ICOT Technical Report: TR-063

TR-063

Design and Implementation of the Relational Database Engine

by

Hiroshi Sakai, Kazuhide Iwata, Shigeo Kamiya

Masa-aki Abe, Akio Tanaka

(Toshiba Corporation)

Shigeki Shibayama, Kunio Murakami

April, 1984

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

Design and Implementaion of the Relational Database Engine

Hiroshi Sakai Kazuhide Iwata Shigeo Kamiya
Masa-aki Abe Akio Tanaka

RESEARCH and DEVELOPMENT CENTER
TOSHIBA CORPORATION

Shigeki Shibayama Kunio Murakami

ICOT RESEARCH CENTER

ABSTRACT

The Relational Database Engine (RDBE for short) is a dedicated component within a backend relational database machine Delta which is under development at ICOT Research Center.

The RDBE consists of a general-purpose processor and a specialized processor, engine core. The former controls the latter and performs operations which are not supported by the latter. The latter performs a wide range of the relational database operations by pipeline processing synchronized with the data transfer rate of 3MB/sec. The engine core is characterized by its efficiency based on our relational database processing algorithm. In particular, the processing of null values and duplicate keys is performed efficiently without disturbing pipeline processing.

Besides the efficiency, the RDBE, which is designed for practical database processing, has the following advantages:

(1) It provides logical operations such as join, sort and aggregate functions.

(2) It manipulates sufficient data types including null values.

(3) It has check mechanisms to improve reliability.

They are realized by the cooperation of the hardware and software.

This paper desicribes an overview of the Delta system, the design considerations and implementaion of the RDBE, the engine core architecture, and roles of the software. A preliminary performance estimation of the RDBE is also presented.

1. Introduction

The Relational Database Engine (RDBE) is a dedicated component used within a backend relational database machine Delta which is under development at ICOT Research Center (Institute for New Generation Computer Technology)[1].

The relational data model is thought to be useful not only for conventional database systems, but also for logic database systems[2][3]. However, relational database operations are time-consuming and pose a heavy burden on conventional computer systems. Since the 1970's, various kinds of database machines have been proposed to improve the response time and the system throughput of relational database systems. However, very few systems have been implemented and very few implementation experiences have been presented till now[4][5]. Under these circumstances, it is very useful to present our design and implementation of the RDBE which performs relational algebra operations such as join and set operations such as intersection efficiently.

The RDBE is composed of a general-purpose processor and a specialized processor named an engine core. The former controls the latter and performs the operations which are not supported by the latter. On the other hand, the latter performs a wide range of database operations by pipeline processing synchronized with the data transfer rate of 3MB/sec. The pipeline processing is carried out by overlapping data processing and data transfer. In currently proposed pipeline hardware designs such as a sorter, the treatment of null values and duplicate keys is not explicitly considered. Therefore, it is necessary to solve such

problems in the application of these hardware designs to practical database processing. Our hardware can perform operations on practically sufficient kinds of data types including null values and duplicate keys without disturbing the pipeline processing.

In Section 2, we present an overview of the Delta architecture and summarize the main functions of the Delta components. Section 3 and 4 describe our design and implementation of the RDBE and the engine core architecture, respectively. Section 5 describes the role of the software in the RDBE. Section 6 estimates the performance of RDBE. Finally, our conclusions and future plans are presented in Section 7.

2. Overview of the Delta architecture

The Delta global architecture is shown in Fig. 1. In this figure, the dotted line shows our future plan. Delta consists of the following components.

(1) An interface processor(IP) which interfaces Delta to a local area network(LAN).

(2) A control processor(CP) which provides the database management functions such as concurrency control and database recovery.

(3) A relational database engine(RDBE) which is the key component for the processing of relational database operations in Delta. The RDBE is implemented by using a specialized processor to be described later in detail.

(4) A maintenance processor (MP) which provides functions for maintaining reliability and serviceability of Delta.

(5) A hierarchical memory(HM) which provides the functions for storing, accessing, clustering and maintaining relations. The HM is implemented by using a general-purpose computer as a controller, a large amount of semiconductor memory and large moving head disks. The HM is connected to other components by high speed channels.

The general processing sequence of commands in Delta is as follows.

The IP receives relational algebra level commands called Delta commands from a host connected to the LAN and sends them to CP. The CP translates these commands to a sequence of internal subcommands for RDBE and HM. After the execution of the Delta commands is completed, the IP transfers the result from HM to the host via LAN.

3. Design and implementation of the RDBE

3.1 Configuration

Fig.2 shows the RDBE configuration which is designed in consideration of relational database operations described later in detail. The RDBE is implemented by using the following modules.

(1) A 16 bit CPU, with 512K bytes of main memory which is used as the controller of the RDBE.

(2) A key processor named an engine core which consists of the IN module, sort module(sorter) and merge module(merger).

(3) Two input/output controllers(IOC) which control the internal bus connection in Delta.

(4) Two HM adapters which provide the interface function between RDBE and HM.

In Fig.2, DT,PT,NL and DP stand for data lines(16 bit), parity lines(2 bit), a null line(1 bit) and a duplication line(1 bit) respectively. The null line is used to denote that a tuple with a null value is on the data lines. The duplication line is used to denote that a tuple which has the same key value as the next one is on the data lines.

These modules are controlled to run simultaneously so that pipeline processing synchronized with the data transfer rate 3MB/sec may be achieved.

The main data path is from the HM adapter(IN) to the HM adapter(OUT) through the engine core. In case the RDBE operation needs two relations, one is stored into the engine core from the HM adapter(IN) first. Then the other comes from the HM

adapter(IN) and the engine core compares the tuples of those two relations and generates the result, which is sent to HM via the HM adapter(OUT).

In case the CPU itself is required to manipulate the data, the result from the engine core is stored into the main memory via the HM adapter(OUT), and after the CPU has finished the manipulation, the final result is sent to HM via the HM adapter(OUT).

3.2 Operations

RDBE offers various operations necessary for relational database processing. They are classified into the following categories.

