TM-0002

A Relational Database Machine "Delta"

by

Shigeki Shibayama, Takeo Kakuta,
Nobuyoshi Miyazaki, Haruo Yokota
and Kunio Murakami

November, 1982

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191－5
Telex ICOT J32964

Institute for New Generation Computer Technology

A Relational Database Machine "Delta"

(Draft)

Shigeki Shibayama, Takeo Kakuta, Nobuyoshi Miyazaki,

Haruo Yokota, Kunio Murakami

Institute for New Generation Computer Technology

(ICOT)

0. BACKGROUND

Japan's Fifth Generation Computer System (FGCS) project is scheduled to be a ten-year-long activity. Knowledge Base Machine is one of its expected achievements in that research period. When completed, KBM (Knowledge Base Machine) will be seen as an intelligent Knowledge store, which co-operates interactively with the inference-based new architecture computers, to provide users with natural and intelligent interfaces to the consolidated system.

In our first-stage three-year project, the primary object of KBM (Knowledge Base Machine) Group is to provide a working Relational Database Machine which serves multiple of Sequential Inference Machine (SIM) users via a local area network (LAN). Investigation for the methods to amalgamate database management system (DBMS) and logic programming language, represented now by Prolog, is also an important subject which we are now greatly interested.

1. INTRODUCTION

In order to naturally manipulate knowledge base information with logic programming languages to form a knowledge base system, it is considered necessary to combine logic programming languages with data management systems.

Relational model was chosen as our working database model for database machine because of its affinity to logic programming languages such as Prolog and ICOT'S FGKL(Fifth Generation Kernel Language). In the experimental knowledge base systems, the databases, or knowledge bases, are not so large that all the program including the database could be stored in main memory. So long as applications remain this small, there would be no needs to positively introduce a database management system to knowledge base systems. Yet this is not likely, because in every computer fields, as the new concepts come into reality and become widely used by a lot of people, the size of applications has had the ever-growing tendency that requirements for faster and larger computer have never been ceased. As number of applications are going to be implemented in Artificial Intelligence fields, there will arise needs for a consolidated data management system which can be shared and provide a non-procedural utilities for data handling. This was the history in the fields of business system computer applications and will remain true similarly in the next generation computer era. So researches to combine database systems and new concept languages as, Prolog or FGKL, are considered important for future computer applications. To begin with, it is considered useful to combine relational database and logic programming languages because both of them adopts predicate calculus as their basis. Some researches are being carried out to naturally combine logic programming language and relational database. We think that research towards Knowledge Base Mechanism and, eventually, Knowledge Base Machine implementation is in the right direction with the logic programming language and relational DBM's as starting points. The aim of this brief paper is to describe the concepts and to give an overview of the ICOT's Database Machine named "Delta".

2. ARCHITECTURE OVERVIEW

As described in the first chapter, ICOT's RDBM "Delta" is projected to be completed to actually serve the researchers in three years, so not only considerations for the novel RDBM architecture which is extensible and compatible with future VLSI technologies but also realizability utilizing the current technologies is important to determine the architecture of the RDBM. Though the incorporation of future extensibility and parallelism applicable to VLSI's should be considered to define a new architecture, number of trade-offs to implement it in this short period of time be made to fulfill first objective. If some part of this architecture seemed to be rather conservative, this could be the effect of the above realistic standpoints. Yet we believe that Delta could be the first life-size working RDBM adopting the proposed effective RDB manipulating mechanisms.

2.1 RDBM CONCEPTS

The basic RDBM concepts are :

i) High-level interface based on Relational Algebra applicable to logic programming class of languages
ii) Efficient query processing capability by the use of dedicated relational DB operation engine hardware
iii) large capacity Hierarchical-structured Memory.

