Concurrent operations along one or more access paths against the same object are also coordinated by the HLMI, ensuring the accuracy of the data retrieved by multiple users.

The second major function of the HLMI is the management of machine resources which are shared by concurrently executing user programs. These resources include the processing unit and storage devices. Before they are executed, user programs on the S/38 are translated into an internal object code. The HLMI manages the allocation of working storage for program variables, as well as the control locks placed on objects accessed by the program. Finally, the HLMI coordinates the activities of certain hardware components, such as the I/O channel (see below), communications networks, system printers and other asynchronous devices.

B. Control Program Facility (CPF)

The CPF as mentioned above is the software subsystem whose activities, along with those of the hardware subsystem, are coordinated by the HLMI. There are three major aspects to the services provided by the CPF. The first of these is the establishment of a user interface to the functions of the S/38. The other services are database data management and data communications.

1. User Interface

The user interface provides various means by which users can communicate with the S/38. The common thread of communication is the S/38's Control Language (CL), maintained by the CPF. CL, not unlike DBC/SQL, is a set of commands used to request system functions ranging from creating a new database file, to sending messages to other users of the system. CL commands can be entered at the terminal interactively, or may be compiled into a CL program which may in turn be called by a program written in a higher language such as COBOL, PL/I or RPG III.[24]

Another component of the CPF's user interface functions is called Data Description Specifications (DDS). This facility allows users to establish definitions of specific data fields to be retrieved from records stored on the database. Instead of defining the format of the data within the program accessing that data, programs may refer to DDSs stored externally to the program. Changes made to the actual format of the stored records will not affect the programs, unless fields specified in the particular DDS are changed. This eliminates a fair amount of recompilation.[25] The second service provided by the CPF is the management of database data.

2. Database Data Management

All data on the S/38 Is stored in "physical files" which contain fixed length records all of the same format within a physical file. The description of the physical files' format is contained in DDS entries. Logical files are the second type of file managed by the CPF. Logical files define the access paths by which data in physical files may be accessed. More than one logical file can define an access path to the same physical file. Moreover, data fields from the records of up to thirty-two physical files

---

23 A type of index.

24 The programming language of the S/38.

25 While intended to provide program independence, the CPF's DDS is not as versatile as relational databases' "views" of data.

can be specified for retrieval on one logical file. Control over the use of access paths is maintained by the HLMI.

3. Data Communications

The third major service of the CPF is Data Communications which is comprised of two separate facilities: Distributed Data Management (DDM) [IBM 286-250] and the Distributed Host Command Facility (DHCF). By means of DDM, users of one S/38 (the "source") can access data in physical files on a remote ("target") S/38. Likewise, users at the target S/38 can access data on the source S/38. With DHCF, users of 370 architecture mainframes may access and execute programs against data in the S/38.

C. S/38's Hardware Subsystem

The activities of the S/38 Hardware Subsystem are coordinated with those of the CPF by the HLMI. The components of the Hardware Subsystem include a central processing unit, storage management facilities, an I/O channel, main storage and auxiliary storage. There are many similarities between these components and those found in a system conforming to the 370-XA or ESA/370 architectures. It is the CPF and the HLMI that distinguishes the S/38. Currently, the S/38 is available in six models: 100, 200, 300, 400, 600 and 700 [IBM 186-103].

1. Central Processing Unit

Models 100 and 200 of the S/38 are equipped with IBM 5381 processors; the other models have 5382s. As in systems following the 370 architectures, the S/38's CPU controls fetching and storing data, arithmetic and logic processing and executing instructions. Unlike the other designs, the S/38 does not have a channel subsystem to control communications between the CPU, main storage and the I/O devices. In the S/38, the CPU manages all of these functions.

2. Storage Management

The Storage Management facilities of the S/38 maintain a directory of the storage locations for all of the objects in the database. 370-XA users must specify certain information when retrieving data, program execution space when running programs and allocation parameters when storing data. Users of the S/38 are relieved of this interaction[26] by the machine's storage management functions. As a program runs, the S/38 allocates as much storage space as is required. Similarly, DASD space defined to a S/38 (see below) is treated as one contiguous volume of storage. Placement of physical and logical files in storage is coordinated by the machine; these files may in fact be allocated space across more than one volume.

3. I/O Channel

While the S/38 does not have a CSS like that of a 370 system, it nevertheless does have an I/O channel, coordinated by the CPU which establishes a connection between that component, main storage and the I/O devices. I/O operations on the S/38 do overlap with other processing occurring at the CPU. The I/O channel operates in two modes; in "byte mode", information passes through the channel in 8-bit units; in "halfword mode", 16 bits, or 2 bytes are transferred at one time. Data transfer rates in the channel are 2.5 MB per second in byte mode, 5.0 MB per second in halfword mode.

---

26 As will 370 users once system managed storage becomes a reality.

4. Main Storage

One of the primary distinctions between models of the S/38 lies in main storage capacities. Figure 15 shows the different amounts available. As the 370-XA's main storage is available for application programs and data, so to is the S/38's. Model 700 utilizes IBM's new one-million-bit dynamic random access memory chip (DRAM) [IBM 186-104], expanding its main storage capacity to 32 MB.

Model Number	Memory Capacity (MB)
100	2, 4
200	4, 6
300	6, 8
400	6, 8
600	8, 12, 16
700	16, 24, 32

Figure 15. S/38 Model Memory Capacities.

5. Auxiliary Storage

The standard DASD subsystem for S/38 has been the 3370 model, fashioned after the 3380 subsystem of 370-XA installations. 3370 units have two models, A12 and B12; the former, like the 3380 A-units, is equipped with a controller. Also like the 3380s, up to three B12 units may be connected in a string called an "attachment" to an A12 unit. One attachment of up to four DASD units may be connected to S/38 Models 200 and 300; two to Models 400, 600 and 700. Since each 3370 unit provides approximately 730 MB of storage capacity, attachments to S/38s can provide from 730 to 2,920 MB capacity for the lower end models; up to 5,840 MB on the higher models. Data transfer rates for 3370 devices average 1.86 MB per second.

The S/38 can be connected with two new DASD models in addition to the standard 3370s. These new models are the 9332 [IBM 186-109] and the 9335 [IBM 186-107] both of which may be connected to any S/38 model. The 9332 is a medium capacity subsystem which comes in three models. Model 220 has one access arm (called an "actuator"), 200 MB of capacity and a data transfer rate of 2.5 MB per second. Up to thirty-two Model 220 devices may be connected to the higher model S/38s for a total capacity of 6,400 MB. Models 400 and 420 of the 9332 DASD each have two actuator arms. These two models have a storage capacity of 400 MB each and up to sixteen devices may be connected to upper model S/38s. The data transfer rates are the same as that of the Model 200.

The 9335 DASD is a high capacity subsystem with models A01 and B01, similar in function to the 3370 or 3380 units, except that up to four B01 units may be connected in a string. Each of these models has two actuator arms and a storage capacity of approximately 856 MB. As many as four 9335 strings (up to twenty devices) may be connected to upper model S/38s. The data transfer rate is the same as the 3380 DASD, 3.0 MB per second.

VII. Silverlake

On June 21, 1988 IBM announced the long awaited results of its Silverlake project. The IBM AS/400 system is a middle tier machine combining features of the System/36 and System/38 processors [IBM AS/400], [IBM 188-089]. Like the S/38, it too is a database machine; like the ESA/370 architecture in the upper tier, it is the basis for support of SAA in the middle tier [IBM 288-289]. There are six AS/400

system models each running the new OS/400 operating system. Availability of these new machines is planned for late 1988.

The AS/400 models are the B10, B20, B30, B40, B50 and B60. They have been designed with upward migration in mind so that small businesses can move to higher models as their needs grow; this is not unlike the DBC/1012's modular design of adding more IFPs and AMPs. Memory capacities range from 4-16 MB in the lower range models, to 150 MB in the B60. DASD capacities range from 9.45 MB to 27.3 GB.

A product announced along with the new AS/400 machines is Structured Query Language/400 (SQL/400) [IBM 288-293] providing users with the ability to access data in the AS/400's databases using a subset of SQL commands. Conforming to the SQL Relational Database Interface defined in SAA, it is intended that SQL/400 applications may be readily transported to other machines under the SAA umbrella. This may be prove significant in light of the need discussed above for such interfaces in the DB2 environment.

VIII. Conclusion

It is clear that IBM plans to incorporate all three architectural tiers under SAA. The importance here is IBM's apparent commitment to standardization. The common thread that is woven through all of the above discussions is relational database support. As has been seen, increased hardware support for relational database functions is evident in the new ESA/370 architecture as well as in the S/38 and the AS/400 systems. This improvement in hardware must however, be matched by efficient user tools and interfaces. IBM is first and foremost a hardware company [MART 84]. It will be interesting to see how it addresses its customers software support needs in the next few years.

Chapter 5

The Future of Database Machines

I. Introduction

Given the competition of IBM's advancements in mainframe technology and its focus on the integration of hardware support for database processing, is there a future for database machines? The theoretical aspects of this type of system continue to be researched at academic institutions; more hardware designs, software algorithms and data placement techniques have been proposed. But what of the role of database machines in the computer industry? Do they offer a viable enough solution to the needs of MIS to capture a market that will endure? In short, are database machines here to stay?

Conservatively, since only one company, Teradata, has successfully offered a machine targeted at the support of very large databases[1] speculation on the solvency of this branch of computer technology may be premature. Some new database machine proposals will be examined later in this chapter. These and those discussed in Chapter 2, remain for the most part theoretical, having at best been implemented as unrefined prototypes. The true test of their abilities will come only if they are incorporated into systems where they will be subjected to day-to-day production workloads in the business arena. Until then, their contribution to this technology cannot be fully measured and MIS's exposure to database machines will affectively be limited to the one design that has been successfully marketed, the DBC/1012.

Optimistically, Teradata has performed extremely well in the marketplace and continues to add to its list of Fortune 500 company customers [SALO 87]. The nature of database processing has become such that some [MALL 87] suggest that conventional computer systems, even with increased processor speeds and enhancements in software capabilities, cannot compete with the performance gains attainable with dedicated backend database machines. Few anticipated that stores of data would assume the magnitude that we see today. Compounded by the growing number of applications utilizing databases, the capacity of the DBC/1012 in both processing and storage is an attractive alternative for many companies.

Realistically, the concept of database processing by dedicated backend hardware is becoming more and more generally recognized. In 1986, Teradata received the acclaimed Product of the Year Award for Outstanding System from the American Federation of Information Processing Societies (AFIPS) for the DBC/1012 [SALO 87].

There remains however, the possiblity that IBM's mainframe/database technology and the loyalty of its ever expanding, firmly established customer base will curtail the growth of the database machine industry. A great deal depends on the business strategies of Teradata over the next two to five years. Analysts have thus far viewed the company's performance favorably and are generally encouraging about its prospects for the future [BRAU 87a]. The caliber of company that Teradata has won over as customers is impressive, an indicator that the DBC/1012 at least, is here to stay.

This chapter will examine four new database machine designs that have been published since 1984. One of these, GRACE [KITS 84] has emerged from research conducted at the University of Tokyo, Japan, further suggesting that database machine technology is being treated as a serious business

---

1 Britton-Lee has introduced a large system database machine, the BL8000 [BRAU 87b], but it is held with skepticism by some [BRAU 87d] because it does not offer MVS support.

opportunity. The design of GRACE, like that of a second design to be reviewed here, GAMMA [DEWI 86], emphasizes the correction of specific shortcomings in the features of the database machine DIRECT [DEWI 79] discussed in Chapter 2; GAMMA evolved from the previous DIRECT design. JASMIN [FISH 84] is a design produced by Bell Communications Research. The fourth machine, BUBBA [WILK 86], [COPE 87] is a current project at the Microelectronics and Computer Technology Corporation (MCC) in Austin, Texas. [2]

The section following the discussion of the new database machine designs will focus on Teradata, where it stands today and where it intends to be tomorrow. There will be a discussion of the company and its products, as well as a look at its endeavors slated for future enhancements. Finally, a number of issues have emerged from the growth of database technology in general and from database machine technology in particular. The concluding sections of this chapter will discuss those issues that have gained overall acceptance with regard to database machines and also look at those still in need of resolution.

II. JASMIN

The JASMIN database machine [FISH 84] was designed at Bell Communications Research. A prototype of this machine was connected to an AT&T 3B20S host computer running the Unix operating system, version 5.0. Like the DBC/1012, JASMIN is a multiprocessor database machine intended for use with very large databases and high transaction volumes. The key features of the design, again paralleling those of the Teradata machine, include the ability to increase performance by expanding its configuration and its use of off-the-shelf parts. Software components and hardware components work together in various combinations.

A. Software Components

There are four software components in JASMIN; the Kernal, Intelligent Store, Record Manager and Data Manager. Communication among these modules is accomplished by means of 16 byte fixed-length messages, control of which is handled by the Kernal. Larger amounts of data can be passed between the components by associating a buffer with the message and sending all of the information across the system's communication channels.

The Kernal establishes the environment on which the other software components are implemented. In addtion to the service of intercomponent messaging, the Kernal also provides multi-tasking within modules and scheduling services. In JASMIN, the concept of "module" refers to one or more related tasks running on the same processor and sharing an address space. "Tasks" are parallel computations. Multi-tasking in JASMIN then, refers to the simultaneous execution of computations within an address space, on a given processor. The coordination of these events is handled by the Kernal. Numerous modules may run simultaneously on the same processor; the Kernal is responsible for scheduling the activities of these modules.

The remaining software components, implemented directly on the Kernal, are shown in Figure 1. The lowest level is the Intelligent Store (IS) which acts as a page manager. Page requests are made via the Kernal using logical page identifiers. The IS resolves these logical identifiers into physical block addresses in disk storage.

