TR-295

Design,Implementation, and Evaluation of a Relational Database Engine for Variable Length Records

by
F. Itoh, H. Itoh, M. Oba, K. Shimakawa
K. Togo & S. Matsuda

August ,1987

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, IMPLEMENTATION, AND EVALUATION OF A RELATIONAL DATABASE ENGINE FOR VARIABLE LENGTH RECORDS

F. ITOH*# K. SHIMAKAWA* K. TOGO† S. MATSUDA† H. ITOH‡
M. OBA‡

*Information and Communication Systems Laboratory, Toshiba Corporation, 2-9 Suehiro-cho, Ome, 198, Japan
†Ome Works, Toshiba Corporation, 2-9 Suehiro-cho, Ome, 198, Japan
‡ICOT Research Center, Institute for New Generation Computer Technology, Mita Kokusai Bldg., 21F, 1-4-28 Mita, Minato-ku, Tokyo 108, Japan

ABSTRACT

This paper reports the design, implementation, and evaluation of a relational database engine. The engine has special purpose hardware, the nucleus of which is a pipeline two-way merge sorter, to execute sorting and relational algebra for variable length records. In the execution of these operations, key fields are extracted from records. The engine has on-the-fly processing in which it inputs data directly from disk. The evaluation shows the characteristics of processing time, the effect of variable length records, and the effectiveness of on-the-fly processing.

INTRODUCTION

The Institute for New Generation Computer Technology (ICOT) considers that fifth generation computer systems have the integrated functions of inference and knowledge bases, and has been researching and developing inference machines (IMs) and knowledge base machines (KBMs). For KBMs, a relational database machine, Delta (1,2), was developed because the relational database is said to be suitable for logic programming which is used in IMs. In Delta, facts, as in Prolog, are stored as relations and retrieved by relational algebra. Another KBM is now under development, in which Horn clauses are stored as relations and retrieved by operations called retrieval-by-unification (RBU) (3). In RBU operations, data is selected and joined based on unifiability, in contrast to relational algebra, in which data is selected and joined based on equality. The KBM has unification engines (UEs) (4), which perform RBU operations. Although a Horn clause is represented as a structure which consists of literals and variables, UEs treat the structure as a variable length string.

With this background, we have developed a relational database engine (RDBE) with following aims.

#Currently working at Institute for New Generation Computer Technology

Processing variable length records using hardware

The RDBE performs sorting and relational algebra (RA) operations for one fixed or variable length key field in variable length records.

Efficient database processing

On-the-fly processing. To process data on disk efficiently, the RDBE has on-the-fly processing in which it inputs data directly from disk. Without on-the-fly processing, data on disk is loaded to main memory (MM) before it is input to the RDBE, then the data path in the entire processing is "disk → MM → RDBE → MM". On-the-fly processing makes the data path "disk → RDBE → MM", and achieves more effective processing.

Stream processing in the RDBE. The RDBE realizes the stream processing of sorting and RA operations.

The search processor, SHP (5), is an example of a processor which has on-the-fly processing. Examples of processors which have stream processing are VLSIs based on the systolic array (6), the RDBE in Delta (7), multiple processors for the join operation (8), and some processors for sorting, for example, a sort engine using pipeline heap sorting (9) and a pipeline merge sorter (10). An algorithm has been proposed for a sorting processor for variable length records (11), but has not been implemented yet.

Later sections discuss the design, implementation, and performance evaluation of the RDBE. Section 2 presents the basic ideas of design and features of the RDBE. Section 3 summarizes the configuration, format of data to be processed, functions, and processing. Section 4 shows the implementation of all components of the RDBE. Section 5 evaluates performance based on the design values and measurement.

BASIC IDEAS

Configuration of the Database Machine

Fig. 1 shows the configuration of the database machine. It mainly consists of a central processing unit (CPU), MM, RDBE, and database disk. Relations manipulated by the database machine have both fixed and variable length attributes. Relation schemata and data are stored in different files on disk. Data of one relation is stored in one file. One record in a file corresponds to one tuple in a relation. Fig. 2 shows the structure of a record.