(a) Relational operations such as join, projection, and selection.

(b) Sort operation both in ascending and descending order.

(c) Set operations such as Intersection, Union, and Difference.

(d) Arithmetic operations.

(e) Aggregate operations over an entire relation and also over nonintersecting partitions of a relation.

(f) Miscellaneous operations peculiar to the way of the database management in Delta.

The RDBE operations are listed in Fig.3.

One of the advantages of the RDBE is that projection operation is performed with other operations at the same time, which is explained in section 4.

Some of the operations are described below.

(1) Pass

The Pass operation requires a single relation and generates the result by passing the input tuples through the engine core. It is useful for projection, arithmetic operations, and selection with complicated conditions. The former is performed by the engine core, and the others are done by its CPU.

(2) Sort

The Sort operation arranges the sequence of tuples of a relation according to the value of the key field. The CP does not have to care how the operation is performed. When the size of the relation is not greater than the capacity of the engine core, the RDBE can generate the result directly by the engine core. Otherwise, the RDBE uses some work buffers and generates the final result by repeating the two-way merge operations.

Most of the operations are efficiently performed by the engine core, and the others are performed by the CPU as well as the engine core.

3.3 Structure of a tuple

The internal representation of a tuple is as follows. The RDBE is designed to treat it efficiently.

Each tuple in a relation has the same length less than 4Kbytes and the same number of fields, and also the corresponding field over a relation has the same data type including its length.

A field usually has an extra area named tag, which indicates whether the value is null or not. The data types are as follows.

(1) Unsigned Integer of 2, 4, ... ,4094 byte long.

(2) Signed Integer of 2, 4, ... ,4094 byte long.

(3) Single precision floating point number.

3.4 Trade-off between hardware and software

It is necessary to mix hardware and software implementations optimumly, in order to achieve a high cost/performance database processing. We analyzed the basic relational operations from the points of operation frequency and operation complexity. The analysis revealed that the operations based on sorting were time-consuming and the operations on arithmetic were not used frequently. So we decided the former type of operations could be realized by hardware devices and the others by the cooperation of hardware and software. Adopting the design, the RDBE has the advantage that it can perform some combined operations by passing the data through the engine core first and then by CPU processing.

In the next section, we will discuss the hardware architecture of the engine core.

4. Engine core architecture

The engine core is a key processor, which is composed of the IN module, sorter and merger, as shown in Fig.2. It performs relational database operations on an input data stream from the HM adapter(IN) and sends the results to the HM adapter(OUT). Each module of the engine core is designed to provide the following functions so as to achieve pipeline processing.

4.1 IN module

The main function of this module is to transform the input data format to an internal one suitable for the trailing modules of the engine core. These transformations are as follows.

(1) Field ordering

As the engine core is designed to process the whole tuple (not only the key field) in order to perform set operations as well as relational algebra operations, a field ordering is carried out so that the key field is positioned at the head of the tuple. It is possible by this function for the engine core to perform pipeline processing from head to tail of the tuple.

(2) Format arrangement

Data stored in a compact format in the HM is converted to a standard format suitable for the trailing modules of the engine core.

(3) Transformation in the key field

As an engine core can compare only absolute values, signed integer and floating point numbers are converted so that the comparison mechanism can work well.

(4) Generation of null value bit signal

In Delta, the null value is indicated in the tag field. The null bit of the tag field is put on a special line (NL) so that the trailing modules do not have to check the tag field.

4.2 Sorter

Many sorting algorithms have been proposed and studied[6]. Various kinds of sorters have been investigated[7]. But applications of them to practical database machines are very few.

Our sorter adopts the well-known two-way merge sort algorithm[8]. Its configuration is shown in Fig.4. It consists of 12 processing elements called the sorting cell and one processing element called the sorting checker. Each element contains two memories each with a first-in/first-out function (FIFO), a comparator and a control circuit.

In general, the i-th sorting cell creates a sorted sequence of 2^i tuples (one stream) by merging two output sorted sequences of 2^{i-1} tuples from the (i-1)th cell. Each cell runs synchronously. The i-th cell starts the merge operation when it has received one complete string and the first 2 bytes of a second string from the (i-1)th cell. As the last cell has two 32Kbyte memories, our sorter creates 4096 sorted tuples when the tuple length is not greater than 16 bytes.

Each cell has two operation modes called the sort mode and pass mode. The former merges two sorted strings of the previous cell into one sorted string and transfers it to the next cell. On the other hand, the latter does not merge but transfers input data to the next cell directly. This mode is used when the tuple length is greater than 16 bytes and it is not necessary to activate all the cells because of a small amount of data.

The sorting checker connected to the last sorting cell is an unique element which verifies the results and marks duplicate

tuples by turning the duplication line on. The role of the sorting checker is to gain reliability of the engine core and to provide a tuple duplication signal to the merger in order to achieve hardware implementation of our relational database processing algorithm.

The operations of each cell are divided into three cycles. They are the 2 byte read cycle of one string, 2 byte read cycle of the other string and compare-transfer cycle. Each cycle takes 220ns, and the 2 byte merge operation takes 660 ns.

Besides the characteristics mentioned above, our sorter is characterized by its capability of processing equal keys and null values. In the former case, it keeps the original order of the input tuple occurrence (stable sorting) and in the latter case it outputs the sorted sequence of tuples with normal values followed by the tuples with null values.

4.3 Merger

The merger is the central module of the engine core which performs relational algebra operations and other operations by using a processing algorithm based on the merge-sort operation. They are called merger commands and are classified into five types of operations. They are listed in Fig.5. They are characterized by their capability of processing null values and duplicate tuples without disturbing the pipeline operation.

The block diagram of the merger is shown in Fig.6. It consists of an operation part and an output control part.

The operation part contains two 64Kbyte memories (U-memory and L-memory) each with a FIFO function, a comparator, a control ROM

table and other circuits. The function of this part involves the following steps:

(1) Storing two sorted streams from the sorter into the memories.

(2) Reading each tuple from two memories simultaneously and providing them to the comparator and the tuple memory in the next part.