---

2 While a majority of the research involved in this project is confidential to the shareholders of MCC, sufficient non-proprietary information was available to include the design in this report.

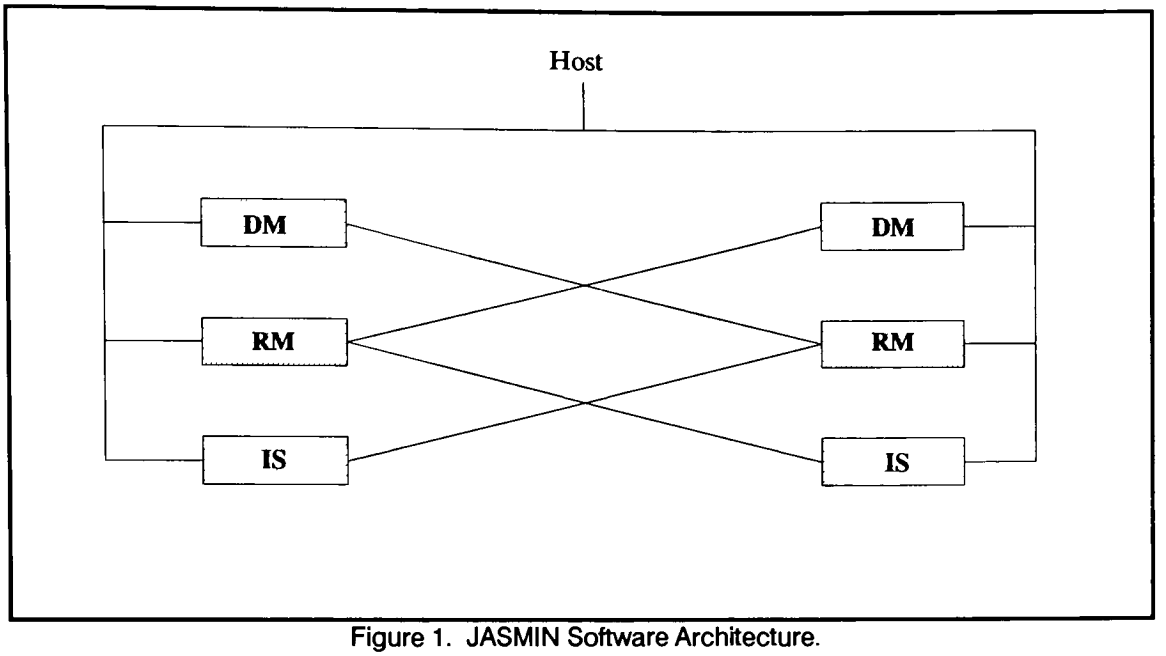

Figure 1. JASMIN Software Architecture.

The next level, Record Manager (RM) provides the mechanism for accessing data stored in records on the pages mapped by the IS. Operations carried out by the RM are limited however, to those that require a single scan of one relation (e.g. a select operation). This is because data retrieval requests in JASMIN are associative; a relation is scanned for records meeting certain selection critieria.

Finally, the Data Manager (DM) is the software component with the highest level of abstraction. Query processing occurs in the DM in a pipelined fashion, allowing for more complex operations against the relations, such as joins and aggregations. In the DM, specialized modules accept data streams which are the output of one or more RMs. These streams may then be sorted for example, and joined to produce the results for a specific query.

B. Hardware Components

JASMIN effectively has a two-level hardware system architecture as shown in Figure 2. The first level, called a node, is a grouping of processors on a high-speed bus. Nodes can be attached at the next level to a Local Area Network (LAN). This configuration is intended to support databases distributed over more than one node. A single node without the LAN attachment could also be implemented. The primary requirement for the type of processor that can be used in a JASMIN configuration is that it support addresses of 24 bits or higher. The Bell prototype used Motorola 68000 processors, but Intel 80386 processors [SEYB 86] would likely serve as well.

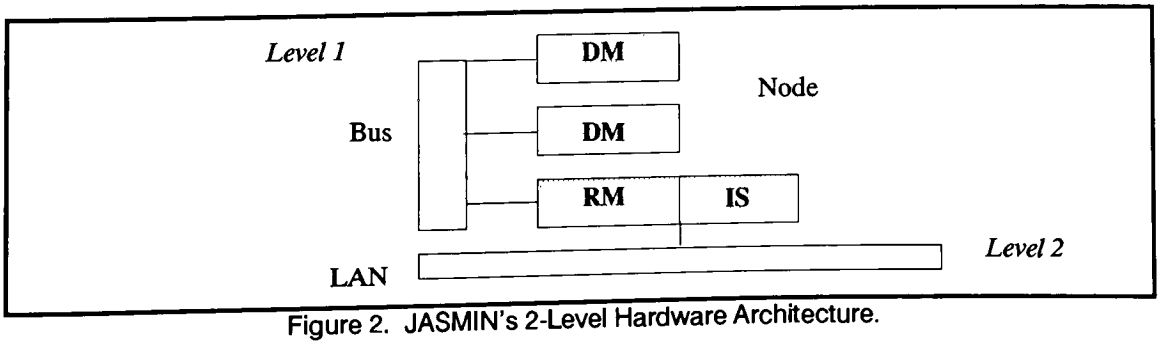

Figure 2. JASMIN's 2-Level Hardware Architecture.

The designers of JASMIN were faced with the decision of where to place the various software components on the system's processors. One option was to place all of the software components on each processor; another was to run each component on its own processor. A compromise between these two alternatives was to place the RM and IS components on the same processor and the DMs on separate processors. Since it was anticipated that IS processing would be I/O intensive and that of the RM, CPU intensive, the two might coexist without significantly degrading performance. In addition, messaging between IS and RM components was more likely to be heavier than between RMs and DMs; the physically close proximity of the former pair might enhance their inter-communications.

The researchers discovered however, that optimal placement of software components was directly influenced by the application running against them. For example, given an application with simple query transactions referencing a common portion of a database, it was more advantageous to have the entire database serviced by a single IS component on a separate processor. The RM and DM components could reside on the same processor since, for simple queries, the DM would perform very little work. For applications where references were made to diverse database portions, it was found that the data was best serviced by multiple ISs, each sharing its processor with an RM.

The JASMIN design has been perhaps the least interesting of all of those covered in this chapter. It offers virtually no novel approaches to database processing and it is uncertain exactly how much data such a machine would be capable of processing efficiently. Its reliance on the nature of the application to determine proper placement of its software components is not a workable prospect for the type and volume of processing handled by the DBC/1012. It is significant however, that this design embraces the concepts of an expandable modular configuration and the use of off-the-shelf parts.

III. GRACE

The design of the GRACE database machine [KITS 84], [FUSH 86] on the other hand, has the potential of becoming Japan's challenge to the DBC/1012. It specifically addresses two of the most critical performance factors regarding database machine technology: the overhead involved in controlling the execution of relational operations in parallel and the problem of I/O bottlenecks. The prototype of the DIRECT database machine [DEWI 79] exhibits the severity of performance degradation when these two components of database processing are not refined. GRACE's developers have dealt with the I/O bottleneck problem by means of a highly efficient disk system, the details of which can be found in [KITS 86]. Here, we will examine the architecture of GRACE, its task control mechanisms and the concept of "data stream oriented processing".

GRACE's developers analyzed the performance of DIRECT and determined that the overhead associated with the activities of the controller were the source of that design's problems. They further concluded that DIRECT's "page-level granularity" was the cause of that overhead. Recall from the discussion of DIRECT in Chapter 2 that the query processor modules executed operations against portions of relations that were temporarily stored in fixed size page frames of 16 KB each. Coordinating the movement of these frames between the query processors, the CCD memory modules and the mass storage modules placed demands on the controller such that over one-half of its processing time was consumed by these activities.

The designers of GRACE decided that reduction of this overhead on the controller could not be realized by adding more or faster processors. Rather, they proposed that the overhead had to be eliminated altogether. They attempted to accomplish this by way of efficient system software and by implementing a less restrictive processing granularity.

The concepts of "data stream oriented processing" were adopted, along with task-level granularity. The former held that the execution of relational operations should overlap the transfer of data; that is, operations should be performed against the data as the tuples are moved into the processing

modules, not after they are all assembled there. Under task-level granularity, a page full of tuples from a relation is no longer the unit upon which operations are performed, but rather the entire set of relevant tuples in a relation are acted upon. The execution of the required operations against this set of tuples is referred to as a "task". The controller is invoked only at the beginning and at the end of the task, providing the resources necessary to complete it.

The authors created an analogy in keeping with the Japanese style that conveys an intuitive picture of both task-level granularity and data stream oriented processing. They describe a task as consisting of water and a river; the water is the stream of data, the river is the execution environment for the particular operation - the required machine resources. A database operation then, is the stream of data flowing through its execution environment; the water flowing through the river.

From a hardware perspective, the GRACE architecture consists of four types of modules: processing modules (PM), memory modules (MM), disk modules (DM) and control modlues (CM). The PM and MM components are connected, as shown in Figure 3, by a multi-channel bus network referred to as the processing ring; DMs and MMs are connected by the staging ring. Tuples from relations stored in DMs are moved, or "staged", into the MMs by way of the staging network. This is the "staging phase" of query execution in GRACE. Data streams generated in the MMs flow into the PMs through the processing ring. This takes place during the "processing phase".

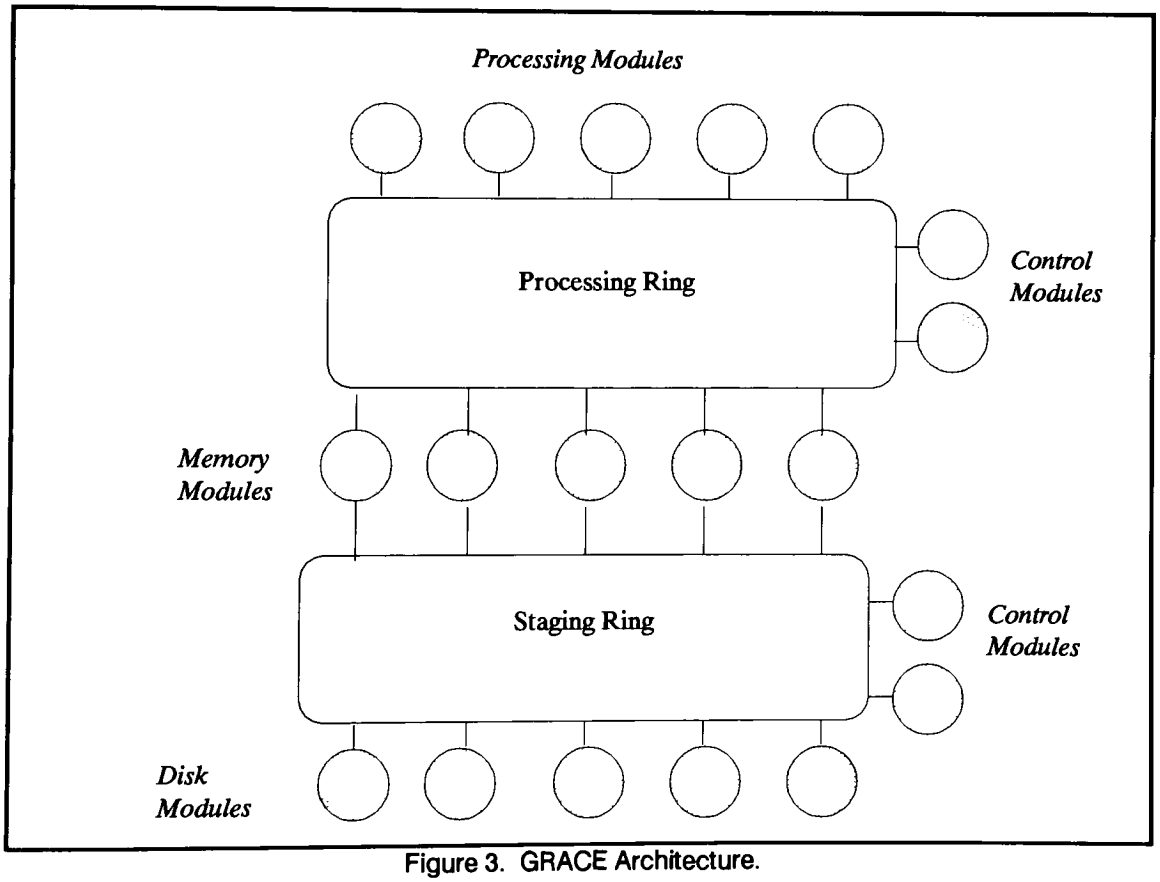

Figure 3. GRACE Architecture.

A GRACE Disk Module is shown in Figure 4. During the first phase of query execution, the staging phase, the CM sends a command to the DMs which have the relation(s) required by the query. Once the DM has been activiated by the CM, the tuples are retrieved from disk, their addresses having been resolved by the DM's Disk Controller component. The Filter Processor performs a kind of preprocess-

ing function against the tuples, on-the-fly as they are read off the disk; it removes those tuples that do not meet certain selection criteria, also specified by the CM.

