THE ARCHITECTURE OF A DATABASE COMPUTER - A SUMMARY*

David K. Hsiao and Krishnamurthi Kannan
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

INTRODUCTION

The motivation for seeking hardware solutions to database management functions traditionally carried out by software has been apparent to database designers for sometime now. Firstly, database management software has grown in complexity and size over the years. This growth is prompted by the increase in user requirements, by the formulation of sophisticated models and by the change in data processing mode from an off-line, batched, single user environment to an on-line, concurrent and multi-user environment. Large and complex software systems tend to be failure-prone. Furthermore, practical verification methods for software systems are still not in sight. On the other hand, methods for verifying hardware functionality, design and production have long been available. Advanced technology has also overcome some of the problems of the logic complexity and capacity requirements, making the construction of relatively large and complex computers viable. By incorporating basic database management functions into hardware, not only can we provide more reliable basic functions, but we can also improve the software reliability since the software requirements will be less complex and the system software will be smaller in size.

Secondly, conventional computer systems were not designed for database management. These von Neumann-type computers are good for the preparation and execution of programs for numerical computations and for simple data processing. Database management activity on the other hand is concerned with the storage, retrieval and management of large databases and requires quick search and good update operation for concurrent access. For example, it is well known that the execution of an IMS data management call on an IBM 370 computer takes an average of 130 cpu instructions for preparations of eventual I/O. Thus a typical conventional computer spends much of the time interpreting data management calls instead of executing them. Consequently, the response time for a call is degraded. By relegating the database management functions to specialized hardware, the response time can be improved. Furthermore, the von Neumann-type computer can concentrate on its traditional role of program preparation and execution with freed-up cpu cycles. Both the general-purpose conventional computers and special-purpose database hardware can then yield high performance.

The database computer (DBC) to be described below is a specialized back-end computer. It is capable of managing data of 10^9-10^{10} bytes in size and supporting known data models such as relational, network, and hierarchical models [1]. In addition to its intended purpose of handling large databases and interfacing with various data models, the DBC is one of the first database machines [2] which have built-in protection mechanisms for access control and clustering mechanisms for performance enhancement [3].

THE DBC DATA MODEL

The data model represents a user's view of the data stored in the DBC and the way that the user is permitted to manipulate the data according to that view. There are four aspects associated with the data model of the DBC; the data structure, the query, the clustering and the security. The data structure is the way that the user information is represented in the hardware. The query specifies the way that the data structure can be manipulated. The clustering effects the way the data structure is physically stored and the security controls the way that the data structure is protected from unauthorized use.

The definition of a database starts with two terms: a set AT of "attributes" and a set VA of "values". These terms are left undefined to allow the broadest possible interpretation. A DBC record is a subset of the Cartesian product AT x VA. We will assume that in a record all attributes are distinct. Thus, R is a set of pairs of the form:

(an attribute, a value)

The set of all DBC records which are physically stored in the DBC is called the DBC database. The DBC database may be partitioned into subsets called files. To distinguish among several files, each file is given a unique name called its file name.

The keywords of a DBC record are those attribute-value pairs which characterize the DBC record. Other attribute-value pairs of the record, if any, are collectively called the record body. A DBC record, therefore, consists of a set of keywords and a (possibly empty) string of characters referred to as the record body.

Queries are used in access commands to retrieve and update DBC records. A query is a Boolean

*The work reported herein is supported by the Office of Naval Research, through contract N00014-75-C-0573.

expression of keyword predicates of the form:

<attribute, relational operator, comparative value>

where attribute is an attribute of a keyword, a relational operator is one of the set $\{=,\neq,<,\leq,>,\geq\}$ and comparative value is the value against which the keyword value of the specified attribute is to be tested. The queries will be expressed in disjunctive normal form.

A keyword predicate is true for a DBC record if some keyword in the record satisfies the predicate. A conjunct of predicates is true for a record if each predicate in the conjunct is true for the record. A query is true for a record if one or more conjuncts in the query is true for the record; such a DBC record is said to satisfy the query. The set of all DBC records in a file of the DBC database that satisfy a query will be called its response set or the response data.

The DBC, instead of supporting a fixed record placement scheme for all DBC records, has a record placement mechanism which carries out record placement polices supplied by the DBC users and database administrator. The clustering mechanism allows a DBC user to have some control over the physical placement of a DBC record when it is inserted into the database.