(3) Comparing the keys of each tuple and detecting output tuples satisfying the condition of the command.

These functions are executed under the control of a ROM table of 1Kwords *10bit. The address of the ROM table consists of a null flag, duplication flag, comparison result and so on. On the other hand, the output of the ROM table consists of memory address control signals, tuple selection signals used for the next part, an operation end signal and so on.

The output control part consists of two 16Kbyte tuple memories, two field ordering circuits, two field selection circuits, two data type transformation circuits, a new TID generator, a selector and an output sequence controller. The function of this part involves the following steps which are executed, if necessary, according to the assignment of the software:

(1) Reordering the fields of an output tuple.

(2) Selecting fields of an output tuple.

(3) Undoing the transformation of the key field

(4) Adding a new TID to an output tuple.

The examples of these functions are explained in Fig.7.

Fig.7(a) shows the function of reordering the field of an output tuple. That is to say, a tuple(1) with 5 fields (A, B, C, D, E, B is a key field) is rotated to tuple(2) by the IN module so that the key field is positioned at the head of the tuple, and the tuple(2) is rearranged to the original tuple(3) by the merger.

Fig.7(b) shows the function of selecting fields of an output tuple. That is to say, the tuple(4) is projected to tuple(5) or tuple(6) by the appointment of the two pointers(P_1,P_2). Fig.7(c) shows the function of adding a new TID(NTID) to an output tuple.

Fig.8 shows a processing example of the JOIN-EQ command which is of great importance in the relational database system. Fig.8(a) describes two input sorted streams (S1 and S2) according to the ascending order of keys (A1 and B1), which are stored in the U-memory and L-memory, respectively. UADR and LADR present a sequence number for explaining the address control scheme of each memory. Fig.8(b) describes output tuples and (c) presents an execution process.

The processing algorithm of the JOIN-EQ command is as follows:

If A1 > B1, then UADR = UADR and LADR = LADR + 1

If A1 < B1, then UADR = UADR + 1 and LADR = LADR

If A1 = B1, then output a matched tuple pair

and

if the DP of A1 and the DP of B1 are on,

then UADR = UADR and LADR = LADR + 1

if the DP of A1 is on and the DP of B1 is off,

then UADR = UADR + 1 and LADR = LADR*

if the DP of A1 is off and the DP of B1 is on,

then UADR = UADR and LADR = LADR + 1

if the DP of A1 and the DP of B1 are off,

then UADR = UADR + 1 and LADR = LADR + 1

where DP stands for the duplication line and LADR* points to the first tuple of those which have the same values.

This algorithm is more efficient than a simple merge algorithm, because the duplication is considered to control the selection of the tuple to be processed. Other merger commands are also executed by using a similar algorithm to the above example.

The operation of the merger is also divided into three cycles. They are the 4 byte read cycle, compare-transfer cycle and ROM table read cycle. Each cycle takes 220 ns in accordance with the sorter.

The merger is characterized by its capability of providing high performance operations and processing the null values correctly.

5. Roles of the software

It is the engine core that performs most of the RDBE operations efficiently. The software, however, also plays important roles. It controls the engine core and the HM adapters to perform the RDBE operations and makes up for the lack of hardware functions if necessary.

The following is how RDBE works.

Upon receiving a request from the CP into its main memory, the control program analyzes it to produce the following parameters.

(1) A pattern of driving the engine core and the HM adapters.

(2) Requests to the HM for data transfer and so on.

(3) Set of parameters for the engine core.

(4) Object codes for the database processing by the CPU, if necessary.

Then RDBE sends the requests to HM and set up the engine core and the two HM adapters in succession according to the pattern.

The roles of the control program are as follows.

5.1 Control of the devices

In order to perform the RDBE operations by using the engine core, the control program drives the HM adapters to transfer data to/from the engine core. In the two-way merge operation, this is not easy because the relation to be supplied to the engine core is determined dynamically according to the state of the engine core. The control program is designed carefully to drive them without unessential delay.

The control program also uses the engine core in an interesting way. In the Delete operation where RDBE deletes those tuples the key of which match any of the condition values, the control program controls the engine core in the following steps.

(1) Load the condition values to the U-memory

(2) Output tuples which enter from the HM adapter(IN) and unmatch any of the condition values. (All tuples from the HM adapter(IN) are stored in the L-memory)

(3) Output tuples in the L-memory which match one of the condition values.

(4) Repeat steps(2) and (3) to the tuples in each page.

Fig.9 shows an example of the operation. It helps the rollback operation, because it updates each page where the deleted tuples are gathered below the ones not deleted.

In general, RDBE provides powerful operations, by setting up the engine core twice to the same data in different modes.

5.2 Auxiliary data processing

The operations which need data processing by the CPU are as follows.

(1) Selection with complicated conditions.

(2) Arithmetic operations.

(3) Aggregate operations.

The RDBE has a compiler which generates the native object code for data processing into the main memory in order to improve the efficency.

5.3 Error recovery

When an error occurs in a RDBE operation, rather complicated operations are required because the data runs through the engine core and the HM adapters which are connected to the HM.

For error recovery, the control program restarts the operation from some suitable retry point in contact with the HM.

5.4 Self check

During the power up sequence, the control program drives the engine core and the HM adapters with test data to check them. This is useful in order to improve the reliability.

6. Preliminary performance estimation of the RDBE

Estimation of the performance of Delta as a whole is very difficult because it does not work yet (Apr. 1984) and it is a complex multi-processor system. But the performance of the RDBE is easily estimated because the activities of the hardware and the software are very transparent. In this section, we estimate the performance of the typical operation, sort.

Let N be the number of tuples of the target relation, L be the length of a tuple and C be the effective capacity of the sorter. Then the following relation holds.

$$C = \min(4096L,\ 2^{16})$$

The performance can be estimated by classification of NL/C.

Case 1 $\quad NL/C \leq 1$

Since each module of the engine core works as shown in Fig.10(a), the total time T is estimated at (NL/1500 + 12) msec, where 12 msec is the sum of the time to analyze the request from the CP and the overhead time to drive the hardware devices.