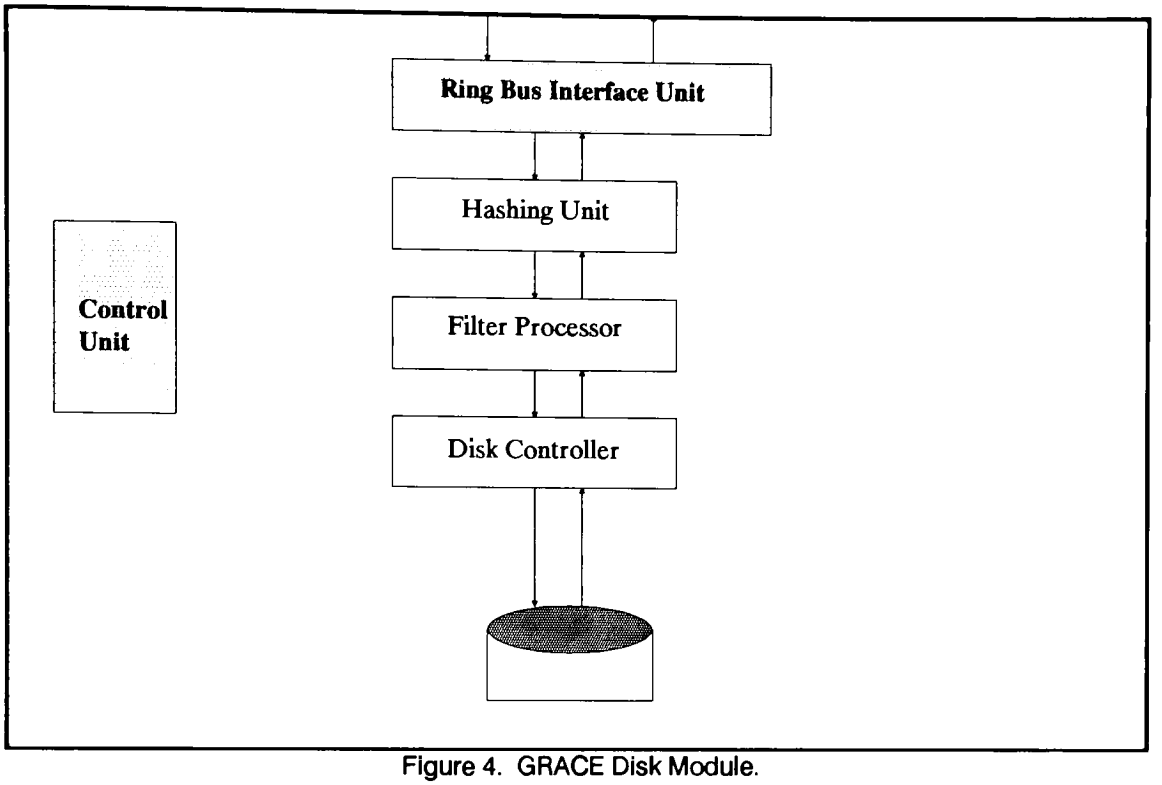

Figure 4. GRACE Disk Module.

Qualifying tuples are forwarded to the DM's Hashing Unit where a hash value is dynamically added to each tuple. The hash value depends on the value of a preselected attribute in the tuple. The hashed data streams from each DM are then distributed according to their newly acquired hash value over the staging ring to the MMs at the end of the staging phase.

In the MMs, tuples are stored according to their hashed values in "buckets" or "clusters". Any one MM will contain a number of different clusters; all clusters are distributed across a number of MMs. During the processing phase of query execution, the PMs gather the clusters from the MMs and perform the necessary operations.

A sinlge PM is assigned to process the tuples associated with a particular cluster. Since numerous MMs contain tuples from the same cluster, the PM "visits" each MM, gathering the appropriate tuples. Those tuples are assembled into a data stream by the MM and the stream sent through to the PM's Sorting Unit (see Figure 5). The sorted data stream emerging from this unit enters the PM's Tuple Manipulation Unit where the required relational algebraic operations are performed. If further processing against the tuples is required, as may be the case in very complex query execution, the PM's Hashing Unit may be invoked to further hash the resulting tuples over those attributes involved in the next operation.

This clustering feature of GRACE's design lends a great deal of efficiency to the processing of complex queries. Join operations for instance, are not performed on all of the tuples from each relation to be joined. Processing time using this method is proportional to the product of the cardinality of those relations. Instead, the join process is reduced to a number of small joins, performed on the

appropriate clusters from each relation. The processing time here is proportional to the cardinality of the buckets involved.

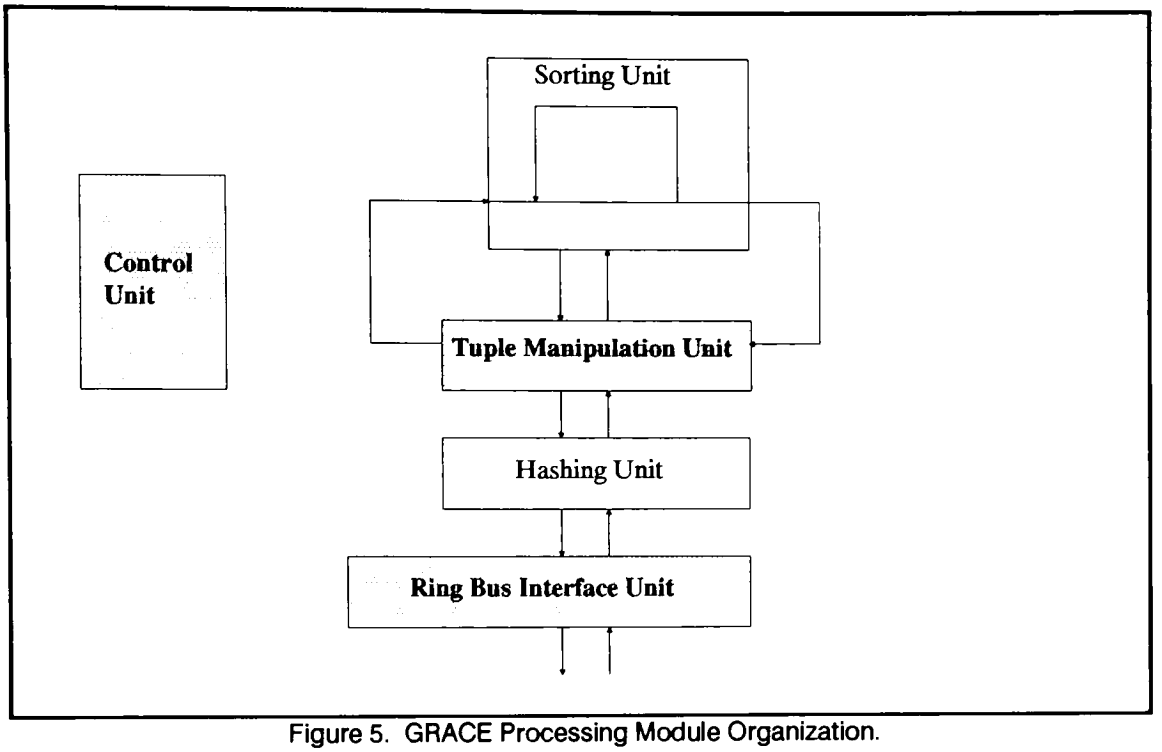

Figure 5. GRACE Processing Module Organization.

For example, consider the full join against two relations, A and B, each containing 1,000 tuples. The time required to process this join is proportional to the product of the cardinality of the two relations (i.e. 1,000 x 1,000 = 1,000,000). Using GRACE's clustering feature, the tuples from relation A might be distributed over a hashed value into 10 clusters of 100 tuples each; assume the same distribution for relation B. The total processing time for these 10 small joins (i.e. joining the nth cluster of A with the nth cluster of B, where n assumes the values from 1 to 10), is based on the product of the cardinalities of the clusters themselves; that is, 10(100 x 100) = 100,000. This represents a considerable reduction in processing time. See Figure 6 for an illustration of this concept.

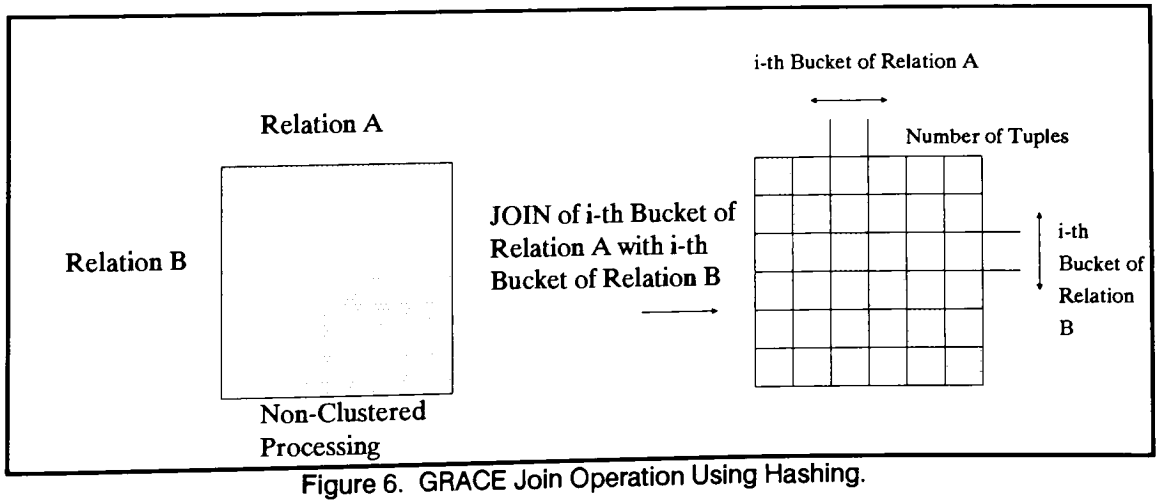

Figure 6. GRACE Join Operation Using Hashing.

It is true that all queries do not require complex operations like the joins discussed above. GRACE's design is flexible enough to conserve on the utilization of hardware components for simple queries. Two types of tasks are defined in GRACE, each requiring different machine resources during their execution. "Read" tasks are those that directly access the database - for example, selections. These tasks are executed entirely within the staging phase and need not proceed to the processing phase. The execution environment for read tasks consists of those DMs that store the required tuples, a set of MMs in which to store the results of the task (a "sink space") and a set of channels on the staging network to present the selected tuples.

The second type of task, internal tasks, are comprised of more complex operations like joins and aggregations. These tasks are completed in the processing phase and have an execution environment that includes sets of MMs to store operand data and results, a set of PMs to carry out the operations and a set of channels on the processing network.

In general, GRACE has many features which make it an intriguing design. Data continuously flows through the system in streams and it is upon these streams that processing takes place. Hashing is prevalent and is fundamental to the reduction in overhead throughout the system. This machine can handle the execution of complex queries by reducing them to manageable operations performed in parallel against small clusters of data and combining the output of those operations to produce overall query results. The configuration is expandable, allowing more hardware modules to be added as the requirements of the system grow. GRACE has the potential to be the DBC/1012's primary competitor. If this is to be the case, indicators will likely be seen in the next few years.

IV. GAMMA

The designers of DIRECT have developed a new database machine prototype, GAMMA [DEWI 86], based on the knowledge and experience they gained from their previous endeavor. The GAMMA project is underway at the University of Wisconsin and has received funding from both the United States Department of Energy and Digital Equipment Corporation. Citing control overhead as a primary factor contributing to DIRECT's poor performance, the developers of GAMMA, like those of GRACE, have addressed this issue.

The current GAMMA prototype is comprised of twenty VAX 11/750 processors with 2 MB of memory each. Eight 160 MB Fujitsu disk drives are attached to eight of these processors and store the databases. The processors are interconnected by an 80 megabit per second token ring, which in turn connects to another Vax machine running Berkeley UNIX. This machine serves as the system's host. See Figure 7.

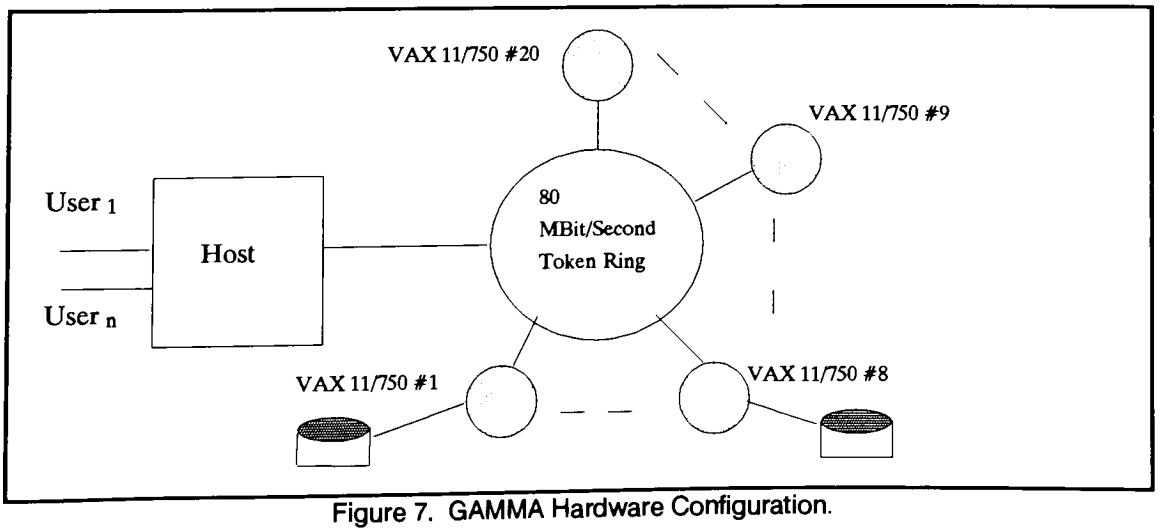

Figure 7. GAMMA Hardware Configuration.

There are some similarities between the design philosophy of GAMMA and that of the DBC/1012. Both utilize conventional disk drives and associate a processor with each. All processors in the system are attached to an interconnection network - the Ynet in the DBC/1012, the token ring in GAMMA. Also, GAMMA offers a number of schemes for distributing the tuples of a relation across the disk drives, one of which is the same as that employed in the DBC/1012 (see below). One similarity between GAMMA and GRACE is that both use tuple streams as the object of processing; referred to as "data stream processing" in GRACE, and as "dataflow query processing" in GAMMA.