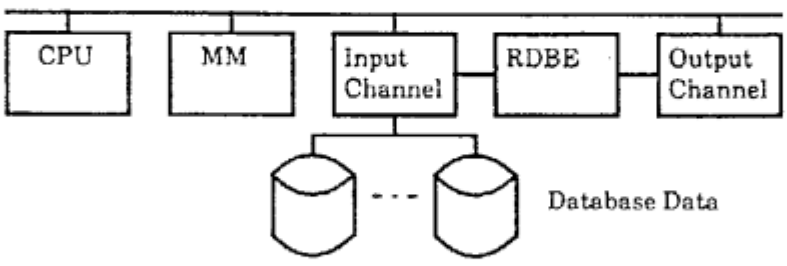


Fig. 1. Configuration of the Database Machine

A record consists of a record header (RH), fixed length fields (FFs), variable length field headers (FHs), and variable length field bodies (VFs). There are the same number of FHs and VFs, and they have one-to-one correspondence in order. The RH consists of record length (RL) and a deleting flag (DF). A DF shows whether the record is valid (logically not deleted). An FH consists of field length (FL) and field position (FP). FL denotes the length of the corresponding VF. FP means the length from the head of a record to the head of the corresponding VF. A VF has an instance of a variable length attribute. Relation data on disk is a series of records. Relation schemata include the length and the position of each FF and the position of each FH, in addition to ordinary schemata.

Data processing is shared between the RDBE and the CPU. When a query arrives at the CPU, the CPU executes a combination of the following three types of processing to obtain the answer, and returns the answer to the user.

The CPU requests the RDBE to input data from disk, process it, and output the result to MM.

The CPU requests the RDBE to input data from MM, process it, and output the result to MM.

The CPU directly processes data in MM.

The RDBE quickly performs the following processing, which is simple but time-consuming.

Sorting. Records in target data are sorted in ascending or descending order by their key values.

Duplication elimination. Records in target data are sorted in ascending or descending order. If there are two or more records which have the same key value, one is left and the others are removed.

Selection. Records are selected from target data whose key values have a specified large or small relationship to the conditional value.

Intersection and difference. Records are selected from second target data whose key values are (in intersection) or are not (in difference) included in the set of key values of first target data.

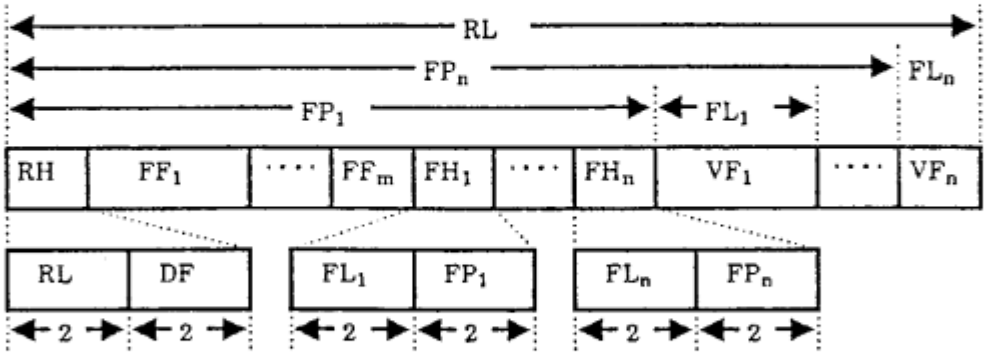


Fig. 2. Format of a Variable Length Record

Equal join. Pairs of records are made between first and second target data, each of which has the same key values. However, the creation of one record from each pair is not supported by the RDBE, but by the CPU.

The CPU performs following processing which is too complex for hardware or causes a data transfer bottleneck although the processing by hardware is quick.

Arithmetic operations, aggregate functions, and reforming records after join in the RDBE. These are too complex for hardware.

External sorting and external duplication elimination. In sorting and duplication elimination, if the volume of target data is beyond the RDBE's capacity, data is divided into several parts less than the RDBE's capacity, the RDBE sorts or eliminates duplications from each part, and the CPU performs external sorting or duplication elimination of all parts. Even if the RDBE could perform external sorting and duplication elimination, there would be a data transfer bottleneck.

Processing Methods of the Relational Database Engine

The RDBE adopts the following methods:

Stream processing by a pipeline two-way merge sorter and an RA processor. The sorting and relational algebra processing (RAP) modules are arranged in series. The sorting module consists of a pipeline two-way merge sorter (12). In the intersection, difference, and join operations, the comparison to search for equivalent or non-equivalent key values is executed more effectively by sorting key values beforehand.

Key field extraction processing. Only the key fields flow into the sorting and RAP modules. In pipeline two-way merge sorting, memories are used to store data being sorted. Different record lengths cause delays in pipeline processing. To reduce the contents of memory and the delays, the key fields are extracted from records.

Use of tags. The sorting and RAP modules use tags attached to data to execute processing. One item of information in tags indicates the end of key fields. With this information, it is not necessary to consider whether the key field length is fixed or variable.

OVERVIEW OF RELATIONAL DATABASE ENGINE

Configuration and Data Format

Fig. 3 shows the configuration of the RDBE and its data format. The RDBE consists of a controller and four modules. There are three data formats as follows,

Records. The input and output data of the RDBE is the records shown in Fig. 2. Input records are sent to the RB and IN modules simultaneously, and output records are read from the RB module. Records are input and output in two-byte units.

