

TM-0300

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

by

F. Itoh, K. Shimakawa, K. Togo, S. Matsuda  
(Toshiba), H. Itoh and M. Oba

May, 1987

©1987, ICOT

**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

Fumihide Itoh\* Kazunori Shimakawa\* Kazuo Togo\* Susumu Matsuda\*

Hidehiko Itoh+ Masahiro Oba+

\*Toshiba Corporation +Institute for New Generation Computer Technology

## ABSTRACT

This paper reports the design, implementation, and evaluation of a relational database engine for variable length records. The engine executes operations for data on disks by special purpose hardware. It has the functions of sorting and relational algebra for variable length records, and on-the-fly processing in which it inputs data directly from a disk. It consists of multiple processors, the nucleus of which is a pipeline two-way merge sorter, and executes processing using key fields extracted from records. The evaluation shows the effectiveness of on-the-fly processing and the key extraction for variable length records.

## 1. 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 and knowledge base machines (KBMs). For KBMs, a relational database machine, Delta, was developed [Shibayama 84] [Kakuta 85] because the relational database is said to be suitable for logic programming. In Delta, facts in Prolog are stored as relations and retrieved by relational algebra. A KBM is now under development, in which knowledge represented in the format of Horn clauses is stored as relations and retrieved by operations called retrieve-by-unification (RBU) [Yokota 86]. In RBU operations, knowledge 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) which perform RBU operations [Morita 86]. Although a Horn clause is represented as a structure which consists of literals and variables, UEs treat the structure as a variable length string. This research is about the relational database engine (RDBE), a special purpose processor which performs relational algebra for variable length data.

There are two types of special purpose processors for high speed database processing. One has on-the-fly processing, which is concurrently reading data from a disk and filtering it. The other has stream processing, which is concurrent execution of an operation and transfer of data. The search processor, SHP [Hayami 86], is an example of the former. Examples of the latter are VLSIs based on the systolic array [Kung 80], the RDBE in Delta [Sakai 84], multiple processors for the join operation [Vanduriez 84], and some processors for sorting, for example, a sort engine by pipeline heap sorting [Tanaka 80] and a pipeline merge sorter [Kitsuregawa 83]. Sorting is a primitive operation for database processing. An algorithm has been proposed for a sorting processor for variable length records [Yang 86], but has not been implemented yet.

With this background, we have developed an RDBE for variable length records with following aims:

(1) Processing variable length records using hardware

The RDBE has hardware which performs sorting and relational algebra (RA) operations for variable length records by using one fixed or variable length key field. By attaching tag data which indicates the end of key fields, the same hardware performs processing using both fixed and variable length key fields.

(2) Efficient database processing

① On-the-fly processing

To process data in disks efficiently, the RDBE has on-the-fly processing in

which it inputs data directly from a disk. Without on-the-fly processing, data in disks is loaded to main memory (MM) before it is input to the RDBE, then the data path in the entire processing is "disk  $\rightarrow$  MM  $\rightarrow$  RDBE  $\rightarrow$  MM". On-the-fly processing makes the data path "disk  $\rightarrow$  RDBE  $\rightarrow$  MM", and achieves more effective processing.

## ② High speed processing in the RDBE

For high speed processing of sorting and RA operations, the RDBE includes multiple processors, which mainly consist of a pipeline two-way merge sorter [Todd 78] and a RA processor, realizing stream processing.

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.

## 2. Basic Ideas

### 2.1 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 disks. Relations manipulated by the database machine have both fixed and variable length attributes. Relation schemata and data are stored in different files in disks. 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. 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). RL means the length of the record. A DF shows whether the record is valid (logically not deleted). A FH consists of field length (FL) and field position (FP). FL means

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 in disks 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 schema (attribute name and data type).

Data processing is shared between the RDBE and the CPU. The hardware of RDBE quickly performs processing that is too heavy for the CPU. The CPU performs processing that is too complex for the hardware or causes a data transfer bottleneck although it is performed quickly by the hardware. The RDBE performs the following data processing:

① Sorting

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

② Duplication elimination

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

③ Selection

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

④ Intersection and difference

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

⑤ Equal join

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

The CPU performs the following data processing:

① Arithmetic and aggregates

② Forming records after join in the RDBE

These two types of processing are too complex for the hardware.

③ External sorting and 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.

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.

① The CPU requests the RDBE to input data from a disk, process it, and output the result to MM.

② The CPU requests the RDBE to input data from MM, process it, and output the result to MM.

③ The CPU directly processes data in MM.

Thus, the RDBE inputs data from both a disk and the CPU.

## 2.2 Features of the Relational Database Engine

In the database machine described in the previous section, the RDBE must have the following functions.

① Variable length records are manipulated.

② Operation processing is executed quickly.

To realize these functions, the following issues are considered:

① Operation processing is executed by the same hardware for both fixed and variable length fields.

② Operation processing is executed quickly by multiple processors in pipeline form.

③ Memories are used effectively in processing variable length records.