As mentioned above, GAMMA supports a number of methods for distributing data across the disk drives. These are: round robin, hashing, range partitioned with placement determined by user-specified parameters and range partitioned with uniform distribution. The first of these, round robin, is the default for GAMMA. As a relation is loaded with tuples, they are distributed one at a time to each drive. With the second strategy, incoming tuples are hashed over a predetermined key attribute, the hash value determining to which drive the tuple is assigned. This is essentially the technique used in the DBC/1012.

The last two distribution methods involve range partitioning; that is, tuples are distributed across the drives based on certain ranges of key values. In the first of these two strategies, the user may specify the range used to determine the distribution. In the second, tuples are initially distributed in a round robin fashion, sorted on the key attribute. The sorted relation is then redistributed among the drives so that an equal number of tuples from a relation reside on each disk.

The components of GAMMA's system software are called "processes", the most important of these for this discussion being the scheduler, operator and query manager processes. Queries executed in GAMMA are first compiled into trees of operators. Each operator is executed by at least one operator process. A query manager process is associated with each GAMMA user. The scheduler controls the execution of complex queries and coordinates the activities of the operators. A query execution example illustrate the roles of these various processes.

The query manager (see Figure 8) sends the compiled query to the scheduler which in turn activates an operator at each processor involved in executing the query. A stream of tuples is input to the operator, along with an "operator control packet" provided by the scheduler. The operator performs its functions against each tuple as it is read from the stream. The output from the operator is also a stream of data, hence the phrase "dataflow query processing".

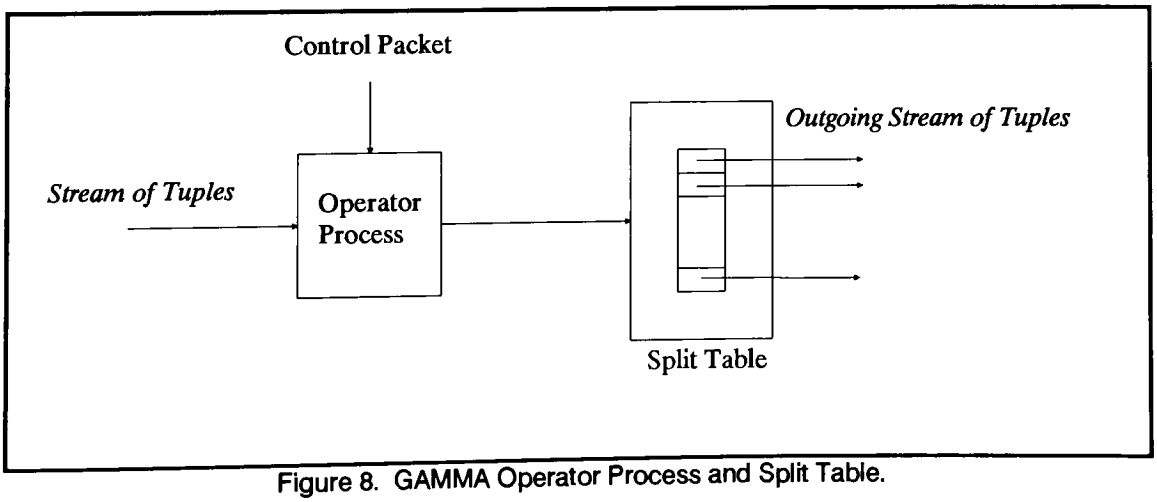

Figure 8. GAMMA Operator Process and Split Table.

As tuples exit the operator, they flow through a "split table" which applies a hash function to an attribute on the tuple, serving much the same purpose as GRACE's clustering feature. Tuples are assigned by way of the split table to clusters upon which for example, "small joins" may subsequently be performed.

When the end of the input stream is reached, a control message is sent to the scheduler, indicating that the operator is finished. Only three control messages are involved in the activites of an operator: the control packet sent by the scheduler, a completion message sent to the scheduler and an "end of stream" message sent to the next process to receive the current output stream. Aside from these, the operator process is self-controlling, minimizing the control overhead experienced in the design of DIRECT.

The GAMMA design is very simple. A number of physical similarities exist between it and the DBC/1012; the dataflow concept is also found in the GRACE design. GAMMA uses off-the-shelf parts and has an expandable configuration, features common to the most recent database machine designs. It will be interesting to see if Digital Equipment Company's funding of the GAMMA research results in that company's entrance into the database machine industry.

V. BUBBA

The final design to be discussed in this research is BUBBA, a parallel database machine being developed at the Microelectronics and Computer Technology Corporation (MCC) in Austin, Texas [WILK 86], [COPE 87]. The objective of this project is to deliver by the mid 1990's a database and knowledgebase management system with price/performance improvements of one to two orders of magnitude over today's conventional general purpose mainframes. BUBBA is intended to support a wide variety of applications involving very large databases. Since the nature of MCC's research is highly confidential, limited amounts of documentation on this project are available in the public domain. An overview of the hardware architecture is unrestricted however, as well as some important design concepts which will be discussed below.

BUBBA's hardware architecture is comprised of three major components: intelligent repositories (IRs), interface processors (IPs) and an interconnection network. The IPs control interactions with the users of the system. The IRs perform the greatest amount of work and are themselves composed of five subunits: main processor, disk controller, communications processor, main memory and a disk storage unit. The interconnect, as in GAMMA and the DBC/1012, forms a network between the IRs and the IPs (see Figure 9).

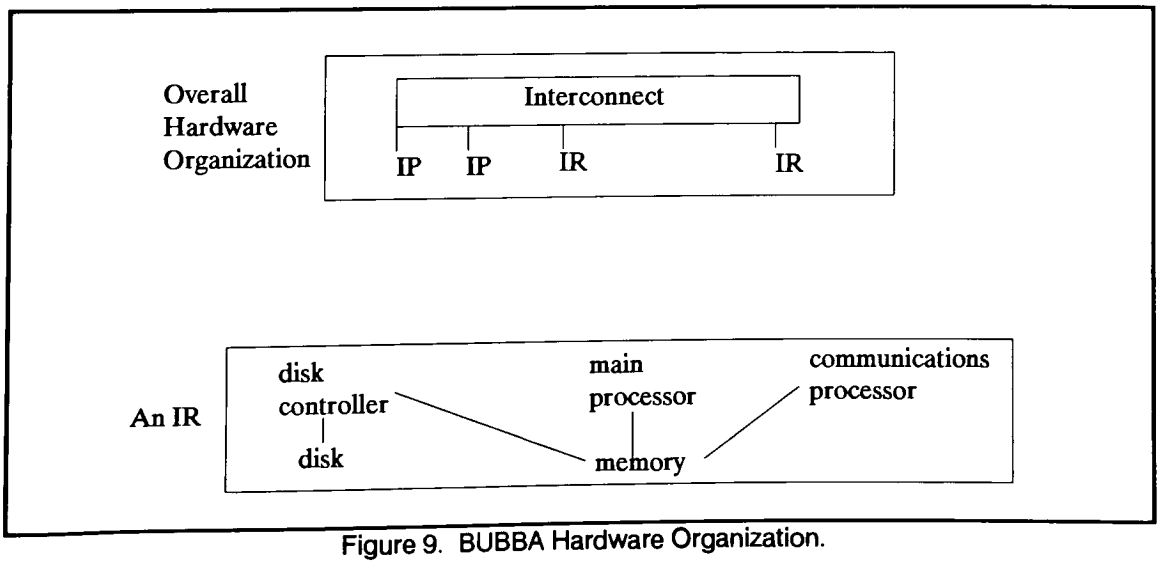

Figure 9. BUBBA Hardware Organization.

The IRs are virtually autonomous in BUBBA's "share-nothing" environment. One of the designers' primary goals is to make the IRs as small as possible, yet retaining their required functionality. The prototype model can support from 50 to 1,000 IRs in an effort to increase parallelism and reduce the impact of individual failure on overall performance.

Throughput in a parallel system depends, according to the BUBBA developers, on two factors. First is the total amount of work to be done; second, the amount of processing capacity lost by poor load balancing. This refers to the distribution of the workload among the system's processors. As discussed in previous sections, both GAMMA and the DBC/1012 utilize a hashing algorithm for distributing tuples across the processing components - a "horizontal partitioning" scheme. The designers of BUBBA analyzed the issue of data placement and determined that three decisions needed to be made; the number of nodes over which to distribute the data, which nodes to involve and whether to process the data on disk or in a memory cache.

BUBBA's data placement schemes incorporate "variable declustering" as opposed to what the BUBBA designers call "full declustering" in GAMMA and the DBC/1012. Relations are declustered according to a key attribute value and different relations are distributed among different numbers of IRs in order to optimize performance. The number of IRs over which a relation is distributed is termed the "degree of declustering" or "DegDecl". Only those IRs that actually contain the desired tuples are involved in processing the request.

The concept of variable declustering is important and may eventually be incorporated in the DBC/1012 and GAMMA designs. There are similarities among these designs, such as the interconnection matrix. The final results of the BUBBA project when fully disclosed will no doubt add much to the advances in database machine technology.

VI. Teradata

In many ways the future of database machines depends on the continued soundness of Teradata Corporation - its ability to attract new customers while satisfying the growing needs of existing ones, to introduce and successfully market new products to complement the DBC/1012 and to enhance that machine by delivering more processing capacity for less money than competing mainframes can.

There is a high probability that even if Teradata should fail in these endeavors, research in the field will continue. Indeed, we may see some of the database machines designs reviewed here, such as GAMMA or BUBBA, marketed by other companies in the next few years. However, the longevity of Teradata and the testimonials of its large corporate customers will serve to validate the database machine as an alternative to software database management systems.

Teradata has benefited both from its strategies in hardware implementation and from its basic philosophies toward compatability. With regard to hardware, as has been established in Chapter 3, Teradata chose to incorporate readily available, off-the-shelf parts in its design; this is true of both its processor components and the DSUs. Its processors have been Intel chips, beginning with the 8086 in the first releases of the DBC/1012. As processor technology advanced, Teradata enhanced its product by introducing Model 2, based on the 80286 processor family and the newest Model 3 which will be available in November 1988 [TERA 88d] using 803xx processors. By making use of standard disk units rather than relying on custom designs, not only does Teradata have the benefit of readily available parts and service, it can also take rapid advantage of improvements in disk technology.

Teradata's hardware strategies have been coupled with its continued "commitment to compatability" [TERA 88a]. Recently, the company adopted a policy of Shared Information Architecture (SIA) [BRAU 87a], the primary goal of which is to enable Teradata customers to access the data on the DBC/1012

through a variety of processing environments. To date, such access can be made from IBM mainframes running either VM or MVS, IBM PCs and PC-compatibles, Digital Equipment Company VAX and Micro VAX systems; as well as systems from Metaphor Computer Systems, Inc.; Honeywell Bull, Inc. and AT&T's 3B series [NEE 88], [TERA 88a]. Communications protocols supported by DBC/1012 via Ethernet LANs [BARR 87] include TCP/IP (Transmission Control Protocol/Internet Protocol) and ISO/OSI (International Standards Organization/Open Systems Interconnect) [TERA 88a].

Presently, Teradata has agreements with some other companies which provide a DBC/1012 interface with their own software products. These products include Intellect, a natural language query facility marketed by Artificial Intelligence Corporation; NOMAD2, a fourth generation query language from D&B Computing Services; PC/SQL-link, a product of Micro Desionware, Inc. that provides access to the DBC/1012 from IBM PC or PC-compatibles; and FOCUS, a fourth-generation language by Information Builders, Inc. [TERA 87]. Teradata has positioned the DBC/1012 in such a way that these companies' marketing strategies for their interfaces benefit not only themselves, but Teradata as well.

Teradata's SIA allows the DBC/1012 to be readily integrated into existing configurations, an attractive solution to the growth problems experienced by companies who have already made substantial investments in their computer systems. It is this very growth in the size and number of online database systems that has caused major corporations to consider the Teradata alternative. Teradata currently has fifty customers, including American Airlines, AT&T, Chicago Board Options Exchange, Citibank, K-Mart and NASA [SALO 87]. The number of installed DBC/1012 systems tripled during 1987 [BRAU 87c], reaching ninety. The company has achieved consecutive profitable quarters after the fourth quarter of fiscal 1986.

One can be optimistic about Teradata's growth and profitability, but there exist a number of factors which demonstrate considerable exposure for the company, exposure that has the potential to inhibit the database machine industry by adversely affecting its sole contender [SALO 87], [BRAU 87c].

First, Teradata has a very short history, having been started less than ten years ago in 1979. Through its fiscal year ending in June 1984, the company was considered to be in its development stage - a highly speculative, upstart company, relying on the investments of several venture capitalists and on the promise of its products' designs. The first DBC/1012 was released in October 1984. Still, the company experienced consistent losses through the end of fiscal 1986.

Second, Teradata depends on a few, very large accounts for its revenues. During the 1987 fiscal year, 44% of the company's total revenues came from just three accounts: K-Mart, AT&T and Citibank. These same customers accounted for over 46% of Teradata's revenue from its inception in 1979 through the fiscal year ending June 1987. The loyalty of the company's existing customers is critical, since that base provides 67% of Terdata's revenues. Analysts feel that this reliance on a few customers could be eliminated if the company could attract five or six new customers per quarter. It is clear that Teradata's future hinges largely on support for existing customers and on alluring new ones.

Third, the DBC/1012's processor components are supplied by one company, Intel Corporation. Should that company be unable to supply Teradata with its required parts, the latter could be severely adversely affected. Finally, Teradata is vulnerable to the competition presented by other vendors, particularly IBM. As mentioned above, Britton-Lee's BL8000 database machine is not viewed as substantially dangerous because the company is one-third the size of Teradata and its product does not support MVS [BRAU 87d]. IBM on the other hand, has resources, experience and a customer base that virtually dwarf Teradata's. Indeed, IBM's ESA architecture and its incorporation of DB2 Release 2.0 poses the single greatest threat to the future of Teradata and to database machine technology, in general.