Key fields and record identifiers with tags. Data flowing from the IN module to the RAP module consists of key fields and record identifiers (RIDs) with tags. Key fields and RIDs flow one after the other. Each RID corresponds to the record from which the preceding key field is extracted. A one-byte tag is added to each two bytes of key fields or RIDs. Tags are used for flags and parities. Flags indicate the end of key fields and the duplication of key values. Parities are used as bit parity for data integrity. Three bytes of key fields or RIDs and tags flow in parallel.

RIDs with tags. RIDs and their tags flow between the RAP and RB modules. The content of tags is parities. Three bytes of key fields and tags flow in parallel.

Controller and Module Functions

The function of the controller and each module are as follows:

Controller. The controller interprets a command from the CPU and sets up modules. In the IN module, it determines whether key fields are of fixed or variable length, their position and length if they are of fixed length, and the position of the corresponding variable length field headers if of variable length. In the IN module, it determines the key data type (character, integer, or floating point). In the sorting module, it determines whether sorting is used in the operation processing, and whether sorting is ascending or descending. In the RAP module, it determines the kind of RA.

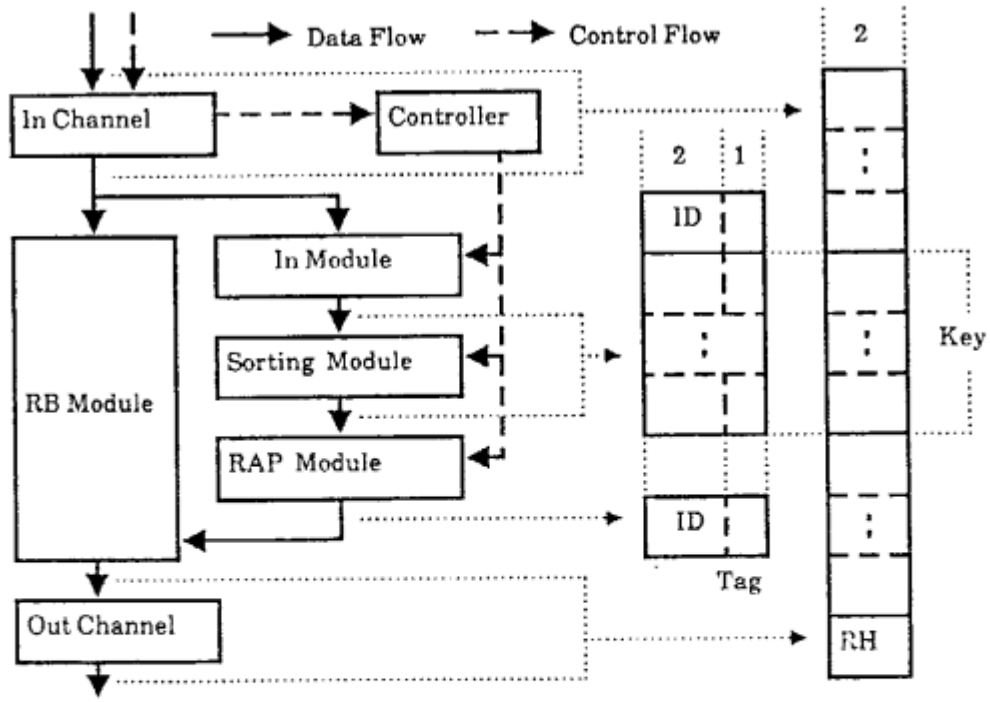


Fig. 3 Configuration and Data Format

RB module. When target data is input, the RB module stores whole records in memory, and determines the correspondence between the input serial number (used as the RID) of each record and its address in memory. When result data is output, the RB module reads the records from memory which correspond to RIDs sent from the RAP module.

IN module. The IN module extracts key fields from records, attaches RIDs, and sets key end flags. If the data type of the key is numerical (integer or floating point), it manipulates bits of key values to compare them as a character.

Sorting module. The sorting module sorts character-type key values in ascending or descending order. If there are two or more of the same key values, it sets duplication flags on all the key values other than the last one.

RAP module. The RAP module executes duplication elimination, selection, intersection, difference, and equal join operations for character-type key values. Except for selection, it requires that key values be sorted and it uses duplication flags. As a result, it outputs only RIDs.

Processing Method for Sorting and Relational Algebra Operations

Sorting and RA operations are executed with the sorting and RAP modules as follows:

Sorting. The sorting module sorts target key values. The RAP module only extracts RIDs from sorted data.

Duplication elimination. The sorting module sorts target key values. The RAP module removes redundancies from duplicated data by selecting data on which duplication flags are not set.