The RDBM provides a high-level command or instruction set based on relational algebra in order to serve SIM's at a higher and non-procedural concept, therewith it reduces SIM software loads and translation overheads. The RDBM engine performs commands close to the interface commands of whole RDBM. This high-level architecture is also useful to reduce software loads in CP as is RDBM to SIM's. Command execution mechanism is based on sorting and merging the attributes of relations transferred on-the-fly from the Hierarchical Memory Subsystem. RDB engine will have no particular indexing mechanism. It scans the set of whole transferred attributes and produces results as the attributes flows through the RDB engine. Both the Control Processor and Hierarchical Memory are responsible for reducing the total number of tuples to be scanned by RDBM engine using a kind of indexing and clustering technique. Hierarchical Memory provides the means of storing bulky database in the RDBM and actually stores relations and directory/dictionary information. As the name indicates, the subsystem is hierarchically structured, to keep good average access time with low access costs utilizing the sophisticated staging/destaging mechanism among each hierarchy.

Several functionally independent subsystems are necessary to compose an RDBM. As shown in fig.1, the RDBM is composed of three main subsystems:

i) Control Processor (CP)

ii) RDB Engine (RDBE)

iii) Hierarchical Memory (HM).

Other subsystems provided to form a working RDBM are:

vi) Interface Processor, a front-end processor which interfaces Delta to LAN environment and

v) Maintenance Processor, a supervisory processor which monitors other subsystems.

A brief descriptions of the listed subsystems are given in the following chapters.

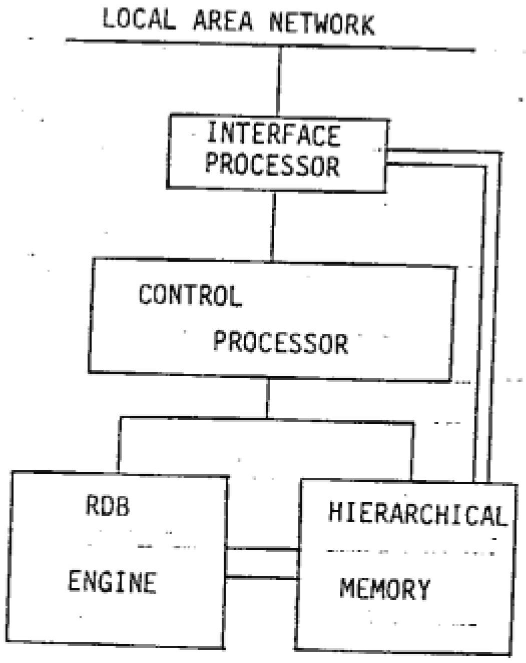

FIG.1 Delta Block Diagram

2.2 CONTROL PROCESSOR

Control Processor's main role is to generally control the whole RDBM. CP receives a command-tree composed of series of RDBM commands structured to a logical tree which represents a DB transaction, then translates it to a set of internal subcommands to activate the hardware resources, namely RDB engine and HM, which actually performs query processing. CP's another important role is to provide a multi-user environment for interactive use via LAN. This transaction concurrency is, we think, one of the most complex functions among levels of software in CP to implement a working Database Machine.

CP's software is divided into two levels of hierarchical structure as listed below.

i) DB Management level

ii) Unit-Command Process level

The upper DB Management level is similar to an existent DBMS in that multi-users' service along with DBMS manipulating utilities should be provided. The difference from current DBMS is the interface to the lower Unit-Command Process level of software, which, if implemented in conventional von Neumann type computer, usually accommodates complex access method handling where various navigational strategies are employed. In this RDBM, macroscopic navigation method is implemented in the CP software whereas microscopic tuple-level navigation is performed in hardware attribute scanning mechanism thereby eliminating the software costs and complexity. CP's Unit-Command Process software controls hardware resources to complete a unit DB operation command in the command set which is determined in conformity to Relational Algebra. The implementation of CP can be a dedicated architecture DBM Controlling Processor, which will be efficient in handling queries and controlling RDBM, yet it will need a certain amount of time and effort to design and build from scratch. So it seems practical and natural to use a conventional mini-computer for our purposes.