Case 2 $\quad 1 < NL/C \leq 2$

According to Fig.10(b), T is (NL/1500 + C/3000 + 20) msec.

Case 3 $\quad 2^i < NL/C \leq 2^{i+1}$

The RDBE first sorts the original relation blockwise like in case 2. Then it repeats the two-way merge i-times to get the final result. T is approximately

$$((2.5 + 1.5i)NL/3000 + 10(i+1)NL/C + 10) \text{ msec},$$

where the first term is the time the engine core spends.

Fig. 11 shows the performance when L is 16 bytes.

7. Conclusions and future plans

We have described our design considerations and implementation of the RDBE which is a dedicated hardware used in Delta. The RDBE is characterized by the following advantages.

(1) High level operations.

(2) High performance by pipeline processing.

(3) Sufficient data types including null values.

(4) High reliability.

The RDBE was designed and implemented for about one year and a half by using currently available software and hardware technologies. It is presently (Spring 1984) incorporated in the Delta system and is undergoing a system test.

The activities planned for the future include the incorporation of multiple RDBEs into the Delta system in order to investigate parallel processing of the Delta commands, a gate-array implementation of the sorting cell without memories, and the implementation of an external interface function for connecting the Delta tightly to a sequential inference machine in order to be used as one of the software development tools in ICOT.

Acknowledgements

The present research effort is part of a major research and development project on the Fifth Generation Computer, conducted under a program set up by the Ministry of International Trade and Industry's Agency of Industrial Science and Technology.

We would like to express our sincere appreciation to the members of the ICOT KBM group for their valuable discussions, and to Dr. K. Mori, director of the TOSHIBA Information System Laboratory, who provided the opportunity to conduct the present research. We would like to thank the TOSHIBA development group, in particular K. Oda, T. Oka and S. Matsuda for their cooperation in designing and implementing the RDBE.

References

[1] Shibayama.S., et al: A Relational Database Machine with Large Semiconductor Disk and Hardware Relational Algebra Processor, ICOT Technical Report, TR-055, 1984.

[2] Codd.E.F. : A Relational Model of Data for Large Shared Data Banks, CACM, Vol.13, No.6, June, 1970.

[3] Gallaire.H. and Minker.J.(eds) : Logic and Data Bases, Plenum Press, 1978.

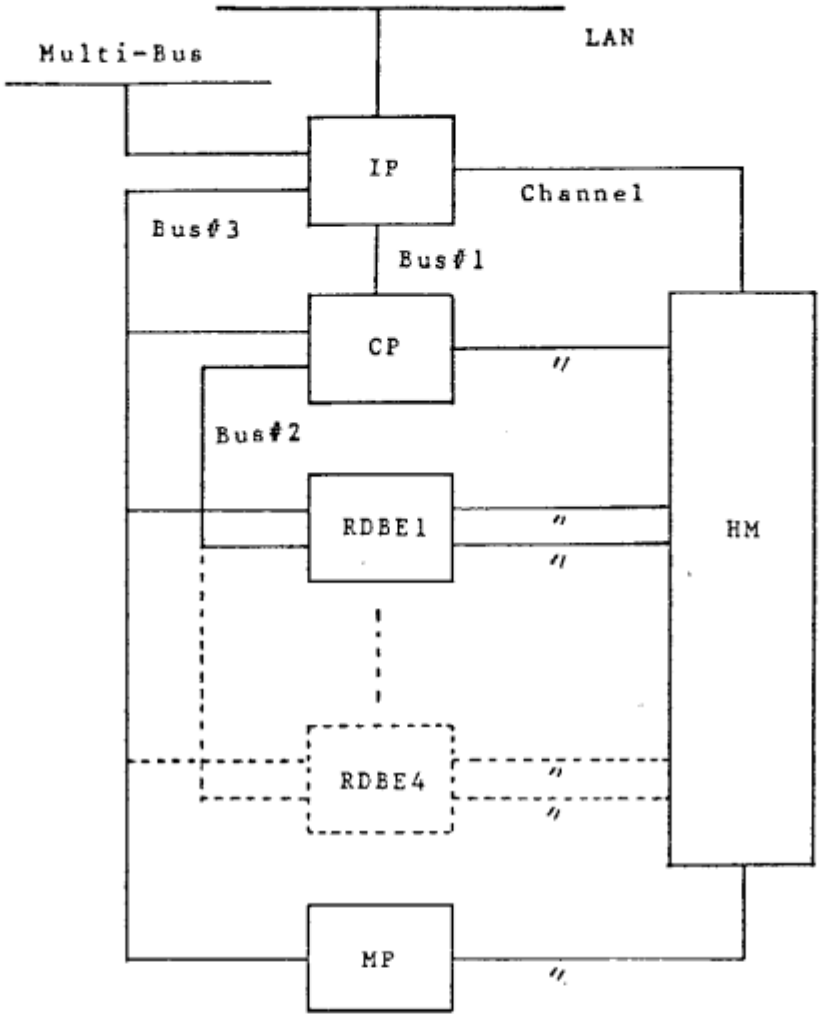
[4] Boral.H., et al: Implementation of the Database Machine DIRECT, IEEE Trans., Software Eng., Vol. SE-8, No.6, Nov., 1982.

[5] Hsio.D.K. (ed): Advanced Database Machine Architecture, Prentice-Hall, 1983.

[6] Knuth.D.E.: The Art of Computer Programming, Vol. 3 / Sorting and Searching, Addition-Wesley, 1973.

[7] Yasuura.H., et al: The Parallel Enumeration Sorting Scheme for VLSI, IEEE Trans. Comput., Vol. C-31, No.12, Dec., 1982.

[8] Todd.S.: Algorithm and Hardware for a Merge Sort Using Multiple Processors, IBM J. Res. Develop., Vol.22, No.5, Sept., 1978.