Selection. First, a conditional key value flows through the sorting module, and is stored in memory of the RAP module. Next, target key values flow through the sorting module, but are not sorted. The RAP module compares them in input order with the conditional key value, and extracts target RIDs whose key values satisfy the condition.

Intersection and difference. The first target key values are sorted by the sorting module and stored in memory of the RAP module. The second target key values are also sorted and flow into the RAP module. The RAP module compares the key values of the two targets and extracts the second target RIDs whose key values are (in intersection) or are not (in difference) included in the set of first target key values.

Equal join. The first target key values are sorted by the sorting module and stored in memory of the RAP module. The second target key values are also sorted and flow into the RAP module. The RAP module compares the key values of the two targets and extracts pairs of the first and the second target RIDs whose key values are equivalent.

DESIGN AND IMPLEMENTATION

Record Buffer Module and IN Module

Memory for storing whole records is one megabyte. The RID length is two bytes. In one operation, the maximum size of whole target records is one megabyte and the maximum number of target records is 64 kilobytes. A table controls the correspondence between RIDs and addresses in memory. In some operations, typically selection, result record output starts before target record input ends, thereby realizing concurrent record input and output.

Sorting Module

The pipeline two-way merge sorter consists of 12 sorting cells. The maximum number of target records in sorting is 4096 (2^{12}). Fig. 4 a) shows the configuration of a sorting cell. The memory management method is double memory (11) for ease of implementation. Memory in the 12th sorting cell is 128 kilobytes (in practice, 192 kilobytes including 64 kilobytes for the tags). Essentially, in the double memory method, two items of data to be merged are stored in district memory. However, in our implementation, they are stored in two parts of one physical memory because of the hardware size restriction.

Relational Algebra Processing Module

Fig. 4 b) shows the configuration of the RAP module. There are two memories, each of which is 128 kilobytes (in practice, 196 kilobytes). In intersection, difference, and equal join, all first target key values are stored in one memory. Comparison of the first and second target key values begins as soon as the second target key values flow into the RAP

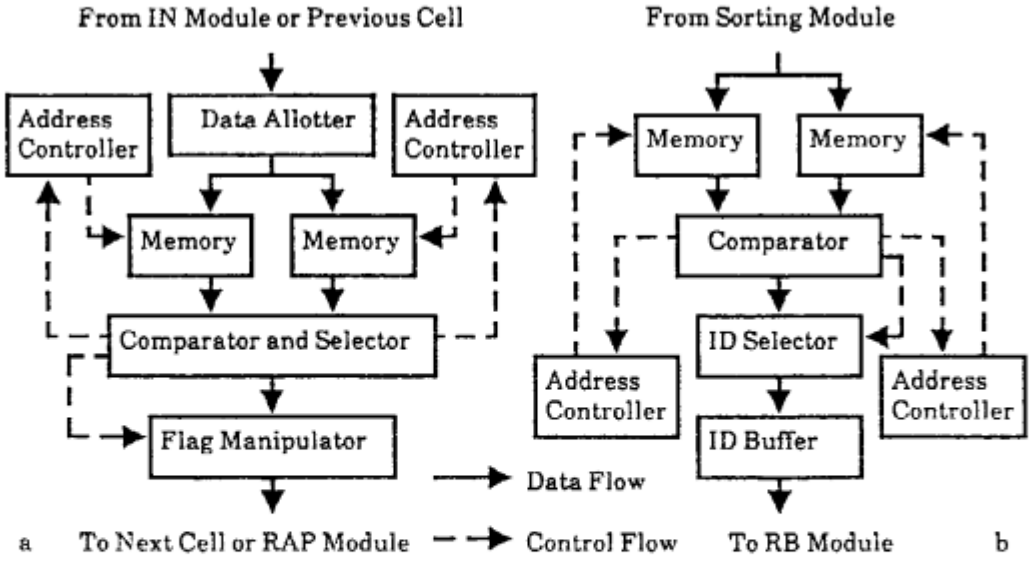


Fig.4 a) Configuration of a Sorting Cell b) Configuration of the RAP Module

module. If the speed of second target key values flowing into the RAP module is faster than the speed of the values being cast off after comparison, second target key values are stored in another memory. If another memory becomes full, the RAP module stops second target key values from flowing in until a vacancy occurs. The maximum size of first target key values and RIDs is 128 kilobytes, but the size of second target key values is not limited.

Clock Time of Data Flow

In the RDBE, data are transferred in two-byte or three-byte units, including one byte for the tag (this two or three bytes is called one word). The data path in each module is made up of registers and memories. The data transfer speed depends on memory access time because data transfer between registers and comparison of two items of data are faster than memory access.