2.3 RDB Engine

RDB Engine (RDBE) is a piece of dedicated hardware for executing RDB commands at the lowest level. RDBE has its own command set associated with the RDB command set. As RDB command set has followed Relational Algebra and RDBE should efficiently execute that command set at the data manipulation level, RDBE command has been determined to naturally support the higher level RDBM command set. Furthermore, RDBE command set not only matches higher-level one, but also has efficient processing algorithm done at the hardware level, which will be discussed later.

RDBE is tightly connected to Hierarchical Memory (HM) in the sense that data stream path is provided between them to flow data stream in both directions. When a query is translated in CP and RDBE receives a command from CP, it waits for arrival of data stream from HM, which also received a command specifying a particular data streaming into RDBE. Upon arrival of first data from HM, RDBE operates on the data stream on-the-fly to complete most of the command overlapped with data transfer. The basic RDBE command processing mechanism is preprocess sorting and command executing merging. Input data stream is first sorted on-the-fly using pipeline merge-sorting algorithm. Sorted stream is then sent to a Relational Operation Unit (ROU) where Relational Algebra-like command is executed on sorted data stream. Basic processing scheme in ROU is merging on comparing input stream against another data stream stored in backtraceable first-in-first-out (FIFO) and then controlling the result output. Heavy relational algebra operations such as join and projection can be efficiently processed if the relation or attribute is sorted by its values. Restriction requires linear order of processing time to the cardinality of a relation, still sorting reduces the constant coefficient of processing time and provides a way to execute multiple-conditioned restriction in one scan. Although on-the-fly scanning of relation is simple and, if properly hardware-supported, fast in many cases, it would be too time-consuming to scan a whole relation every time. To eliminate the number of tuples in a relation to be scanned by RDB Engine, two-level clustering technique is employed in the RDBM system. This clustering technique is described in a later chapter.

2.4 Hierarchical Memory (HM)

Hierarchical Memory subsystem is the actual database store. To provide fast access time and large storage space is always the ultimate goal of every large capacity storage system. To achieve this goal, it is widely accepted that to construct a memory system with layers of different-natured memory devices, incorporating mechanisms to stage and destage bunch of data. This is a most general approach to give a memory system whose access time is closer to the fastest but expensive devices (usually semiconductor RAM's) and cost is to the cheapest devices. Cache is now considered to be the most popular and viable technique to efficiently realize the aforesaid goal, if proper replacement algorithm is applied in the cache control part. However, to manipulate a large capacity memory system, cache alone would not be able to fully support a really efficient processing, because there does exist a class of data which is always needed in the processing. This kind of data requires a more predefined treatment in the context of global RDBM control. Another reason why cache alone could not optimally manage a memory system is that cache sometimes degrades system performance by replacing a part of logically associated data cluster, one such example is dictionary/directory information. In order to eliminate this kind of non-intelligent behavior associated with simple cache, some kind of explicit upper-level control mechanism should be provided in HM. That is to say, it seems helpful to assign buffer space in HM, where cache replacement algorithm can not affect. Of course, there should be an explicit control mechanism to handle this, which could be a cause of overhead unless appropriately reconciled.

The lowermost storage device will be moving-head magnetic disk, which are still considered to be the most appropriate device for storing a good amount of data with moderate cost. In actual database query manipulation, the access speed of moving-head disk is apparently too unsatisfactory if every tempprary result will go to disk and needs time-consuming disk activation every time the temporary result is accessed. Large semiconductor disk comprised of random access memory is considered to fill this gap. This semiconductor disk works as both cache and buffer, in the sense that transparent data management is carried out in cache part and programmable control can be accomodated in buffer part, thereby optimizes system performance. The control mechanism for cache and buffer is self-contained in HM. Other subsystems have only to specify HM minimum amount of information or parameter to utilize this facility.

Another important HM function is tuple reconstruction. As is explained in a later chapter, Delta stores relations attribute- or domain-based, so tuple reconstruction, that is to say, to make up a tuple by gathering the corresponding attribute values by their tuple-identifier as a key, is needed in the later phase of query processing. This problem is solved by the two-level clustering, which is used to reduce the number of attributes to be reconstructed. The actual reconstruction is carried out in HM such that attributes to be reconstructed are stuck vertically in memory, sorted by their tuple-id, and then read out horizontally to make a tuple-wise concatenation. Unless this function is provided in HM, the tuple reconstruction cost would be too high to tolerate.