Still, Teradata continues to grow, as does interest in database machines. Some expect the company to reach annual sales of $100 million during 1989, placing Teradata in a category with Cray Research and Tandem Computers, now considered two of the most successful computer companies today [BRAU 87c]. Teradata is looking toward the future; it plans to improve the DBC/1012's hardware, has announced three major new products and has set some general directions.

As mentioned earlier, Model 2 of the DBC/1012 release utilizes 80286 processors. The next step for the company was to incorporate the new 80386 chips; support for this new Model is available with Release 3.1 of the Teradata software and upgraded with Release 4.0 [TERA 88d]. Briefly, the 80386 is a "true" 32-bit microprocessor; that is, not only do its registers and addresses contain 32 bits, its data paths do as well. The chips provide clock speeds of up to 20 MHz, compared to the 80286 which generally runs at 8 MHz. At 16 MHz clock speeds, the 80386 can process up to 3 MIPS, in contrast to the 80286's 1 MIPS. This should deliver transaction response times of less than one second on the new DBC/1012. Other features of the new processor which Teradata has taken advantage of are its 4 GB physical addressing and 64 terabyte (TB) virtual addressing capabilities; the 80286 can address 1 MB and 1 GB in these categories, respectively [SEYB 86], [BRAU 87d].

In addition to the new processor technology, Teradata has incorporated higher capacity disk drives and a new disk drive interface, the Intelligent Peripheral Interface (IPI) which can provide each AMP with more than two drives, the present maximum. IPI is expected to improve the DBC/1012's data redundancy capabilities. Teradata's design decision to utilize standard disk units has allowed it to take advantage of these new technological advances.

New benchmarks performed against Model 2 and Model 3 DBC/1012s running Release 3.1 of the Teradata software show up to 73% less CPU consumption and elapsed time improvements of up to 12 times in the new model (see Figure 10) [TERA 88d]. The company also claims price/performance advantages over DB2 Version 2 of 4.3 to 1. [TERA 88d], [TERA 88e].

DBC/1012	Model #	Software Release
A	2	3.0
B	2	3.1
C	3	3.1

Tabulation of Results of Decision Support Query Suite from Several Teradata Customers

	Elapsed Time	Elapsed Time (x Faster)		CPU Time (x Faster)	
	A (Min:sec)	C:A	C:B	C:A	C:B
Banking Customer 500K rows	17:40	6.7	2.0	11.0	2.7
Insurance Customer 500K rows	14:15	11.3	1.7	27.0	2.4
Retail Customer 500K rows	6:03	3.0	2.7	3.9	3.3
Pharmaceutical Customer 500K rows	7:23	11.7	1.5	36.5	2.6

Figure 10. Performance Improvements in DBC/1012 Model 3.

Teradata has also announced three new interface products that are a part of its SIA strategy: Sun Interface, Transparency Series and the Teradata Service Manager [TERA 88b], [TERA 88c]. The Sun Interface will allow the workstations produced by Sun Microsystems, Inc. to access the DBC/1012 directly across an Ethernet LAN via Teradata's Communications Processor (COP), using the TCP/IP network protocol. Sun workstations running UNIX 4.2 release 3.4 and above are supported by this new interface. As with workstations presently connected through COPs, a single workstation may access multiple DBC/1012s; multiple workstations on the same LAN may communicate with a single DBC/1012. Initially, the Sun Interface will provide a C preprocessor, the Teradata Director Program (TDP), Call-Level Interface (CLI) and the Batch Teradata Query facility (BTEQ), although other capabilities are planned for future releases. The Sun Interface should be available in 1988 and will cost approximately $4,000.

The second announcement is for Teradata's Transparency Series which will allow SQL requests from application programs written for other database management systems to be processed on the DBC/1012. The goal of this product is to provide a means for Teradata customers to retain their current systems of programs, but have them access databases on the DBC/1012 instead of the DBMS for which they were originally written. Initial Transparency Series products include support for QMF, SAS and Metaphor, as well as preprocessors for COBOL and PL/I. Eventually, a Transparency Series product to support DB2 applications will be available as well. The price for this product is estimated at $25,000.

Finally, Teradata has entered into an agreement with Applied Data Research, Inc. (ADR) for a product called the Teradata Service Manager (TSM) [TERA 88b], [TSM 88] which provides a link between ADR's Datacom/DB environment and the DBC/1012.[3] Calls to Datacom/DB are translated into SQL statements and sent to the DBC/1012 for processing.

TSM runs with the current release 3.0 for the DBC/1012 and will run on release 3.1 as well. It operates under MVS or MVS/XA. Three ADR products are required to support TSM: Datacom/DB Release 7.5A or higher, Datadictionary Release 2.4A or later and InfoReach Distributed Data Manager Release 2.0 or higher. Applications developed under three ADR products can access DBC/1012 databases by means of the TSM with virtually no changes. These products are IDEAL Release 1.4 or later, ADR/DL Release 2.2 or higher, and Dataquery Release 4.0 or higher. For installations in the process of converting from the Datacom/DB environment to a DBC/1012, a determination is made by the InfoReach DDM component whether the data requested is located in a Datacom database or if it resides on the DBC/1012; the user need not be concerned about the location of the data. The request is routed to the appropriate site.

Currently, Release 1.0 of TSM provides read only capabilities against DBC/1012 databases; future TSM releases will provide update capabilities [BUI 88]. For now, TSM is appropriate for installations with large, stable databases serving retrieval needs only. Active databases (i.e. those requiring updates) should remain in the Datacom/DB environment. Utilities are provided by both ADR and Teradata to unload and load data across the systems. So, if data residing on a DBC/1012 database needed updating, it could be unloaded from the DBC/1012, loaded into a Datacom/DB database, updated, unloaded from Datacom and loaded back into the DBC/1012 database.

---

3 ADR, with TSM and Release 2.0 of IDEAL (see Chapter 4), has planned business strategies to take advanatage of both IBM's ESA as well as the DBC/1012.

Since ADR's Datacom/DB is probably DB2's primary competitor in processing very large relational database, TSM may prove to be Teradata's edge against IBM in that arena. This product automatically gives Teradata the advantage over IBM in application tools by virtue of ADR's well established Datadictionary, IDEAL, ADR/DL and Dataquery products. DB2's lack of similar tools, except for IDEAL 2.0's DB2 access capabilities, has hindered its ready acceptance by many. Current ADR customers will probably find the processing capablilites of the DBC/1012 an attractive option to mainframe capacities. It will be interesting to see whether Datacom/DB falls by the wayside in favor of the DBC/1012, once TSM achieves full update capabilties.

Teradata has other development projects which have not yet been formally announced. Interfaces are planned for Apple Computer Inc.'s MacIntosh computers, Tandem Computer Inc.'s Non-Stop line, Unisys' 1100 series and Hewlett-Packard Co.'s 3000 computers [NEE 88]. The company is also working on an interface for Citibank to IBM's Transaction Processing Facility (TPF) [BRAU 87d].

Teradata's future looks promising, but it remains highly speculative. The company has a definite advantage in the quality of the DBC/1012 and its related products. It has won over a substantial number of customers in its brief operating history, many of these being Fortune 500 companies. For the most part, Teradata's customers are pleased with the performance delivered by and with the cost of their systems [KULL 86]. The company has targeted some potentially lucrative markets with its interfaces and transparencies.

IBM however, will play a major role in the future of Teradata and in the future of database machine technology. In all likelihood, IBM will achieve its goals of deliverables with the combinations of DB2, MVS/ESA, the Sierra running the with ESA/370 processor architecture and ultimately the Summit lines. One question is, will IBM succeed in providing the tools its users require; tools like ADR's Datadictionary, Dataquery and IDEAL, which Teradata has found a way to incorporate? Another asks, how soon will all of these tools be made available to IBM customers? Given enough lead time, Teradata can capture business away from IBM; the business of installations whose database processing needs simply cannot wait for an IBM solution. If Teradata can secure enough customers and can continue to deliver products of equal and even better quality, it can survive and may well prosper.

The next two years will tell much. In that time, many currently announced products will become available from IBM, Teradata and ADR. We may also see a marketable database machine evolving from the GAMMA or BUBBA projects, not to mention the possibility of a Japanese design. In addition, database machine research will continue; as technologies advance, more and more sophisticated designs will emerge.

VII. Database Machine Design Conventions

A number of important issues have evolved from within database machine research. Common threads can be seen in the latest designs, demonstrating that some concepts have become generally established as conventions in the field. The first of these is an adherence to relational database technology. Today it seems a given that relational databases are here to stay. Indeed, IBM in contrast to its earlier stands [HESS 84] has begun recommending DB2, not IMS, as an appropriate DBMS in over 90% of its customers new applications [BABC 88b]. Future database machine designs will continue to refine their relational database techniques.

Second, off-the-shelf parts have effectively become the norm. Improved technologies involving processors, disk drives and disk storage, as well as memory can be taken advantage of by database machine designers if they continue to use readily available products instead of custom made ones. The cost of the former products is steadily declining, while their performance capabilities continue to increase. Also, these parts are becoming more and more standardized, making it easier to produce and implement them. Custom designed components on the other hand, have proven to be more

costly, less readily available and lagging behind the technological advances of standard parts. Some older database machine designs, like RAP.1 and DIRECT, have become obsolete in light of these factors.

A third concept that is generally accepted for database machine designs is parallel transaction processing. Distributing the work to be done across a number of processing modules has proven instrumental in achieving high transaction processing rates. Optimizing the algorithms for such distribution is the topic of a number of research projects. [COPE 87], [BABA 87].

Along with the notion of parallelism is the principle of modular, expandable configurations. One of the DBC/1012's biggest selling features is the ability for a customer to begin with a minimally sized machine and to expand its capacity as the installation's needs grow. The four database machines reviewed in this chapter also adhere to expandable configuration designs.

Three other concepts have gained acceptance in the database machine field: interconnection networks, multi-user environments and workstations. Beginning with DIRECT reviewed in Chapter 2, the feature of a hardware network connecting the processors in a database machine has been adopted in the designs of DBC (in the DBCCP module), GAMMA, BUBBA and the DBC/1012. The function of the interconnection network is to coordinate the flow of information throughout the entire system, relieving individual processors from this sort of overhead.

Support of environments that have multiple users is a necessity in today's computer installations. While single user systems can be advantageous in the early design stages of a new machine, realistically database environments - especially very large database environments - usually require simultaneous access by hundreds of users. Any designers who seriously intend to market their machines, must incorporate this feature into their systems.

Finally, the workstation is becoming a very popular computing mechanism. The same microprocessor technology that has given Teradata increased capacity in the DBC/1012 has likewise benefited the workstation market, making workstations more and more powerful. Memory capacities in today's workstations rival those of some mainframes. Teradata has introduced its COP and Sun Interface products specifically designed to take advantage of this technological niche. The company has made a mainframeless system one of its goals [ZENG 87]. While it is doubtful that workstations will ever replace the mainframe completely, their role must be taken seriously and accounted for in any credible database machine system design.

VIII. Outstanding Issues

While the above issues seem to have gained a general acceptance in the fields of database and database machine technologies, there are others which have not yet been resolved satisfactorily. The most important of these are the lack of complete referential integrity, user tools and interfaces and finally, data dictionary standards.

A. Referential Integrity

As discussed in Chapter 1, referential integrity has remained an enigma since the inception of relational databases. IBM has recently announced that Release 2.0 of DB2 will further support referential integrity constraints [IBM 288-196]. Some are concerned however, that it still cannot guarantee integrity in all situations [BABC 88a]. Until this matter is resolved, program intervention will still be required to flag instances where integrity is not maintained.

B. User Tools and Interfaces

The need for sophisticated user tools and interfaces is another outstanding problem. Teradata has addressed this by providing the utilities and preprocessors described in Chapter 3. There remains a great demand for more tools however, and for improvements to those that do exist. Other database machine designs discussed in this research are silent for the most part when it comes to the subject of user tools and interfaces.

From a practical standpoint, database end users, while functioning with a growing amount of computer literacy, are not generally technically oriented, nor should they be expected to be so. They generally wish to be removed as much as possible from the operational aspects of the systems they use. Data and system availability are of critical concern, as is prompt data retrieval. These needs are at the apex of that which a database processing environment must deliver; this delivery is directly dependent on fully functional and efficient tools and interfaces.

1. System Tools and Utilities

To achieve acceptable levels of overall system availability and response time, this functionality of tools and interfaces is required beginning at the most technical levels. First, system analysts must have adequate performance monitoring tools both for problem solving and problem avoidance. Many potential system difficulties can be avoided by ongoing monitoring and tuning processes. But efficient utilities are essential because of the time-consuming and detailed nature of this type of activity in very large installations. Further, when problems do occur, proper tools must be available to pinpoint the cause and implement a solution as quickly as possible, affecting the fewest users. Downtime for very large databases carries a very large price tag.

Second, efficient data backup and restore procedures and data recovery processes are essential. Teradata has provided the fallback AMP feature (see Chapter 3) in which the DBC/1012 automatically creates a second copy of specified data. IBM has a similar facility in its Dual Copy on the 3990 Controller (see Chapter 4). Such facilities should be continually refined so that they never generate overhead that may prove detrimental to overall system performance. From the CBOE benchmark in Chapter 3, it was observed that the DBC/1012's backup and load utilities on very large databases can be extremely time consuming. More and more large installations require twenty-four hour availability. Taking precautionary backups and, if the need arises, restoring data from a backup must execute quickly and efficiently to meet this availability demand. Finally, facilities for forward and backward recovery must provide fast, accurate results. Teradata has only begun the implementation of permanent journaling in its DBC/1012 Release 3.0. Further refinement of this, as well as of the recovery process itself, is necessary.

The above issues seem inadequately emphasized in the database machine designs reviewed in Chapter 2 and at the beginning of this chapter. Yet, in real world situations, these factors can carry as much weight as response time in evaluating a database system. They are absolutely essential in any credible database machine design.