RB module and IN module. Data input into the RB module and the IN module are synchronized. In the RB module, alternate input at word level and output to memory is adopted to realize concurrent input and output at record level. Essentially, data is input and output at a speed of one word per two clocks. In the IN module, key fields are transferred at a speed of one word per three clocks because, as described later, the sorting module receives them at that speed. Parts other than key fields are cast off without being received by the sorting module.

Then, data is input into the RB and IN module at one word per three clocks in key fields, and one word per two clocks in parts other than key fields. Data is output from the RB module at one word per two clocks or a little slower because read from memory waits if requirements of both write and read occur in the same clock.

Sorting module. To realize pipeline processing, each sorting cell must transfer data at the same speed. Consider that each sorting cell receives and sends one word for a certain time, and that there are three memory accesses in each sorting cell. Two of them are read of both words to be compared, and one is write of a word from the preceding cell. These three memory accesses are assigned every three clocks. Data is transferred at one word per three clocks.

RAP module. Essentially, it is possible to transfer data at one word per two clocks. In intersection, difference, and equal join, both words to be compared are read simultaneously because two memories are physically separated from each other, unlike the sorting module. However, for ease of implementation, data is transferred at one word per three clocks, synchronized with output from the sorting module.

Actual Implementation

The modules are controlled by transistor-transistor logic (TTL) and a programmable logic device (PLD). One clock is 250 ns.

PERFORMANCE EVALUATION

Basic Performance

The performance is evaluated when the lengths of records and keys are fixed. First, the processing time of each operation is estimated by examining the register transfer level. The processing time is from beginning input of data to ending output, and does not include the time to interpret a command from the CPU and to set up modules.

Let each record consist of only a variable length key field, and the length of the key field be 14 bytes or more, and be $2a$ bytes. The length of each record is $2a+8$ bytes, including four bytes for a record header and four bytes for a key field header.

Sorting. Let the number of target records be n (where n is a power of 2). The processing time of sorting is divided into input time, delay time, and output time. Each time is defined and summarized as follows.

Let input time be from beginning input of the first record to beginning input of the last (nth) key field. It takes eight clocks from beginning input of the first record to input of the first key field, because there are four words for the record header and key field header. The interval between inputting a key field and inputting the next key field is, essentially, $3a+8$ clocks because an a-word key and a four-word non-key (the record header and key field header of the next record) are input. The RID is transferred while the non-key is cast off. However, as stated above, because a clock where the sorting module receives a word is assigned once every three clocks, the interval becomes $3a+9$, which is the minimum multiple of three from $3a+8$ up. Then, input time is:

$$(3a+9)(n-1)+8 \ (clocks)$$

Let delay time be from beginning input of the last key field to output of the first record. It takes 91 clocks for data to flow through the engine without any processing for operations. However, the RAP module only outputs the RID ignoring the key (it takes $3a$ clocks). Then, delay time is:

$$3a+91 \ (clocks)$$

Let output time be from beginning output of the first record to ending output of the last record. The interval of the RID being output from the RAP module is $3a+3$ clocks because there are a words per key and one word per record ID. The output of one record finishes in $3a+3$ clocks because it takes $2a+8$ clocks, less than $3a+3$ clocks, assuming that $a \geqq 7$. Then, output time is:

$$(3a+3)(n-1)+2a+6 \ (clocks)$$

From the above, the total time of sorting is:

$$(6a+12)n-a+93 \ (clocks)$$

The outline of other operations is as follows.

Duplication elimination. Duplication elimination is the same as sorting.

Selection. Let the record number of target data be n.

$$(3a+9)n+5a+178 \ (clocks)$$

Intersection and difference. Let the record number of first and second target data be m and n respectively (where both are a power of 2 and 4096 or less).

Minimum:

$$(6\alpha+12)m+(6\alpha+12)n-4\alpha+163 \ (clocks)$$

Maximum:

$$(9\alpha+15)m+(6\alpha+12)n-4\alpha+163 \ (clocks)$$

Equal join. Let the record number of first and second target data be m and n respectively (where both are a power of 2 and are 4096 or less). Let the record number of result data be r (where $r \geqq n$).

Minimum:

$$(6\alpha+12)m+(3\alpha+9)n+(4\alpha+20)r-3\alpha+156 \ (clocks)$$

Maximum:

Where $r \leqq (\min[m,n])^2$, let r be a square number.

$$(6\alpha+12)m+(3\alpha+9)n+(4\alpha+20)r+(3\alpha+3)(m+n-2\sqrt{r})-3\alpha+156 \ (clocks)$$

Where $r > (\min[m,n])^2$, let r be a multiple of min[m,n].

$$(6\alpha+12)m+(3\alpha+9)n+(4\alpha+20)r+(3\alpha+3)(max[m,n]-\frac{r}{min[m,n]}-3\alpha+156 \ (clocks)$$

Let each record consist of more than one field. Assume the following:

A key field is the last part of a record.