For each DBC file, there are certain keywords which are designated for record placement purposes and are called clustering keywords. The occurrence of a set of these clustering keywords within a record defines a cluster. When a record is to be inserted into the database, the user can specify clustering conditions which enable the DBC to determine the area where insertion is to take place. These conditions are queries consisting of clustering keywords. The power of the clustering mechanism is derived from the fact that it utilizes the same query facility for clustering as is used for retrieving records. Therefore, any set of records which can be retrieved by a single query can also be placed physically together by the DBC, thereby minimizing the number of accesses to the database.

A database access or simply an access is the name of a DBC operation which transfers information to or extracts information from the database. Examples of accesses are retrieve, insert and delete. Let ACC denote the set of all names of the accesses available in the DBC. Let a member of ACC be represented by a and a subset of ACC by A.

A file sanction or simply a sanction is defined as the couple (Q,A) where Q is a query, and A is a subset of ACC. A sanction (Q,A) induces a relation $S.FS_{Q,A}$ over DBC records R of the database such that

$$S.FS_{Q,A}(R) = \begin{cases} A \text{ if } R \text{ satisfies } Q. \\ ACC, \text{ otherwise.} \end{cases}$$

Thus, a sanction induces a security specification which indicates that only the accesses in A may be performed on the records satisfying Q. When R does not satisfy Q, all accesses may be performed on it. In this case we say that no sanctions of (Q,A) are applicable to R. The sanction is a very powerful type of security specification since it allows the full power of the query language (i.e., Q) to be used to specify DBC records to be protected.

Consider a file named F and a set of sanctions S where

$$S = \{(Q_1,A_1), (Q_2,A_2),\ldots,(Q_m,A_m)\}.$$

A database capability (F,S) induces a security specification $S.DC_{F,S}$ over R of F such that

$$S.DC_{F,S}(R) = \bigcap_{i=1}^{m} S.FS_{Q_i,A_i}(R)$$

In words, $S.DC_{F,S}(R)$ is the set of all access granted for R by one or more file sanctions in S and not denied by any sanction of S. Security specifications are therefore stored in the DBC as database capabilities. The database capabilities specify exactly what access operations are allowed on DBC records. The DBC maintains database capabilities for each active user.

THE DBC ORGANIZATION

The organization of the DBC is shown in Figure 1. It consists of two loops of memories and processors, namely, the structure loop and the data loop. The structure loop [4] is composed of four components: the keyword transformation unit (KXU), the structure memory (SM), the structure memory information processor (SMIP) and the index translation unit (IXU). The KXU converts keywords into their internal representations. The need for and the implications of internal representation are discussed in [4]. Data structure maintained and algorithms executed by the KXU are also presented and a hardware organization to realize the KXU is proposed in [4].

The primary function of the SM is to retrieve and update structural information of the database. This information is likely to be large (10^7-10^9 bytes). Furthermore, the operations on this information must be performed at a rate commensurate with that of database operations performed by the components of the data loop. The concept of a partitioned content addressable memory (PCAM) is used to implement the SM with the above properties. Partitions of the SM are realized by employing a set of processing element-memory unit pairs as shown in Figure 2. Powerful PCAM organizations are possible using emerging technologies. To this end, three design alternatives using three different technologies have been examined. The three technologies are magnetic bubble memories, charge-coupled devices (CCDs) and electron beam addressable memories (EBAMs).

The SMIP is responsible for performing set intersections on structural information retrieved by the SM. The concept of PCAMs is once again utilized to perform rapid intersection. The IXU is intended to decode the structural information output by the SMIP.

The four components are designed to operate concurrently. Keywords are sent to the KXU at regular intervals by the DBC's command and control processor (DBCCP). The output of the KXU is sent to the SM which retrieves index terms for the transformed keyword predicates and sends them to the SMIP. The SMIP output is interpreted by the IXU and sent to the DBCCP. This pipeline of processors results in maximum utilization of the hardware.