LAN : Local Area Network
IP : Interface Processor
CP : Control Processor
RDBE: Relational Database Engine
MP : Maintenance Processor
HM : Hierarchical Memory

Fig.1. Delta global architecture

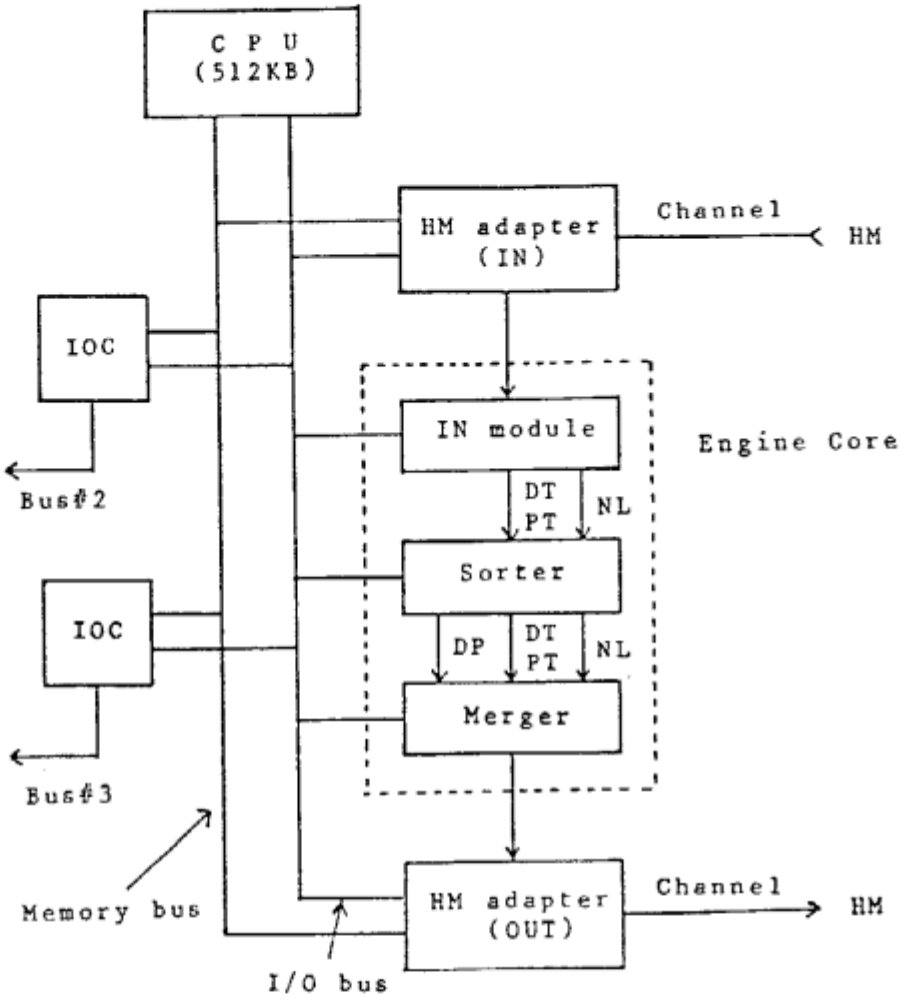


Fig.2. RDBE configuration

Name	Comments
PASS	
JOIN	=, ≠, <, ≦, >, ≧, Cartesian product
RESTRICT	=, ≠, range
SORT	ascending , descending
AGGREGATE	max,min,count,sum,average of an expression
UNIQUE	
UNION	
INTERSECTION	
DIFFERENCE	
EQUAL	
CONTAIN	
COMPARE	compare attributes of a relation
ZONE-SORT	for clustering
DELETE	for update