The length of a key is 2α bytes, and the length of a record is 2β bytes. $\beta-\alpha$ is multiple of 3, and $3\alpha \leqq 2\beta-4$.

The number of target records is n and a power of 2.

The processing time for sorting is as follows.

$$(\alpha+4\beta+2)n+4\alpha-2\beta+99 \ (clocks)$$

The processing time has the following characteristics.

It is linear to the number of target records.

It is linear to the key length and the record length, and depends more on the record length.

In selection, the processing time does not depend on the selectivity.

In intersection and difference, the variation in processing time depends on the number of the same key values in first and second target data. The larger the number, the shorter the processing time, because the number of comparison times in the RAP module decreases.

In join, the processing time is approximately linear to the number of result records. The variation depends on the concurrency between the comparison in the RAP module and output of the result records. If there are the same key values in first and second target data in the later part of the comparison, output of the result records continues after the end of the comparison, making the processing time longer.

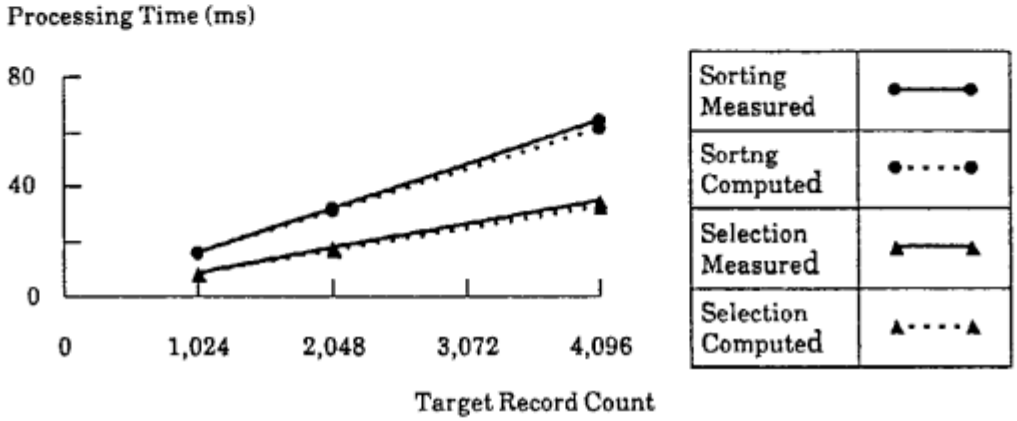
Next, the real processing time is compared with the computed time. The time from beginning input of records to ending output of records is measured by a logic analyzer. The measured time is approximately equal to the computed time. Fig. 5 shows the time for sorting and selection, and Fig. 6 shows that for equal join. The difference between the

measured and computed times is caused by firmware overhead in the input and output channels.

Effect of Variable Length Records

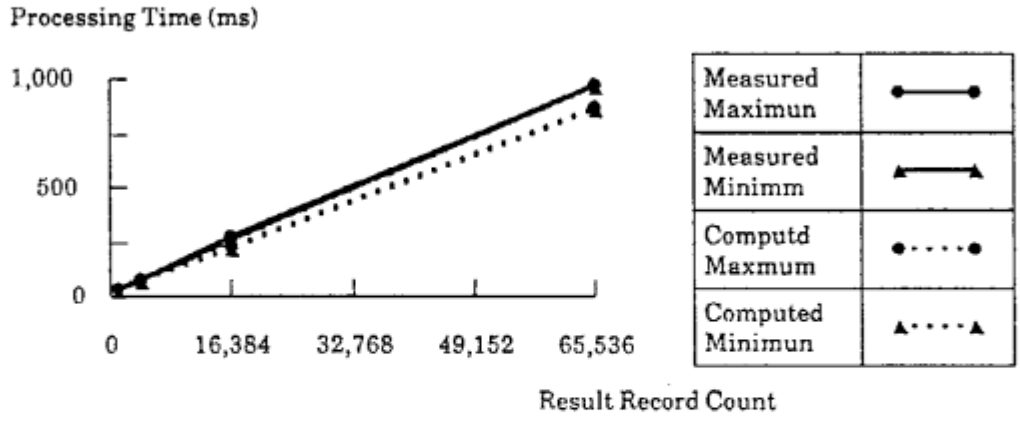
Processing time is measured for target records where record lengths, key lengths, or both are different. In sorting, the time from beginning input of records to ending output of records is measured by a logic analyzer.

Generally, it takes longer to sort variable length records than fixed length records, because the difference of processing time in each sorting cell causes a delay in the data stream. Processing time is measured for the following three cases. In each case, the number of records, the total size of records, and the total size of key fields are constant.