The design of the data loop is presented in [5]. The three components that form the data loop are the database command and control processor (DBCCP), the mass memory (MM) and the security filter processor (SFP). When a query in a command is sent to the DBCCP, the DBCCP decodes the query using the structure loop. The DBCCP then uses the

structure information returned from the structure loop to form localized mass memory commands. These commands are then sent to the mass memory (MM) (See Figure 3). The design of the MM is based on the PCAM concept. In the MM, a partition of the PCAM is a cylinder of a moving-head disk unit. The cylinder is made content-addressable by incorporating track information processors (TIPs) (one for each track of a cylinder) for concurrent processing of the tracks of a cylinder. Furthermore, the disk read/write mechanism is modified to allow parallel read/write of all the tracks of a cylinder. The choice of a processor-oriented implementation using TIPs vs. a memory-oriented implementation using a large cylinder memory is argued. Management of MM orders and their execution by the TIPs are discussed. Garbage collection and space reclamations are also discussed in considerable details in terms of compaction and update operations of the MM. By far the most powerful operation of the MM is the search and retrieve operation. The MM is capable of searching for and retrieving records which satisfy queries made up of keyword predicates. Because the records in the MM are addressable by content and carry no address pointers, they need no updating as long as the records exist in the database. This is true even if the security specifications of the database change frequently. The organization of the MM is depicted in Figure 4.

The security filter processor provides the type B security enforcement and sorting. The type B security enforcement mechanism is provided for those users who do not take advantage of the type A security mechanism based on the concept of security atoms. A security atom is a collection of records all which have the same security requirements. It is possible to identify records belonging to a security atom by the presence of certain keywords known as security keywords. The type A security mechanism incurs less security overhead. However, it needs the user's cooperation. First, the user must understand the security atom concept; then, the user must convey the security requirements in terms of file sanctions on his data records. On the other hand, the type B security mechanism does not require such user cooperation. Nevertheless, posterior checking of response data against full file sanctions is an expensive undertaking. The sort mechanism enables the response data to be ordered by values of certain attributes. This is usually the way that the user application programs would like to receive the records in the front-end computer system.

REFERENCES

[1] Hsiao, D. K., Kerr, D. S., and Ng, F. K., "DBC Software Requirements For Supporting Hierarchical Databases," The Ohio State University
Tech. Tep. No. OSU-CISRC-TR-77-1, (April, 1977).

[2] Baum, R. I. and Hsiao, D. K., "Database Computers - A Step Towards Data Utilities" IEEE Transactions on Computers, C25, 12, (Dec. 1976), pp. 1254-1259.

[3] Baum, R. I., Hsiao, D. K., and Kannan, K., "The Architecture of a Database Computer - Part I: Concepts and Capabilities," The Ohio State University
Tech. Rep. No. OSU-CISRC-TR-76-1, (Sept, 1976).

[4] Hsiao, D. K. and Kannan, K., "The Architecture of a Database Computer - Part II: The Design of Structure Memory and its Related Processors", The Ohio State University
Tech. Rep. No. OSU-CISRC-TR-76-2, (October,1976).

[5] Hsiao, D. K. and Kannan, K., "The Architecture of a Database Computer - Part III: The Design of the Mass Memory and its Related Components", The Ohio State University
Tech. Rep. OSU-CISRC-TR-76-3, (December, 1976).

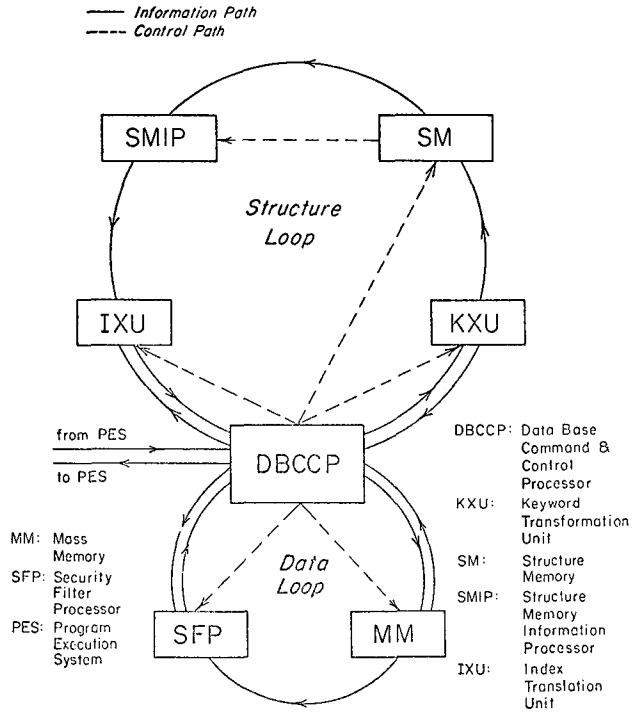

Figure 1. Architecture of DBC.