2.5 INTERFACE PROCESSOR

The Interface Processor is responsible for interfacing LAN and RDBM. It has facilities for handling levels of network protocols, which make it easy for RDBM to communicate the other nodes (SIM's) connected to LAN. To do this, IP has a certain amount of buffer memory to interact with RDBM and LAN, where packing and unpacking of packets should be carried out. It also has a port to HM, this port acts as the main data path to network connected SIM's. Once result relation is calculated and stored in HM, CP delegates the IP to transfer it to the SIM via LAN which made the query and requiring the result.

2.6 MAINTENANCE PROCESSOR

Maintenance Processor (MP) plays a similar role in RDBM as the supervisory processors provided in today's large computer systems. It monitors other subsystems and maintains them. If necessary, local test procedure is invoked by MP and it diagnoses a concerned subsystem. There is another important role which should be done by MP. That is a set of data collection functions. Since Delta is an experimental vehicle to investigate various characteristics in actual operation, statistical data such as resource contention, cache usage, query processing rate and so on are needed for future improvement. Unless data collection function is considered in the first design stage, sometimes it would be very hard to collect necessary information even if expensive hardware monitor equipment is applied at a later time.

3. INTERNAL SCHEMA

To design a database machine, it is of great importance to define an internal schema or how relations are stored in memory, because it greatly influences system performance and flexibility. There are two principal class of internal schemas for storing relations for database machines. Namely one is tuple-based internal schema and the other is attribute- or domain-based schema. B-tree type schema is useful in implementing relational databases on conventional computers. We think, however, that Relational Algebra is naturally mapped on an internal schema closer to its conceptual schema and that manipulating pointers and tuples mingled together within storage is not suited for our hardware algorithm. It would be possible to construct a database machine which simulates by hardware the handling of B-trees in conventional computers. This architecture, however, would be strictly associated with a specific implementation technique and therefore have little room for flexible and extensible improvements.

Among the two candidates for Delta's internal schema, we have chosen attribute-based one. Tuple-based schema is appropriate for those machines which have general class of processor and buffer by which relations are processed. In this case, on-the-fly processing is difficult, because the relation to be processed has first to be staged up to the buffer and then processed by the general-purpose processor by its programs and finally destaged back to relation store. Furthermore, even if clustering is applied, tuples with unnecessary attributes have to be read out, which could result in the undesirable increase in processing time.

For these reasons, we have chosen the attribute-based schema which is assumed to be suited for our RDB Engine hardware in which an attribute of relation is processed on-the-fly to give fast RDB query manipulation. The problem associated with this schema is tuple reconstruction, which has been briefly described in HM chapter.

To overcome this problem, we adopted two-level clustering to efficiently execute tuple reconstruction. The idea is that in first-level, the relation is primarily clustered by the attribute values, then the primary clusters are divided secondarily by the tuple-identifier values to form a secondary clusters which correspond to a storage unit.

By doing this, only small amount of storage is required to be scanned to reconstruct a tuple. And usually all the attributes are not needed to form output tuples in a query, unnecessary attributes don't have to be even staged. There could be other clustering techniques to optimally reduce I/O counts to secondary memory, we think that by adopting the two-level clustering with large semiconductor disk in HM subsystem, sufficient process speed can be achieved.

4. CONCLUSION

In previous chapters, brief descriptions of the ICOT's database machine Delta are presented. Machine implementation in operational form is an primary target with considerations to combine RDB and logic programming language. We have addressed the importance of efficiency, which will be achieved by the use of hardware support of HM and RDBE controlled by CP. In this class of RDBM, attribute-based internal schema is considered more efficient than tuple-based one. Yet the verification and evaluation of these ideas should be pursued in the course of our research, with the investigation of implementation details.