From these principles, the RDBE adopts the following methods:

(1) Stream processing by multiple processors, the nucleus of which is a pipeline two-way merge sorter

The sorting and relational algebra processing (RAP) modules are arranged in series. 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. Pipeline processing in these operations is realized by the sorting and RAP modules.

(2) Key field extraction processing for the effective use of memories and the prevention of delay in variable length record sorting

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 memories and the delays, only the key fields flow into the sorting and RAP modules.

(3) Use of tags in processing

In the sorting and RAP modules, tags are attached to data, and processing is executed using tags. One piece 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.

### 3. Overview of Relational Database Engine

#### 3.1 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. The controller interprets a command from the CPU and sets up other modules. The record buffer (RB) module stores all target records. The IN module extracts key fields from target records. The sorting module sorts key values. The RAP module performs RA operations.

Data flows into the RDBE in two-byte units or three-byte units including one byte for the tag. Data formats are as follows.

① The input and output data of the RDBE is the records. Fig. 2 shows the

format of one of these. They are input and output in two-byte units. Input records are sent to the RB and IN modules simultaneously, and output records are read from the RB module.

② Data flowing from the IN module to the RAP module consists of key fields and record identifiers (RIDs). 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 and RIDs. These three bytes flow in parallel. Tags are used for flags and parities. Flags indicate the end of key fields and the duplication of key values.

③ RIDs and their tags flow between the RAP and RB modules in three-byte units. The content of tags is parities.

### 3.2 Controller and Module Functions

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

#### (1) 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.

#### (2) 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.

#### (3) 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.

#### (4) 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.

#### (5) 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.

### 3.3 Processing Method for Sorting and Relational Algebra Operations

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

#### (1) Sorting

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

#### (2) 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.

#### (3) 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.

#### (4) 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 (intersection) or are not (difference) included in the set of first target key values. Fig. 4(a) shows comparison processing. Pointers which point two key values to be compared do not move back because two key value sequences are sorted. Then, the sorting and RAP modules execute intersection and difference as pipeline processing.

#### (5) 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. Fig. 4(b) shows comparison processing. Pointers which point two key values to be compared move back only where both the first and second target key values have same-value duplications because two key value sequences are sorted. Then, the sorting and RAP modules execute equal join as pipeline processing.

## 4. Design and Implementation

### 4.1 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.

### 4.2 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. 5 shows the configuration of a sorting cell. The memory management method is double memory [Yang 86] 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 distinct memory. However, in our implementation, they are stored in two parts of one physical memory because of the hardware size restriction.

#### 4.3 Relational Algebra Processing Module

Fig. 6 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 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.

#### 4.4 Clock Time of Data Flow

In the RDBE, data are transferred in two-byte units 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.

##### (1) RB module and IN module

Alternate input and output at word level is used to realize record level

concurrent input and output to memory in the RB module. Memory write and read are performed sequentially in clocks. Data input in the RB module is synchronized with data input to the IN module. As described later, the sorting module receives data at a speed of one word per three clocks. Data are 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.

## (2) 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 memory accesses occur three times 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. Data is transferred at one word per three clocks.

## (3) RAP module

Essentially, it is possible to transfer data at one word per two clocks because two memories are in physical districts, unlike the sorting module. However, for ease of implementation, data is transferred at one word per three clocks, synchronized with the sorting module.

# 5. Performance Evaluation

## 5.1 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, the length of the key field be  $14$  bytes or more, and  $2\alpha$  bytes. The length of each record is  $2\alpha + 8$ , including four bytes for a record header and four bytes for a

key field header.

#### (1) Sorting

Let the number of target records be a power of 2 and  $n$ . The processing time of sorting consists of input time, delay time, and output time. Each processing time is summarized as follows:

##### ① Input time

Input time is from beginning input of the first record to input of the last key field. The interval of inputting key fields to the sorting module is calculated as follows:

Each record has  $\alpha$  words for a key and four words for a non-key. It takes three clocks per word for a key and two clocks per word for a non-key to transfer them to the sorting module. The minimum multiple of three from  $3\alpha + 8$  up is  $3\alpha + 9$ . From beginning input of the first record to input of the first key field, it takes eight clocks because there are four words for the record header and key field header. Then, input time is

$$(3\alpha + 9)(n - 1) + 8 \text{ clocks.}$$

##### ② Delay time

Delay time is 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 record ID ignoring key (it takes  $3\alpha$  clocks). Then, delay time is

$$3\alpha + 91 \text{ clocks.}$$

##### ③ Output time

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

$$(3\alpha + 3)(n - 1) + 2\alpha + 6 \text{ clocks.}$$

From the above, the total time of sorting is as follows:

$$(6\alpha + 12)n - \alpha + 93 \quad (\text{clocks})$$

The outline of other operations is as follows:

(2) Duplication elimination

Duplication elimination is the same as sorting.