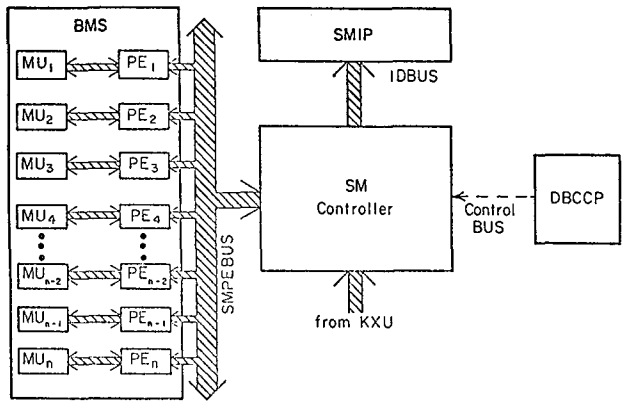

Figure 2. Organization of SM.

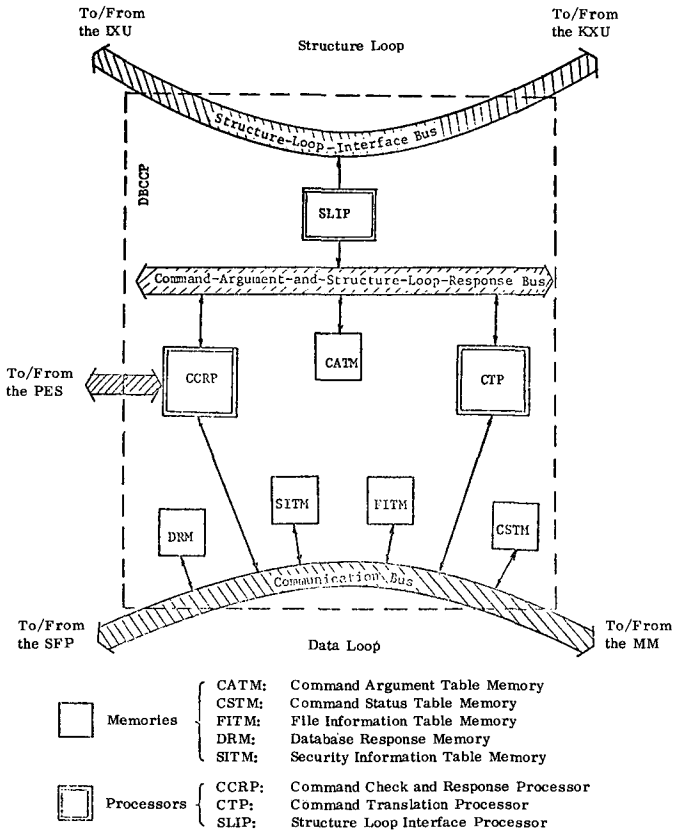

Figure 3. Physical Organization of DBCCP.

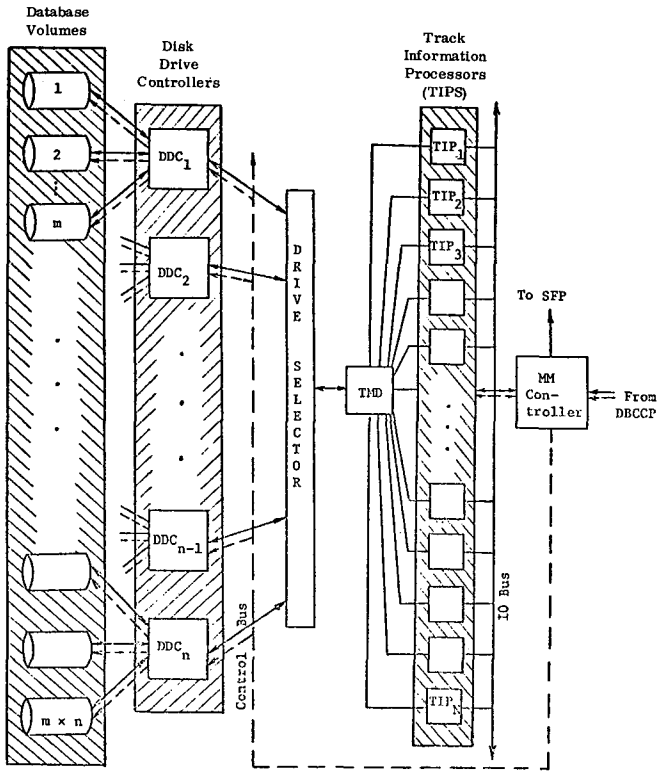

Figure 4. Organization of the Mass Memory (MM).