Key Length 16 Bytes (Constant)
Record Length 24 Bytes (Constant)

Fig. 5. Target Record Count Characteristic in Sorting and Selection



Key Length 16 Bytes (Constant)
Record Length 24 Bytes (Constant)
First Target Record Count 512
Second Target Record Count 512

Fig. 6. Result Record Count Characteristic in Equal Join

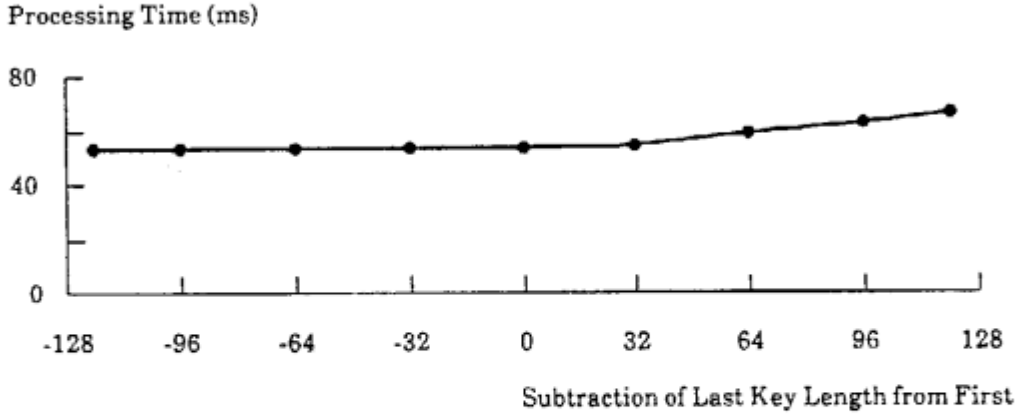
Case 1. The target records consist of only variable length key fields. The key and record lengths become gradually longer, are constant, or become gradually shorter. The gradient varies.

Case 2. The key lengths of the target records are constant. The record lengths of target records become gradually longer, are constant, or become gradually shorter. The gradient varies.

Case 3. The record lengths of the target records are constant. The key lengths of target records become gradually longer, are constant, or become gradually shorter. The gradient varies.

Fig 7 shows the measurement results in case 1. Where lengths become longer, the processing time is the same as where they are constant. However, where lengths become shorter, the greater the gradient, the longer the processing time. In cases 2 and 3, the processing time is equal among the various gradients.

Consider the last sorting cell of the sorting module. It stores the sorted first half of the keys in its memory, and when it obtains the first key of the sorted second half of the keys, it starts output of all of the sorted keys. Storage of the sorted first half of the keys begins after input of the first half of the target records to the IN module finishes. The first key of the sorted second half of the keys is obtained after input of all of the target records to the IN module finishes. Where storage of the sorted first half of keys finishes before input of all of the target records to the IN module finishes, processing time of sorting depends on the total size of the records, which is constant in all cases. However, where the key and record lengths become shorter in case 1, the total size of the first half of the keys is larger than the total size of the second half of the records, which causes longer processing time.



Average Key Length 16 Bytes (Constant) Record Count 1024
Average Record Length 24 Bytes (Constant)

Fig. 7. Target Record Length Distribution Characteristic in Sorting

Effect of On-the-fly Processing

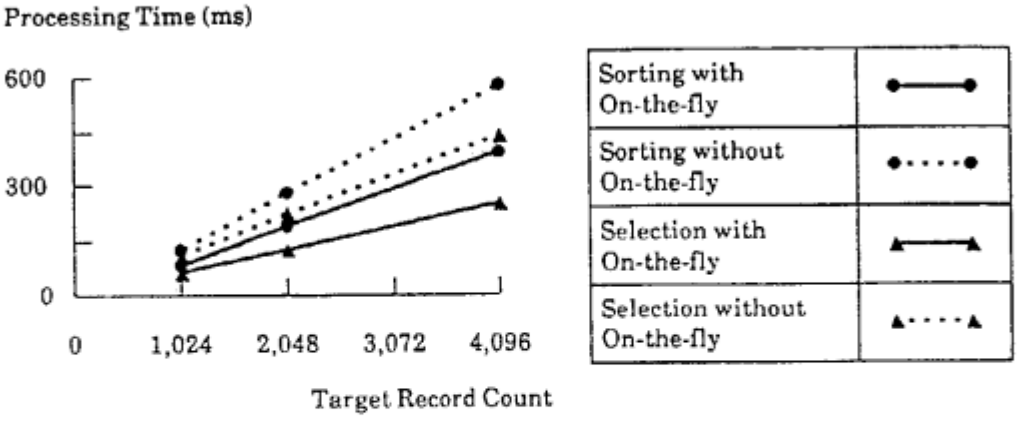
Processing time with and without on-the-fly processing is compared, and the effect of on-the-fly processing is evaluated. With on-the-fly processing, data on disk is directly input to the RDBE. Without it, data on disk is input to the RDBE after it is loaded to MM. Fig. 8 shows the processing time of sorting and selection with and without on-the-fly processing. On-the-fly processing reduces the processing time by about 30% in sorting and by about 40% in selection. The processing time is regarded as the time when the data is being transferred, and is measured by a logic analyzer.