Fig.3. List of the RDBE operations

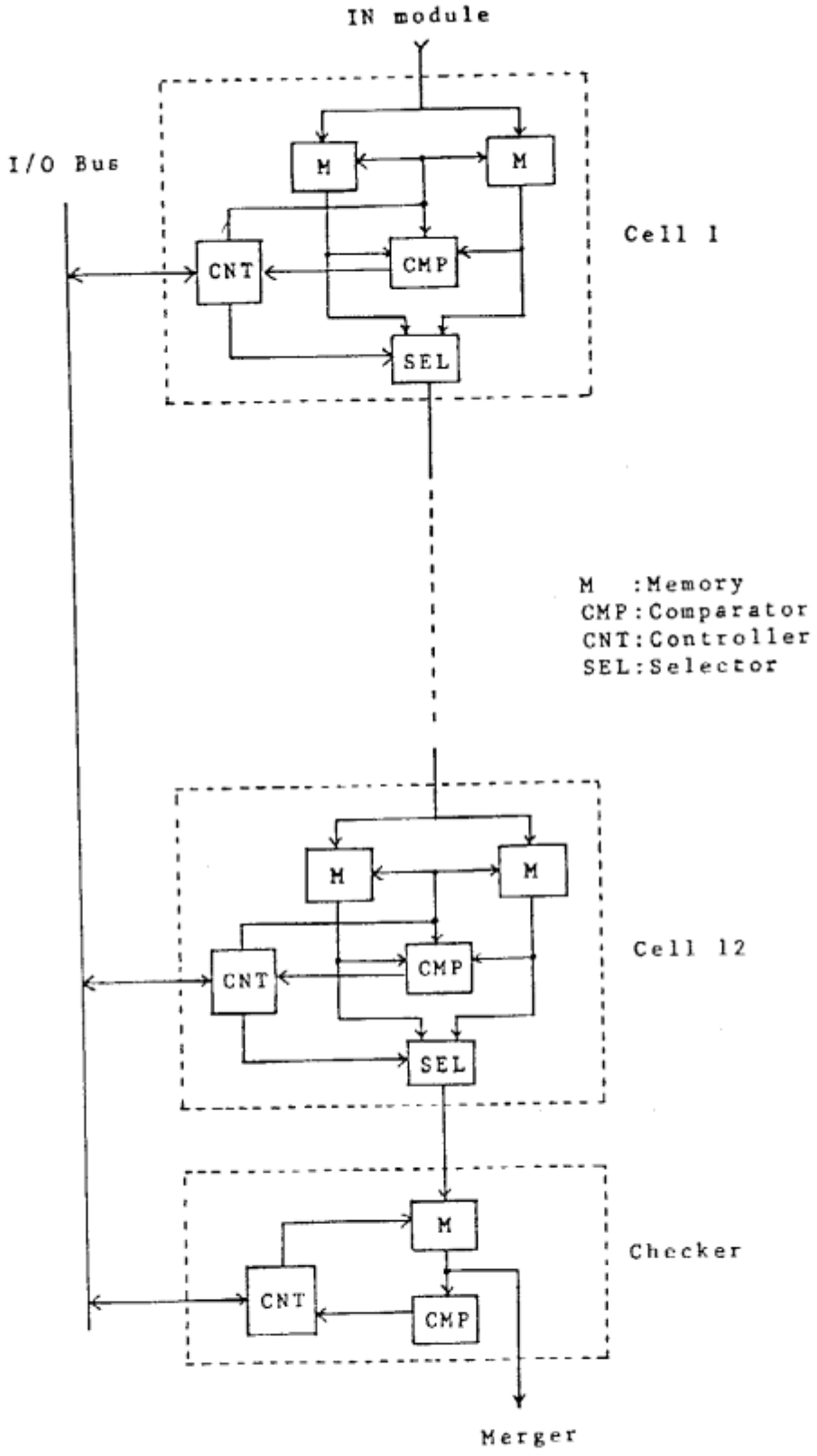


Fig.4. Block diagram of the sorter

PASS COMMAND	RESTRICT COMMAND
LOAD	REST-NULL
PASS-1 (NOP)	REST-NONULL
PASS-2 (UNQ-IN)	REST-EQ
SORT COMMAND	REST-NE
SORT-IN	REST-RANGE
SORT-EX	
UNQ-EX	
COMPARE ATTRIBUTES COMMAND	JOIN COMMAND
COMP-ALL	JOIN-ALL
COMP-NULL	JOIN-NULL
COMP-NONULL	JOIN-NONULL
COMP-EQ	JOIN-EQ
COMP-NE	JOIN-NE
COMP-LT	JOIN-LT
COMP-GT	JOIN-GT
COMP-LE	JOIN-LE
COMP-GE	JOIN-GE