2. Programming and Query Tools

The next level of user tools and interfaces required if users are to be provided with the functionality they demand includes programming languages and query facilities. Fourth generation languages, like relational databases, are generally accepted. IBM's fourth generation language, Cross System Product (CSP), is under development for use with DB2 [POLI 86], [MARG 88a] but it is in need of a considerable amount of refinement and probably will not deliver the functionality required of it for some time. The same is true for IBM's Query Management Facility (QMF) [MARG 88a]. ADR's IDEAL seems to have the become the most popular among users of very large databases; Release 2.0 provides access capabilties to DB2 databases. TSM supports IDEAL and Dataquery access to

DBC/1012 databases, but until facilities are provided to give TSM update capabilities, this access may prove of little use in most installations.

Programming language and query interfaces are critical for database users and yet they appear to be lagging behind the advances made in hardware technologies. Without the former, the capabilities of the latter are not available to the users to the extent they should be. This lack of availabilty of fully functional fourth generation programming languages and query facilities prevents relational database use from reaching its full potiential, especially in a database machine system.

C. Data Dictionary Standards

Finally, there is a need for standards in data dictionaries.[4] Data dictionary standards remain virtually unestablished. Teradata supports its Data Dictionary/Directory [DBC/ 87] which lacks the sophistication of ADR's Datadictionary [DATAD 88]. The TSM however, will provide the DBC/1012 with an interface to ADR's dictionary product. IBM has announced plans to develop a central data dictionary for its SAA umbrella, bridging different operating systems. The first implementation is not expected until sometime in 1989 however, and even then its full potential may not be delivered for years after that [MARG 88a].

The lack of uniformity in data dictionaries prohibits the standardization of other aspects of database design and processing techniques. The Institute for Computer Sciences and Technology at the National Bureau of Standards has been working on a data dictionary standards project with the American National Standards Institute (ANSI) since 1980. The Information Resource Dictionary System (IRDS) is intended to set standards for controlling and managing databases and their related entities, using methods similar to ADR's Datadictionary. Since IRDS has some compatabilities with existing dictionaries, it is expected to be easily adopted.

The International Standards Organization (ISO) on the other hand, is developing yet another data dictionary standard based on data modeling concepts rather than on the entity relationship model of IRDS. This standard is still two to three years away from official endorsement by ISO however, and so it appears that the IRDS standard will be the first. The latter is being supported by many United States organizations and vendors such as IBM and ADR. In the international arena however, England, Canada and Japan indicate support for the ISO standard; West Germany is siding with the United States [MARG 88b]. Even if the IRDS standard is adopted here, a question remains regarding world wide standards. IBM's endorsement of IRDS may cause it to become an international standard; but Japan's support for the ISO standard may challenge that. Meanwhile, database administrators are left with a conglomeration of proprietary dictionaries, often retaining redundant information. Many are severely constrained in their efforts to consolidate their companies' applications to a uniform database system. The topic of data dictionaries remains virtually unaddressed by the database machine designs reviewed here, except in the DBC/1012.

IX. Conclusion

Database machine research has remained very active since the early 1970's. As new ideas emerged and new hardware technologies became available, the general direction of database machine designs changed. Early designs used associative memory techniques and logic-per-track devices. Databases in the business world grew however, to volumes that few anticipated.

---

4 These provide information concerning the logical structure of the databases in a system and are potentially very powerful tools for the DBA.

Associatve searches were inhibited by page sizes and I/O bottlenecks; they eventually gave way to faster, more versatile, less expensive random access memory (RAM). LPT devices became prohibitively expensive and their supply unreliable. They in turn were replaced by standard disk drives found to be more readily available, reliable, cost effective, faster and benefiting from continual advances in disk technologies.

The tremendous advance in microprocessor technology over the past ten years has done the most to change the course of database machine designs. Small, readily available and reliable, these processors have delivered memory capacity, processing capabilities and speed far beyond anything that pioneering database machine designers ever imagined, rendering most of their early models obsolete. The most recent machine proposals reviewed at the beginning of this chapter incorporate all of these technologies - RAM, standard disk drives and microprocessors.

Yet only one machine, the Teradata DBC/1012 has ever been successfully marketed in support of very large databases. Over the past four years, Teradata has succeeded in installing ninety machines in the computer centers of some of the world's largest corporations. In most cases, the DBC/1012 has satisfied their database processing needs at a lower cost and with greater speed than software database management systems running on general purpose computers [KULL 86]. Teradata has brought the database machine alternative to the attention of MIS; they are listening.

Still, IBM remains the single greatest threat to the future of Teradata, its DBC/1012 and to database machines in general. Its new ESA system with its built-in support for DB2, running on its current top-of-the-line mainframes, the Sierra and someday on the Summit machines, poses formidable obstacles. But Teradata challenges to provide the same amount of functionality as IBM for a fraction of the cost. Its ability to accomplish this will be the key to Teradata's continued success and ultimately to the success of database machines.

Glossary

- **abend** An acronym for "abnormal end" generally used in reference to the abmormal termination of a program run.

- **access path** The general sequence in which a database structure is accessed.

- **ADR (Applied Data Research, Inc.)** A software company which markets Datacom/DB, IDEAL, Datadictionary, Dataquery, the Teradata Service Manager and other database-related products.

- **AMP (Access Module Processor)** Subsystem of the Teradata DBC/1012 database machine which performs the actual manipulation of data on the disk storage units (see DSU below).

- **associative search** Database machine search technique that scans the entire database in search of data that satisfies some predefined criteria.

- **centralized control** An advantage offered by software database management systems. This allows a central administrator to exercise control over such things as security, data sharing, data integrity and data redundancy.

- **Charge-coupled device (CCD)** A type of electronic disk.

- **cluster processing** Processing concept expected to be introduced in IBM's Summit mainframe line in which groups of four to sixteen processors function together as a unit.

- **COP (Communications Processor)** Subsystem of the Teradata DBC/1012 database machine which allows connectivity to that machine directly from workstations, bypassing the mainframe.

- **DASD (Direct Access Storage Device)** Storage medium, commonly referred to as a "disk" which utilizes the properties of magnetism to store information. DASDs are the most widely used medium for database storage.

- **database** A computerized system of record-keeping. A collection of stored data used in application systems.

- **data independence** Immunity afforded to application programs by software database management systems with regard to changes in data storage structures and access paths.

- **DBA (Database Administrator)** Person or department responsible for numerous aspects of database systems including design, security and efficiency.

- **database machine** A hardware system dedicated to processing database functions, allowing such functions to be off-loaded from the mainframe.

- **database management system (DBMS)** Software interface between the actual database and the database's users. Access requests made by the users on the data are handled by the database management system.

- **Datacom/DB** A software database management system marketed by ADR.

- **DBC/1012** Hardware-backend database machine produced by Teradata Corporation. This is presently the only commercially successful database machine capable of supporting very large database Installations.

- **DB2 (Database 2)** IBM's relational software database management system.

- **DSU (Disk Storage Unit)** Subsystem of the DBC/1012 comprised of the actual DASD units.

- **emerging machine** Classification by Hsiao of database machines whose availability is contingent upon the future maturation of certain hardware and software technologies.

- **field** Column of a relational database record.

- **fixed-head disk** A disk storage medium configured with immovable (i.e. "fixed") read/write heads mounted on an arm. Generally there is one read/write head for every track on the disk.

- **foreign key** A field in a relational database table whose values must match those of a primary key in some other table.

- **formatted database management machine** Hsiao's application category of database machines. Machines in this category are typical of those found in data processing environments.

- **hardware-backend** A database machine which is separate from the mainframe computer and which performs database functions.

- **HSCI (Host System Communications Interface)** The subsystem of the Teradata DBC/1012 consisting of a software library and providing the connection between the host and the DBC/1012.

- **IBM (International Business Machines)** A leading computer hardware manufacturer, producers of the Sierra, Summit and System/38 computers, as well as the IMS (see below) and DB2 software DBMSs.

- **IDMS** A software database management system marketed by Cullinet, Inc.

- **IFP (Interface Processor)** The Teradata DBC/1012 subsystem that coordinates the flow of information between the host and the database machine.

- **IMS (Information Management System)** IBM's production database management system. This is a hierarchical database system.

- **indexing level** A dimension of Qadah's "three-dimensional database machine space" which describes the smallest addressable unit of data used as an index for making selections from the database.

- **Intelligent Controller** Classification of Hsiao's Architecture category of database machines. A customized database machine coordinating database functions requested by the host.

- **I/O** Input/Output.

- **logical record processing** Database processing method utilizing logical, as opposed to physical addresses to locate data.

- **LPT (Logic-Per-Track) devices** Specially designed fixed-head disks having search logic associated with each read/write head.

- **MIPS (Millions of Instructions Per Second)** Measure of the processing capacity of a computing device generally used in reference to mainframes.

- **MOMD (Multiple Operation Stream-Multiple Data Stream)** Qadah's category of processing multiplicity which describes machines capable of processing more than one operation on one or more records, simultaneously.

- **movable-arm disk** Magnetic disk whose read/write heads are mounted on an access arm capable of moving across the tracks of the disk.

- **"now" machine** Hsiao's category of database machines utilizing existing datbase and hardware technology, as opposed to technology that is still being developed.

- **on-the-fly processing** Method of processing data as it passes under the read/write head of an LPT device.

- **physical record processing** Database processing method utilizing physical not logical addresses to locate data.

- **primary key** Unique identifier in a relational database table. No two rows in that table are allowed to have the same value in this field.

- **processing multiplicity** Description of the number of simultaneous operations that a database machine can perform on a given amount of data. This is the third dimension in Qadah's three-dimensional database machine space.

- **quasi-associative search** An approach to database searching characterized by the use of a relation as an indexing level.

- **query processing place** The physical place in a database machine design where selection criteria are applied against the data.

- **record** Row of information in a relational database; sometimes called a "tuple".

- **referential integrity** The concept of maintaining consistency of the data in a database when certain values are deleted from a table but still exist in others.

- **relation** Unit of database data storage, also referred to as a "table", in which data is organized in rows and columns (tuples and fields).

- **relational database** Database approach whereby the data store is perceived by its users as a collection of relations or tables.

- **Sierra** The IBM mainframe line featuring 3090 processors.

- **Software Single-Backend (SSB)** Architectural classification used by Hsiao to describe a database machine which is merely a separate mainframe dedicated solely to database support.

- **Software Multiple-Backend (SMB)** More than one separate mainframe dedicated to database support.

- **SOMD (Single Operation Stream-Multiple Data Stream)** A measure of processing multiplicity for a database machine that can execute only one operation at a time, but on more than one record at a time.

- **SOSD (Single Operation Stream-Single Data Stream)** A measure of processing multiplicity for a database machine that can execute one operation at a time on one record at a time.

- **Summit** IBM's proposed mainframe computer line that will replace the Sierras.

- **System/38** IBM's department-sized database machine.

- **table** A relation in a relational database.

- **text search and retrieval database machine** A memeber of Hsiao's database machine application category. These database machines use pattern matching techniques against stores of textual data.

- **tps (transactions per second)** A measure of the processing power of a database system.

- **transaction** Unit of work performed at the database level.

- **tuple** A record in a relational database.

- **Ynet** The subsystem of the DBC/1012 that acts as an intelligent bus, connecting the IFP and AMP components.

Bibliography

[ANDR 86] Andrews, David and Manteghian, Frederick. "System/38 Grows Up." Computerworld, April 21, 1986, pp. 79-82.

- Discusses the history of the System/38 and examines the reasons why this machine was slow to gain acceptance in the United States.

[BABA 87] Baba, Takanobu; Yao, S. Bing; and Hevner, Alan R. "Design of a Functionally Distributed, Multiprocessor Database Machine Using Data Flow Analysis." IEEE Transactions of Computers C-36 (June 1987): 650-665.

- Proposal for a design methodology based on data flow analysis and defining a cost model for a multiprocessor machine.

[BABB 79] Babb, F. "Implementing a Relational Database By Means of Specialized Hardware." ACM Transactions on Database Systems 4(1) (March, 1979): 1-29.

- The first definitive paper on the Content Addressable File Store (CAFS) database machine design. Sections concentrate on the hardware as wll as on relational operations performed by CAFS.

[BABC 86a] Babcock, Charles, "Report Says IBM to Unveil Smart Disk Controller to Speed DB2." Computerworld, November 10, 1986, pp. 19+

- An account of a report by Thomas J. Bird and William H. Inmon concerning IBM's plans to incorporate an Intelligent Disk Controler into its mainframe scheme. Mention is made of the far-reaching implications of such a move by IBM.

[BABC 86b] Babcock, Charles, "Users Say DB2 Can Shoulder IMS Chores and Then Some." Computerworld, November 24, 1986, pp. 99.

- Contrary to IBM's marketing strategies for DB2 as a "test" database management system, users are finding that DB2 has the capability to meet their needs as well as or better than IMS.

[BABC 87] Babcock, Charles, "Referential Integrity Eludes DB2." Computerworld, February 2, 1987, pp. 23-24.

- A brief article comments on IBM's priorities for DB2, which at the time did not include solving the referential integrity problem.

[BABC 88a] Babcock, Charles, "DB2 Extends Specs." Computerworld, April 18, 1988, p. 2.

- Discusses DB2 Release 2.0's proposed support for referntial integrity, based on information from Gary J. Ferdinand, IBM's DB2 product manager in Santa Teresa, California.

[BABC 88b] Babcock, Charles, "IBM Propels DB2 Into Data Base Top Spot." Computerworld, April 25, 1988, p. 1+.

- Focuses on the recently announced MVS/ESA operating system and the incorporation of support for DB2 into its design.

[BAER 80] Baer, Jean-Loup. Computer Systems Architecture. Maryland: Computer Science Press, Inc., 1980.