The effect of on-the-fly processing is evaluated. To make the discussion simple, it is assumed that the speed of reading data from disk (disk → MM and disk → RDBE) and that of transferring data between the MM and the RDBE (MM→ RDBE and RDBE → MM) are the same.

Since the processing in the RDBE is concurrent with inputting and outputting data to it, the entire processing time is roughly estimated by the data transfer time.

In sorting, the result data is output from the RDBE almost immediately after the target data is input to it. With on-the-fly processing, there are two data transfers (disk → RDBE and RDBE → MM). Without on-the-fly processing, there are three transfers (disk → MM, MM → RDBE, and RDBE→ MM). Since the time of these data transfers is approximately the same, the processing time of sorting with on-the-fly processing is about two-thirds of that without on-the-fly processing.

In selection, input of target data to the RDBE and output of result data from it are approximately concurrent. With on-the-fly processing, there is one data transfer (disk→ RDBE → MM). Without on-the-fly processing, there are two transfers (disk→ MM and MM → RDBE→ MM). Since the time of these data transfers is approximately the same, the processing time of selection with on-the-fly processing is about half of that without on-the-fly processing.



Key Length 16 Bytes (Constant)
Record Length 64 Bytes (Constant)

Fig. 8. With and Without On-the-fly Processing in Sorting and Selection

CONCLUSION

This paper described the design, implementation, and performance evaluation of the RDBE for variable length records. To process variable length records, the key fields are extracted from records, and tags are added to them. This realizes the same algorithm for both fixed and variable length data, and reduces the delay caused by different record lengths. To execute sorting and RA operations quickly, pipeline processing by multiple processors, the nucleus of which is a two-way merge sorter, is adopted. This realizes stream processing, the concurrent execution of an operation and transfer of data. To process data on disk effectively, on-the-fly processing is used so that the RDBE inputs data directly from disk. This reduces the total processing time by about 30% in sorting and by about 40% in selection in comparison with the case where data are loaded to MM before inputting them to the RDBE.

ACKNOWLEDGMENTS

We wish to thank Dr. K. Iwata and Mr. C. Sakama of ICOT Research Center, and Mr. Y. Hoshino, Mr. S. Shibayama, and Mr. H. Sakai of Toshiba Corporation for useful discussions. We also extend our thanks to the RDBE developers for the implementation and performance measurements of the RDBE.

REFERENCES

1. Shibayama, S., Kakuta, T., Miyazaki, N., Yokota, H., and Murakami, K. New Generation Computing, Vol.2, 2:131-155, 1984.
2. Kakuta, T., Miyazaki, N., Shibayama, S., Yokota, H., and Murakami, K. In: Database Machines Fourth International Workshop (Eds. D. J. DeWitt and H Boral), Springer-Verlag, New York, 1985, pp. 193-34.
3. Yokota, H., and Itoh, H. Proc. 13th International Symposium on Computer Architecture, pp. 2-9, 1986.
4. Morita, Y., Yokota, H., Nishida, K., and Itoh, H. Proc. 12th International Conference on Very Large Data Bases, pp. 52-59, 1986.
5. Hayami, H., and Inoue, U. IPS Japan Technical Report, DB-51-2, 1986 (in Japanese).
6. Kung, H. T., et al. ACM SIGMOD, pp. 105-116, 1980.
7. Sakai, H., Iwata, K., Kamiya, S., Abe, M., Tanaka, A., Shibayama, S., and Murakami, K. Proc. of International Conference on Fifth Generation Computer Systems 1984, pp. 419-426, 1984.
8. Valduriez, P., and Gardarin, G. ACM Trans. Database Syst., Vol.9, 1:133-161, 1984.
9. Tanaka, Y., et. al. Proc. of IFIP Congress, pp. 427-432, 1980.
10. Kitsuregawa, M., Fushimi, S., Kuwabara, K., Tanaka, H., and Moto-oka, T. Trans. IECE Japan (Section J), Vol.J66-D, 3:332-339, 1983 (in Japanese).
11. Yang, W., Kitsuregawa, M., and Takagi, M. IPS Japan Technical Report, CA-63-12, 1986 (in Japanese).
12. Todd, S. IBM J. Res. Dev., Vol.22, 5:509-517, 1978.