NOP: No operation, UNQ-IN: Unique-internal,
EX: External, NONULL: Non null

Fig.5. List of the merger commands

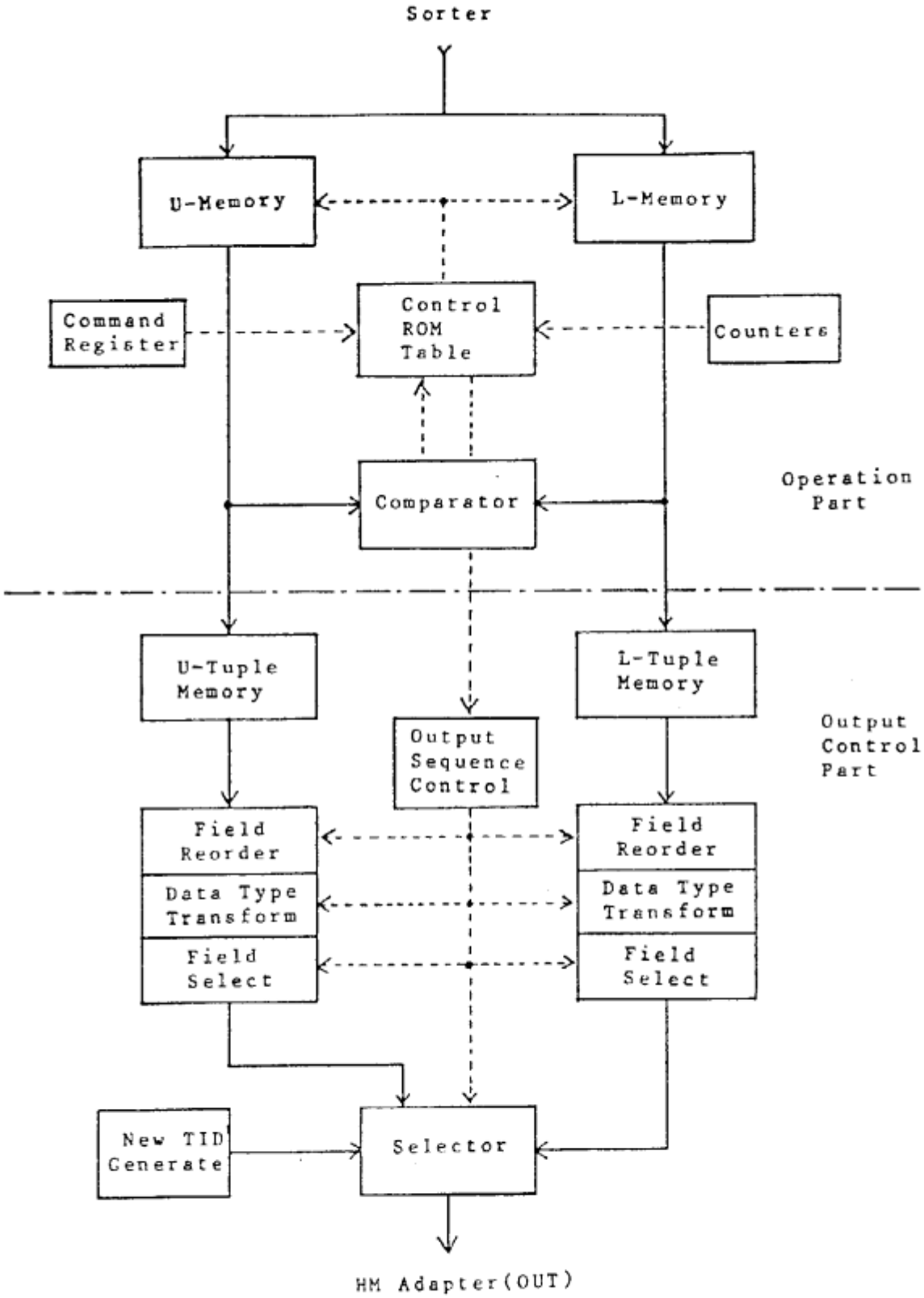


Fig.6. Block diagram of the merger

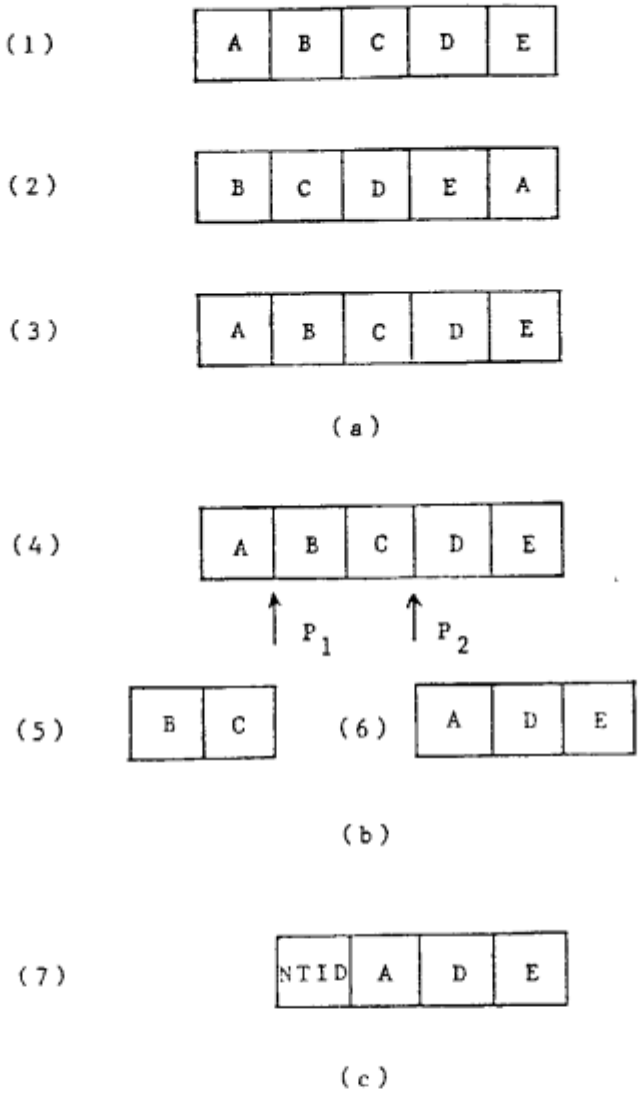


Fig.7. Functions of an output control part in the merger

(a)

STREAM S1 (U-MEMORY)

UADR	A_1	A_2
0	2	B
1	3	C
2	3	D
3	4	A

STREAM S2 (L-MEMORY)

LADR	B_1	B_2
0	1	T
1	3	S
2	3	V
3	5	U

JOIN A_1 = B_1

(b)

A_1	A_2	B_1	B_2
3	C	3	S
3	C	3	V
3	D	3	S
3	D	3	V

(c)

UADR	LADR	U-TUPLE	L-TUPLE	RESULT	OUTPUT
0	0	2 B	1 T	>	
0	1	2 B	3 S	<	
1	1	3 C	3 S	=	3 C 3 S
1	2	3 C	3 V	=	3 C 3 V
2	1	3 D	3 S	=	3 D 3 S
2	2	3 D	3 V	=	3 D 3 V
3	3	4 A	5 U	<	
END	3				