- A text on basic computer systems architecture.

[BANE 78] Banerjee, Jayanta; Hsiao, David K.; and Baum, Richard I. "Concepts and Capabilities of a Database Computer." ACM Transactions on Database Systems 3(4) (December 1978): 347-84.

- A highly informative paper written by some of the leading authorities in the field of database machine research. Discusses in great detail some of the problems facing software solutions to database management.

[BANE 79] Banerjee, Jayanta; Hsiao, David K.; and Kannan, Krishnamurthi. "DBC - A Database Computer for Very Large Databases." IEEE Transactions on Computers C-28 (June 1979): 414-429.

- An in depth presentation on the DBC system, this paper was often cryptic in its discussion. The previous publication is a better source for information on the DBC.

[BARR 87] Barry, Theresa. "Teradata Unveils Ethernet Attachment." Datamation, April 1, 1987, pp. 89-90.

- An article about the introduction of the Teradata Communications Processor (COP).

[BAZE 86] Bazeley, Barry. "DB2 and DBC/1012 Data Base Computers in Perspective." Strategies For Managing Information, Proceedings of the LOMA Systems Forum April 6-8, 1986 pp. 45-52.

- Source for the Liberty Mutual benchmark information presented in Chapter 3 comparing DB2 and the DBC/1012.

[BORA 81] Boral, Haran and DeWill, David J. "Processor Allocation Strategies for Multiprocessor Database Machines." ACM Transactions on Database Systems 6(2) (June, 1981): 227-254.

- The value of this article was in its presentation on the design of the DIRECT database machine. Other portions of the work focus on topics that are not entirely relevant to research on database machines in general.

[BOZM 87] Bozman, Jean S. "Ex-IBM Exec Sees '88 Barrage." Computerworld October 5, 1987 pp. 1+.

[BRAU 86a] Braude, M. "IBM Plans to Breathe New Life Into IMS." Software Management Strategies Scenario Stamford: Gartner Group, Inc., 1986.

- An early Gartner Group opinion that IBM would not be phasing out IMS as some feared. Market strategies are presented showing why IBM has to keep its IMS customers satisfied.

[BRAU 86b] Braude, M. Keynote: The Evolving Software Industry Stamford: Gartner Group, Inc., 1986.

- Authoritative account of the top software companies and their strategies for maintaining their market shares in the IBM mainframe world. Briefly discusses Teradata and its approach on support for its customers. Projections are made for the future of Online Transaction Processing demands.

[BRAU 86c] Braude, M.and Levine, Peter. Software Management Strategies Scenario Stamford: Gartner Group, Inc., 1986.

- Similar in content to the above Keynote, this report is geared toward Information System Managers who are concerned with the long term development of industry standards for IBM mainframes and their software support.

[BRAU 86d] Braude, M. "System-Managed Storage: The Solution Architecture." Software Management Strategies Stamford: Gartner Group, Inc., April 9, 1986.

- A very good analysis of IBM's proposed system managed storage strategies.

[BRAU 87a] Braude, M. "Teradata Casts a Wider Net." Software Management Strategies Stamford: Gartner Group, Inc., March 16, 1987.

- Reviews Teradata and the progress made so far. Discusses its planned strategy to enter the connectivity market.

[BRAU 87b] Braude, M. "Britton-Lee Enters the High-End Database Machine Arena." Software Management Strategies Stamford: Gartner Group, Inc., September 24, 1987.

- Discusses the introduction of Britton-Lee's BL8000 Shared Database System.

[BRAU 87c] Braude, M. "Teradata - Profiting From The Database Explosion." Software Management Strategies Stamford: Gartner Group, Inc., December 31, 1987.

- An business analysis of Teradata and a discussion about its future endeavors.

[BRAU 87d] Braude, M. "The DBC/1012 Matures as an SQL/OLTP Engine." Software Management Strategies Stamford: Gartner Group, Inc., December 31, 1987.

- A review of the products announced by Teradata in 1987 as well as those planned for 1988.

[BUI 88] Bui, Hien. "Teradata Service Manager." Applied Data Research Memorandum. April 19, 1988.

- A discussion of the update capabilities of the ADR Teradata Service Manager's current release and future releases.

[CARD 79] Cardenas, Alfonso F. Data Base Management Systems. Boston: Allyn & Bacon, Inc., 1979.

- An early work on database management which discusses some of the reasons why such systems became necessary. Objectives of database technology are outlined.

[COMP 84] "The Genesis of a Database Computer." IEEE Computer, November 1984, pp. 42-56.

- Transcript of a very informative interview with Jack Shemer, founder of Teradata Corporation and Philip M. Neches, chief scientist. Among other issues, the topic of the Ynet interconnection network as an algorithm based on the tournament sort is discussed.

[CONN 87/88] Connolly, James. "A Coming Revolution in Mainframe Storage" Computerworld, December 28, 1987/January 4, 1988, pp. 28-31.

- A forecast of IBM's plans for future products to support its System Managed Storage, among other speculations about new mainframe products for 1988.

[COPE 73] Copeland, G. P.; Lipovski, G. J.; and Su, S. Y. W. "The Architecture of CASSM: A Cellular System for Non-Numeric Processing." First Annual Symposium on Computer Architecture December (1973): 121-128.

- An introduction to the CASSM architecture and its basic concepts.

[COPE 87] Copeland, George; Alexander, William; Boughter, Ellen; and Keller, Tom. "Data Placement in BUBBA." MCC Technical Report Number: ACA-ST-003-88 December, 1987.

- An MCC Non-Confidential report on the topic of strategies for placement of data in storage in the BUBBA database machine, a project at MCC.

[DATAC 88] Applied Data Research. Datacom/DB Data Base Administration Guide, 1988.

- Technical manual discussing Database Administration for Datacom/DB Release 7.5A.

[DATAD 88] Applied Data Research. Datadictionary Administration Guide, 1988.

- Technical manual discussing Datadictionary Administration for Datadictionary Release 2.4A.

[DATAQ 88] Applied Data Research. Dataquery Administration Guide, 1988.
- Technical manual discussing Dataquery Release 4.0.

[DATE 86] Date, C. J. An Introduction to Database Systems. Reading: Addison-Wesley Publishing Company, 1986.
- One of the most widely accepted texts covering the topic of Relational Databases. The author is a business colleague of E. F. Codd, the founder of the relational database theory.

[DBC/ 86] DBC/1012 Data Base Computer Concepts and Facilities. (Release 2.1). Los Angeles: Teradata Corporation, 1986.
- A system oriented introduction to the DBC/1012 Database MA{chine. The architecture of the machine is the main focus of this document; a chapter on User Facilities however, does discuss DBC/SWL and its applications.

[DBC/ 87] DBC/1012 Data Base Computer Concepts and Facilities. (Release 3.0). Los Angeles: Teradata Corporation, 1987.
- An update of the previous manual, detailing new features of Release 3.0 over Release 2.1.

[DEFI 73] Defiore, C. and Berra, P. B. "A Data Management System Utilizing An Associative Memory." AFIPS Proceedings of the National Computer Conference 42(1973): 181-185.
- One of the earliest papers in the field of database machine research, the authors introduce a database management system called Information Systems for Associative Memories (IFAM). This research effectively marked the beginning of the associative memory approach to dedicating hardware to the support of database management.

[DEWI 79] DeWitt, David J. "DIRECT - A Multiprocessor Organization for Supporting Relational Database Management Systems." IEEE Transactions on Computers C-28 (June 1979): 395-406.
- Introduction to the DIRECT system Architecture. Concentrates primarily on the query mechanisms in the design.

[DEWI 86] DeWitt, David J.; Graefe, Goetz; Kumar, Krishna B.; Gerber, Robert H.; Heytens, Michael L.; and Muralikrishna, M. "GAMMA - A High Performance Dataflow Database Machine." Proceedings of the Twelfth International Conference on Very Large Data Bases August, 1986: 228-237.
- A reference that describes the author's previous database machine design, DIRECT and attempts to correct problems in that machine with the new design of GAMMA.

[FERR 78] Ferrari, Domenico. Computer Systems Performance Evaluation, Prentice-Hall, Inc., New Jersey, 1978.
- A good source of information on evaluation techniques for computer systems. Details are given of the various types of indices used in measuring such performance.

[FERT 86] Fertig, Robert T. "DB2 Release 2 Key to Transaction Processing." Management Information Week, February 10, 1986, pp. 40+.
- This author is a leading market analyst on the subject of IBM. The most notable opinion in the article is that Fertig anticipates IBM is putting itself into a position to introduce an Intelligent Database Machine.

[FISH 84] Fishman, Daniel H.; Lai, Ming-Yee; and Wilkinson, W. Kevin. "Overview of the JASMIN Database Machine." Proceedings of the 1984 ACM SIGMOD Conference June, 1984: 234-239.
- A project supported by Bell Communication Research, the JASMIN database machine design is one of the most recent. It is discussed in Chapter 5.

[FUSH 86] Fushimi, Shinya; Kitsuregawa, Masaru; and Tanaka, Hidehiko. "An Overview of the System Software of a Parallel Database Machine GRACE." Proceedings of the Twelfth International Conference on Very Large Data Bases August, 1986: 209-219.

- A paper on the Japanese database machine design GRACE. This design was also discussed in Chapter 5 as one of the most recent.

[GART 87a] Gartner Group, Inc. "IBM's Systems Application Architecture (SAA)." January 16, 1987.

- An analysis of IBM's plans for SAA.

[GART 87b] Gartner Group, Inc. "SAA's Notion of Common Applications." August 27, 1987.

- Further discussions on SAA.

[GART 87c] Gartner Group, Inc. "Systems Application Architecture: A Critical Analysis." October 26, 1987.

- A thorough analysis of SAA.

[HESS 84] Hessinger, Paul E. "DB2 Complementing IMS." Computerworld. April 9, 1984. pp. ID9-ID30.

- A very good in depth review of IBM's initial strategies regarding IMS and DB2.

[HSIA 83] Hsiao, David K., ed. Advanced Database Machine Architecture. New Jersey: Prentice-Hall, Inc., 1983.

- An accepted authority on database machines, Hsiao has compiled representative samples of works on different database machine theories. This is a good introductory source on the subject.

[IBM DB2 86] "IBM Database 2." (GC26-4108-2) International Business Machines, San Jose March 1986.

- Specifications for DB2 Release 2.

[IBM AS/400] "IBM Announces." International Business Machines, U. S. Marketing Services Group, June 21, 1988.

- A special IBM publication introducing the AS/400 series and OS/400 operating system.

[IBM 0067] "IBM 3880 Storage Control Model 13." (GA32-00067-3) International Business Machines, September, 1987.

- Description of the 3880 Storage Control. Contains a good overview of the unit's integration in the 370/XA system architecture.

[IBM 2/18/88] Notes from IBM Announcement Presentation regarding the ESA/370 and MVS/ESA products. International Business Machines, Rochester, New York. February 18, 1988.

[IBM 1348] "MVS/Extended Architecture Overview." (GC28-1348-0) International Business Machines. March, 1984.

[IBM 1661] "IBM 3880 Storage Control Models 1,2,3, and 4." (GA26-1661-8) International Business Machines. January, 1985.

- A description manual of some older 3880 models.

[IBM 1675] "IBM Disk Storage Management Guide." (GA26-1675-0) International Business Machines. December, 1982.

- Good source as background reference on IBM's disk subsystems.

[IBM 186-103] "IBM System/38 Announcement Overivew." (186-103) International Business Machines. June 16, 1986.

- An overview of the six new S/38 models announced by IBM.

[IBM 186-104] "New Models of the System/38 Family." (186-104) International Business Machines. June 16, 1986.

- Gives highlights and price ranges for the new S/38 models.

[IBM 186-107] "IBM 9335 Direct Access Storage Subsystem." (186-187) International Business Machines. June 16, 1986.

- Description of the S/38's new 9335 DASD subsystem models.

[IBM 186-109] "IBM 9332 Direct Access Storage Device." (186-109) International Business Machines. June 16, 1986.

- Description of the S/38's new 9332 DASD units.

[IBM 188-037] "IBM Large System Announcement Overview." (188-037) International Business Machines. February 15, 1988.

- Summary of IBM's announcements on ESA/370 and MVS/ESA.

[IBM 188-038] "IBM 3090 Processor Unit Models 280E and 500E and IBM 3090 Processor Unit Model 300E to 400E Upgrade." (188-038) International Business Machines. February 15, 1988.

- Highlights of the new 3090 E models and upgrades for the 300E and 400E models in support of the new ESA/370 architecture.

[IBM 188-087] "IBM AS/400 System Overview." (188-087) International Business Machines. June 21, 1988.

- Product announcement system overview of the new AS/400 series, the results of the Silverlake project.

[IBM 188-088] "IBM 9404 System Unit." (188-088) International Business Machines. June 21, 1988.

- Highlights of the B10 and B20 AS/400 models.

[IBM 188-089] "IBM 9406 System Unit." (188-089) International Business Machines. June 21, 1988.

- Highlights of the B30, B40, B50 amd B60 AS/400 models.

[IBM 288-059] "Enterprise System Architecture/370 and MVS/System Product Version 3." (288-059) International Business Machines. February 15, 1988.

- Description of the highlights of ESA/370 and MVA/ESA.

[IBM 288-060] "MVS/Data Facility Product (MVS/DFP), Version 3." (288-059) International Business Machines. February 15, 1988.

- Details on the newest release of MVS/DFP that will support the ESA/370 architecture.

[IBM 288-196] "IBM Database 2 (DB2) Version 2." (288-196) International Business Machines. April 19, 1988.

- Description of the highlights of the newly announced release of DB2. Discusses performance and operational enhancements made to the product as well as a discussion of referential integrity.

[IBM 288-289] "IBM AS/400 System Environment of IBM Systems Application Architecture (SAA)." (288-289) International Business Machines. June 21, 1988.