Fig.8. Processing example of the JOIN-EQ command

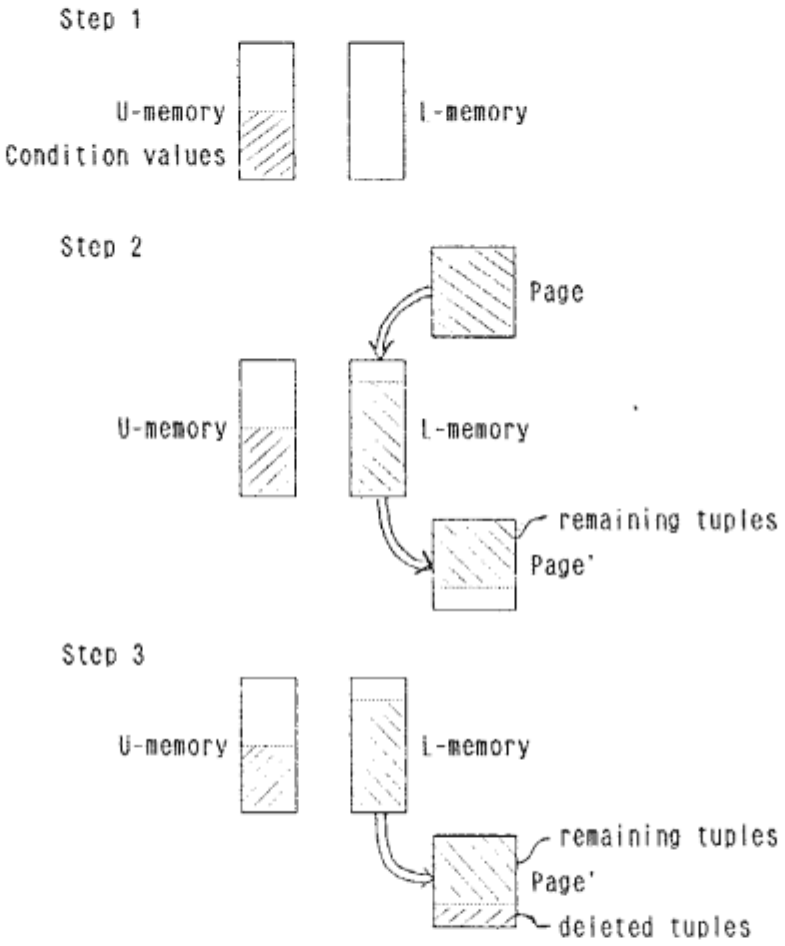


Fig.9. Delete operation by the RDBE

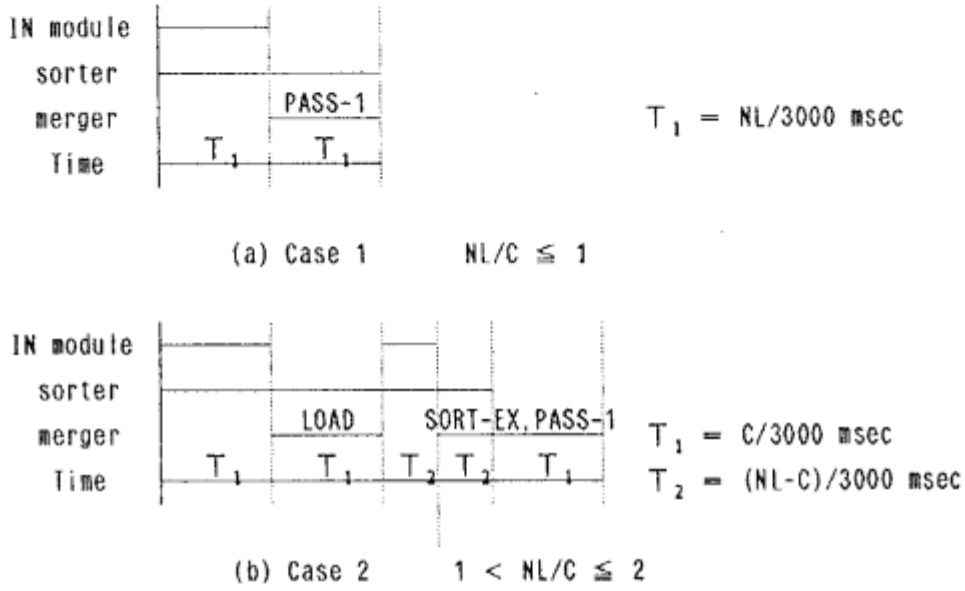


Fig.10. Activities of the engine core during the sort operation

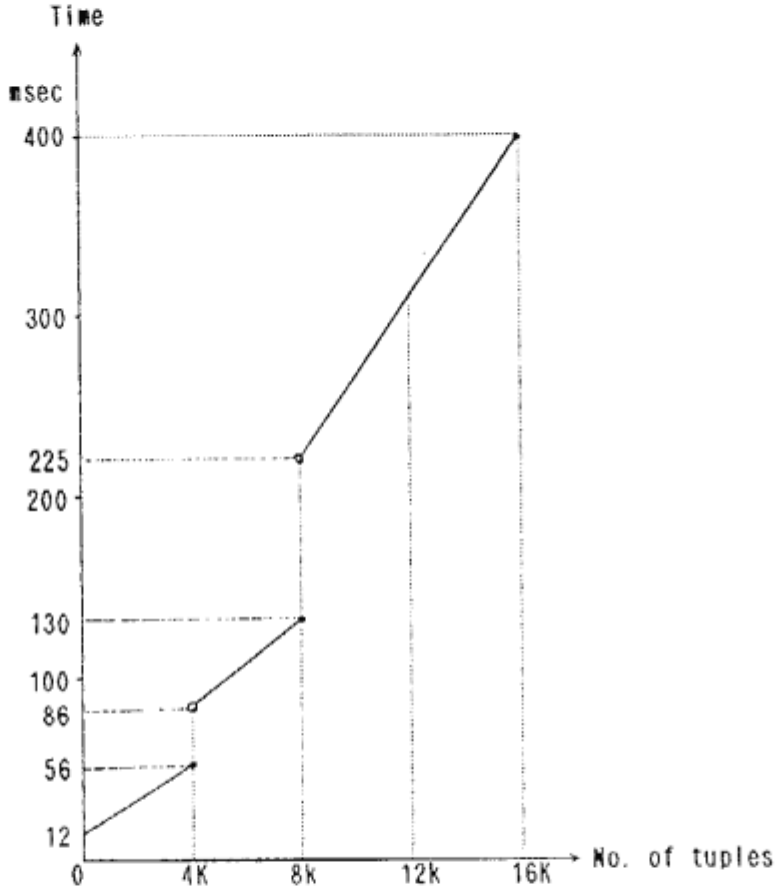


Fig.11. The performance of the sort operation

We would like to express our sincere appreciation to the members of the ICOT KBM group for their valuable discussions, and to Dr. K. Mori, director of the TOSHIBA Information System Laboratory, who provided the opportunity to conduct the present research. We would like to thank the TOSHIBA development group, in particular K. Oda, T. Oka and S. Matsuda for their cooperation in designing and implementing the RDBE.

References

[1] Shibayama.S., et al: A Relational Database Machine with Large Semiconductor Disk and Hardware Relational Algebra Processor, ICOT Technical Report, TR-055, 1984.

[2] Codd.E.F. : A Relational Model of Data for Large Shared Data Banks, CACM, Vol.13, No.6, June, 1970.

[3] Gallaire.H. and Minker.J.(eds) : Logic and Data Bases, Plenum Press, 1978.

[4] Boral.H., et al: Implementation of the Database Machine DIRECT, IEEE Trans., Software Eng., Vol. SE-8, No.6, Nov., 1982.

[5] Hsiao.D.K. (ed): Advanced Database Machine Architecture, Prentice-Hall, 1983.

[6] Knuth.D.E.: The Art of Computer Programming, Vol. 3 / Sorting and Searching, Addition-Wesley, 1973.

[7] Yasuura.H., et al: The Parallel Enumeration Sorting Scheme for VLSI, IEEE Trans. Comput., Vol. C-31, No.12, Dec., 1982.

[8] Todd.S.: Algorithm and Hardware for a Merge Sort Using Multiple Processors, IBM J. Res. Develop., Vol.22, No.5, Sept., 1978.