- Overview of the AS/400 system and its role in IBM's SAA.

[IBM 288-293] "IBM Structure Query Language/400." (288-293) International Business Machines. June 21, 1988.

- An overview of SQL/400, the query language available on the new AS/400 machines.

[IBM 3040] IBM 3090 Processor Complex Operations Training Package for Models 150 & 180. Volume 1. (GG24-3040-0) International Business Machines. October, 1986.

- A training manual for two older 3090 models. Provides an introduction to the processor complex concept.

[IBM 3090] "The IBM 3090 - More Than Just Another Processor." International Business Machines. 1987.

- Marketing brochure discussing the 3090 Processor Family and the various attributes of each model.

[IBM 4142] MVS/Extended Architecture Data Facility Product Version 2: General Information. (GC26-4142-4) International Business Machines. May, 1987.

- Introductory manual giving an excellent overview of Release 3.0 of MVS/DFP Version 2.

[IBM 4193] IBM 3380 Direct Access Storage: General Information. (GC26-4193-2) International Business Machines. March, 1986.

- General Overview of the 3380 DASD Subsystem.

[IBM 4491] IBM 3380 Direct Access Storage: Introduction (GC26-4491-0) International Business Machines. September, 1987.

- A better and more detailed manual on the 3380 DASD Subsystem.

[IBM 7085] IBM System/370 Extended Architecture (SA22-7085-1) International Business Machines. January, 1987.

- A manual describing the principles of operations of the 370-XA architecture. Gives a comparison between 370-XA and 370 designs.

[IBM 7120] Channel Characteristics and Configuration Guide (SA22-7120-4) International Business Machines. March, 1987.

- Overview of the Channel subsystem of the 3090 Processor Complex.

[IBM 7121] Functional Characteristics (SA22-7121-1) International Business Machines. December, 1985.

- Description of the components and functions of 3090 Model 200 and 400 Processor Complexes.

[IBM 7728] IBM System/38 (GC21-7728-9) International Business Machines. May, 1986.

- Introduction to the System/38.

[IDEA 88] "IDEAL Version 2 Release 0." Applied Data Research, Inc. March, 1988.

- Release Notice of the new release of the IDEAL fourth generation programming language that will interface with both DB2 and the DBC/1012.

[INSI 85] Inside Gartner Group This Week. Stamford: Garnter Group, Inc. October 25, 1985.

- An early report speculating that IBM has changed its attitude toward its "dual database strategy" and is becoming more positive about DB2 and its capabilities to fill a significane market niche.

[KITS 84] Kitsuregawa, Masaru; Tanaka, Hidehiko; and Moto-oka, Tohru. "Architecture and Performance of Relational Algebra Machine GRACE." Proceedings of the International Conference on Parallel Processing. (1984): 241-250.

- Description of the GRACE database machine design by the developers of the project at the University of Tokyo.

[KITS 86] Kitsuregawa, Masaru; Nakano, Miyuki; Harada, Lilian; and Takagi, Mikio. "Performance Evaluation of Functional Disk System With Aggregation Query." Proceedings of the International Conference on Computer Design. (1986): 206-210.

- Review of the Disk subsystem used in the GRACE database machine design.

[KULL 86] Kull, David. "Database Machines Come Of Age." Computer Decisions. June 17, 1986, pp. 42+.

- A highly informative article on the uses being made of the Teradata database machine. While generally positive about the DBC/1012, the article does detail some problems and shortcomings experienced by the machine's users.

[LEIL 78] Leilich, H. O.; Steige, G.; and Zeidler, H. Ch. "The Search Processor for Data Base Management Systems." Fourth International Conference on Very Large Databases June (1978): 280-287.

- A presentation on the SURE database machine design. Contains considerable detail on the operations of the SURE search module.

[LIN 76] Lin, C. S.; Smith, D. C. P.; and Smith, J. M. "The Design of a Rotating Associative Memory for Relational Database Applications." ACM Transactions on Database Systems 1 (1) (March, 1976): 53-65.

- The introductory paper on the RARES design, this article discusses previous designs and offers solutions to their inherent problems. A detailed discussion is included on RARES orthogonal storage organization.

[LIPO 78] Lipovski, G. J. "Architectural Features of CASSM: A Context Addressed Segment Sequential Memory." Proceedings of the Fifth Annual Symposium on Computer Architecture April (1978): 31-38.

- A complete description of the CASSM architecture and search mechanisms. Includes a discussion on content and context addressing.

[MALL 87] Maller, Vic. "Database Machines: Have They A Future?" Computer Bulletin 3: June, 1987. pp. 10-11.

- An opinion that database machines will survive due to their ability to process more data faster and cheaper than conventional mainframe systems.

[MARG 88a] Margolis, Nell. "IBM Repository Fog Burning Off." Computerworld. April 11, 1988 p. 4.

- An article on IBM's plans to introduce a central data dictionary repository for SAA support.

[MARG 88b] Margolis, Nell. "Datadictionary Awaits ANSI Seal." Computerworld. April 18, 1988 p. 27.

- Discusses the ANSI central data dictionary standard IRDS.

[MART 84] Martorelli, William P. "IBM's Software Drive Derives From Hardware." Information Week. June 11, 1984, p. 36.

- Discussion of IBM as a hardware vendor first, software vendor second.

[MART 86] Martorelli, William P. "IBM's DB2 Leads the Surge in Mainframe Relational DBMS." Information Week, January 13, 1986, pp. 26-28.

- Another attestation to the fact that DB2 far ouperformed the anticipated processing rates set by IBM. Users confirm that DB2 can satisfy their needs as well as IMS in many situations.

[MITC 76] Mitchell, R. W. "Content Addressable File Store." Online Database Technology Conference, April (1976).

- An introduction to the basic concepts used in the CAFS database machine design.

[MORA 87] Moran, Robert. "The Watchword is Renovation." Computer Decisions, January 12, 1987, pp. 30-35.

- Written shortly before IBM announced its new 3090 Model E's, this article speculates on the capacity of thos new releases. Of greater interest are the author's predictions of future MIPS usage and OLTP requirements. This is a good source of information about IBM's future Summit line.

[NECH 84] Neches, Philip M. "Hardware Support For Advanced Data Management Systems." Computer, November, 1984. pp. 29-40.

- A very important document detailing the author's search for the best environment for efficient database processing. In this study, Neches determined that a parallel processing environment was best suited. From this, Teradata adopted their architectural approach.

[NECH 85] Neches, Philip M. "The Anatomy of a Data Base Computer System." COMPCOM Proceedings, February 1985. pp. 252-254.

- A technical overview of the first Teradata DBC/1012, detailing its components and architecture.

[NECH 86] Neches, Philip M. "The Anatomy of a Data Base Computer System - Revisited." COMPCOM Proceedings, February 1986. pp. 374-377.

- A revision of the previous article, this focusing on Release 2 of the DBC/1012 which used 802xx processors.

[NEE 88] Nee, Eric. "Teradata Debuts Sun Workstation Interface." Computer Systems News, March 14, 1988.

- Discussion of the new Teradata Sun Interface product.

[OZKA 75] Ozkarahan, E. A.; Schuster, S. A.; and Smith, K. C. "RAP - An Associative Processor for Database Management." National Computer Conference Proceedings, 1975 pp. 379-387.

- The presentation that officially iontroduced the RAP database machine design to the computer science community. This article discusses problems with database management systems run on conventional computers. The basic design of RAP (later referred to in the literature of RAP.1) is set forth.

[OZKA 77] Ozkarahan, E. A.; Schuster, S. A.; and Sevick, K. C. "performance Evaluation of a Telational Associatvie Processor." ACM Transactions on Database Systems, 2 (1977) pp. 175-195.

- This article gives a good introduction to the RAP design and then continues with a detailed discussion of the relational search operations performed by RAP. Comparisons are drawn between performance of operations on RAP and those on conventional systems.

[OZKA 85] Ozkarahan, Esen A. "Evolution and Implementation of the RAP Database Machine." New Generation Computing, 3 (1985): 237-271.

- A thorough article discussing the RAP database machine theory from its inceptions with RAP.1, through RAP.2 and finally RAP.3

[PERR 85] Perry, Robert L. "Relational DBMS Takes Off." Computer Decisions, February 12, 1985 pp. 106+.

- Somewhat dated now, this article gives specific costs for numerous database management systems and database machines. In general, the author presents a very informative overview of database management systems and the role they play in modern computing installations.

[POLI 86] Polilli, Steve. "IBM Opens Closet To Unveil 1st 4GL." Management Information Systems Week, September 15, 1986. pp. 1+.

- Discussion of IBM's Cross System Product which the company revived to serve as a fourth generation language for DB2. The author gives good insight into the shortcomings of CSP.

[QADA 85] Qadah, Ghassan Z. "Database Machines: A Survey." 1985 National Computer Conference Proceedings, March 12-14, 1985, pp. 185-195.

- A cumbersome article that nonetheless is a significant contribution to organizing and categorizing database machine research. The author presents the concept of three-dimensional database machine space. The scheme separates types of database machines by where the processing takes place and how the processing is accomplished.

[RADD 87] Radding, Alan. "The Next 12 Months." Computerworld, November 18, 1987 pp. 55-58.

- A look at the computer industry for 1988, focusing primarily on IBM.

[REIN 87] Reinke, Jane. Chicago Board Options Exchange Benchmarks. February 25, 1987.

- Data collected in CBOE's Utility and Application Benchmarks discussed in Chapter 4.

[ROSS 78] Ross, Ronald G. Data Base Systems, Design, Implementation and Management. New York: AMACOM, 1978.

- Another early work discussing the philosophy of database systems and the organizational need to use and share data.

[SALO 87] Salomon Brother Inc. "Teradata Corporation Common Stock Prospectus." August 11, 1987.

[SCHI 87] Shcnidler, Paul E., Jr. "Mainframes: MVS is Still Dominant." Information Week, January 12, 1987 pp. 30-31.

- A Special Report on operating system trends showing MVS still the most popular among IBM mainframe customers.

[SHU 79] Schuster, Stewart A.; Nguyen, H. B.; Ozkarahan, Esen A.; and Smith, Kenneth C. "RAP.2 - An Associative PRocessor for Databases and Its Applications." IEEE Transactions on Computers, C-28 (June 1979): 446-458.

- A presentation on the second version of the RAP database machine design. This discussion covers the enhancements made in RAP.2 and proposes further changes for the next design.

[SEYB 86] "The 80386: The Promise of the New Technology." The Seybold Outlook on Professional Computing. 5:1 pp. 3-7.
- A very informative article on the architecture of the 803xx family of micoprocessor chips.

[SICI 87] Sicilliam, Michael. "Roulette, IBM-Style." Computerworld. March 16, 1987, pp. 63-69.
- This article summarizes the frustrations experienced by many IBM customers who are trying to maintain upgradable installations, but who are in effect at the mercy of IBM's marketing strategies.

[SLOT 70] Slotnick, D. L. "Logic Per Track Devices." Advances in Computers. 10 (1970): 291-296.
- This brief document spearheaded research in the database machine field.

[SMIT 79] Smith, Diane C. P. and Smith, John Miles. "Relational Database Machines." IEEE Transactions on Computers. 12(3) (March 1979), pp. 28-38.
- An excellent discussion of various aspects of relational database managment theory and the early database machine desings. Very Informative, concise and straightforward account.

[TERA 86] Teradata Corporation. "Performance Analysis of DBC/1012." 1986.
- A public domain report of benchmarking efforts conducted against the DBC/1012 installation at Citibank.

[TERA 87a] Teradata Corporation. "Product Announcement DBC/1012 Communications Processor." PA-87-01 February 9, 1987.

[TERA 87b] Teradata Corporation. "Product Announcement DEC VAX/VMS System Support." PA-87-03 September 29, 1987.

[TERA 88a] Teradata Corporation. "Teradata Believes Future is Open Architecture & Industry Compatibility." MArch 7, 1988.
- Press release on the company's statement of direction regarding future strategies and philosophies.

[TERA 88b] Teradata Corporation. "Teradata Transparency Series Provides SQL Transparency For Major DBMS Products." March 7, 1988.
- Press release announcing the Transparency Series discussed in Chapter 5.

[TERA 88c] Teradata Corporation. "Teradata Interfaces to Sun Workstations." March 7, 1988.
- Press release announcing the Sun Interface discussed in Chapter 5.

[TO 86] To, Cho Fong; Chan, John K.; and Yu, Jocelyn S. "Performance Evaluation of Relational Databases in a Corproate Environment." IEEE Computers (June 1986): 693-697.
- Benchmark information from Bank of America's 4GP strategy discussed in Chapter 4.

[TMS 88] Applied Data Research. "Teradata Service Manager. System Guide." February 1988.
- Technical documentation on the TMS product.

[WAH 80] Wah, Benjamin W. and Yao, S. Bing. "DIALOG - A Distributed Processor Organization for Database Machines." 1980 National Computer Conference Proceedings. pp. 243-253.
- A well written and informative presentation on the desing of the DIRECT database machine architecture.

[WEIN 87] Weiner, Hish. "Trouble in the Sierras." Datamation, January 1, 1987, pp. 69-71.

- From an interesting perspective, this article points out that IBM has not succeeded in installing as many Sierra 3090's as they had expected. Also raised are the concerns among IBM users that this line will be replaced much the same as past IBM machines have been.

[WILK 86] Wilkinson, W. Kevin and Boral Haran. "KEV - A Kernal For BUBBA." MCC Technical Report Number: DB-430-86 December, 1986.

- An MCC Non-Confidential report on the subject of the Kernal on the BUBBA database machine, a project at MCC.

[ZENG 87] Zengerle, Patricia. "Teradata Link Cuts Out Mainframe." Management Information Week. February 16, 1987, p. 8.

- An announcement of the Teradata communications Processor (COP) when it was first introduced.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

Y

3