(3) Selection

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

$$(3\alpha + 9)n + 5\alpha + 178 \quad (\text{clocks})$$

(4) 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:

$$(6\alpha + 12)m + (6\alpha + 12)n - 4\alpha + 163 \quad (\text{clocks})$$

Maximum:

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

(5) 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 4096 or less). Let the record number of result data be  $r$  (where  $r \geq n$ ).

Minimum:

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

Maximum:

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

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

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

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

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

- ① A key field is the last part of a record.
- ② The length of a key is  $2\alpha$  bytes, and the length of a record is  $2\beta$  bytes.
- $\beta - \alpha$  is multiple of 3, and  $3\alpha \leq 2\beta - 4$ .
- ③ The number of target records is  $n$  and a power of 2.

The processing time for sorting is as follows:

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

The processing time has the following characteristics:

- ① It is linear to the number of target records.
- ② It is linear to the key length and the record length, and depends more on the record length.
- ③ In selection, the processing time does not depend on the selectivity.
- ④ 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 times of comparison in the RAP module decrease.
- ⑤ 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 computational 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 computational time. Fig. 7(a) shows the time of sorting and selection, and Fig. 7(b) shows that of join. The difference is caused by firmware overhead in the input and output channels.

## 5.2 Influence of Variable Length Data

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, sorting of various length records takes longer 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:

- ① Case 1. Records consist of only variable length key fields, and both key and record lengths vary. The number and total size of records and key fields are constant. The key and record lengths of target records become gradually longer, are constant, or become gradually shorter.
- ② Case 2. Key lengths are constant, and record lengths vary. The number and total size of records and key fields are constant. The record lengths of target records become gradually longer, are constant, or become gradually shorter.
- ③ Case 3. Record lengths are constant, and key lengths vary. The number and total size of records and key fields are constant. The key lengths of target records become gradually longer, are constant, or become gradually shorter.

Fig 8 shows the result of measurement in case 1. The processing time where lengths become longer is the same as where they are constant. However, the processing time where lengths become shorter is longer than the other cases. In cases 2 and 3, the processing time is equal.

Fig. 9 shows the outline of data flow in sorting. To make discussions simple, delay time is omitted, and the key and record lengths of the first half and the second are constant. The figures on the left show distribution of key and record length in target data. The figures on the middle show data flow in the implemented RDBE, in which key extraction is adopted. The figures on the right show data flow where key extraction was not adopted. Fig. 9(a) shows a case where key and record lengths are constant. With key extraction, the



second to last sorting cell starts outputting the sorted first half keys when inputting first half records to the RDBE is finished, and the sorted second half keys when inputting all records to the RDBE is finished. Output of keys takes less time than input of records because the output consists of only keys. When the second to last sorting cell starts output of the sorted second half keys, the last sorting cell starts output of the sorted whole keys, and the RDBE starts outputting the sorted records. Without key extraction, the second to last sorting cell starts outputting the sorted first half records when inputting first half records to the RDBE is finished, and starts outputting the sorted second half records when inputting all records to the RDBE is finished. Outputting records takes the same time as inputting records because both input and output consist of whole records. When the second to last sorting cell starts output of the sorted second half records, the last sorting cell and the RDBE start output of the sorted whole records. In this case, the processing time with and without key extraction is the same.

Fig. 9(b) shows the data flow in case 1. When key and record lengths become shorter, output of the sorted second half keys in the previous cell of the last one must wait until the end of output of the sorted first half keys or records, causing longer processing time than when they are constant or become longer. However, the wait is shorter with key extraction than without it.

Fig. 9(c) shows the data flow in case 2. When record lengths become shorter, output of the sorted second half records in the second to last sorting cell must wait until the end of output of the sorted first half records without key extraction. However, with key extraction, output does not wait because output of the sorted first half keys ends before output of the sorted second half keys begins.

Fig. 9(d) shows the data flow in case 3. In this case, the processing time with and without key extraction is the same.

Key extraction is effective in sorting the variable length records when record lengths in the first half are rather long.

### 5.3 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 in a disk is directly input to the RDBE. Without it, data in a disk is input to the RDBE after it is loaded to MM. Fig. 10 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 disks (disk  $\rightarrow$  MM and disk  $\rightarrow$  RDBE) and that of transferring data between the MM and the RDBE (MM  $\rightarrow$  RDBE and RDBE  $\rightarrow$  MM) are the same.

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

In sorting, the result data is output from the RDBE approximately immediately after the target data is input to it. With on-the-fly processing, there are two data transfers (disk  $\rightarrow$  RDBE and RDBE  $\rightarrow$  MM). Without on-the-fly processing, there are three transfers (disk  $\rightarrow$  MM, MM  $\rightarrow$  RDBE, and RDBE  $\rightarrow$  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  $\rightarrow$  RDBE  $\rightarrow$  MM). Without on-the-fly processing, there are two transfers (disk  $\rightarrow$  MM and MM  $\rightarrow$  RDBE  $\rightarrow$  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.

## 6. Conclusion

The design, implementation, and performance evaluation of the RDBE for variable length records were described. 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 the 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 disks effectively, on-the-fly processing is used so that the RDBE inputs data directly from a 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.

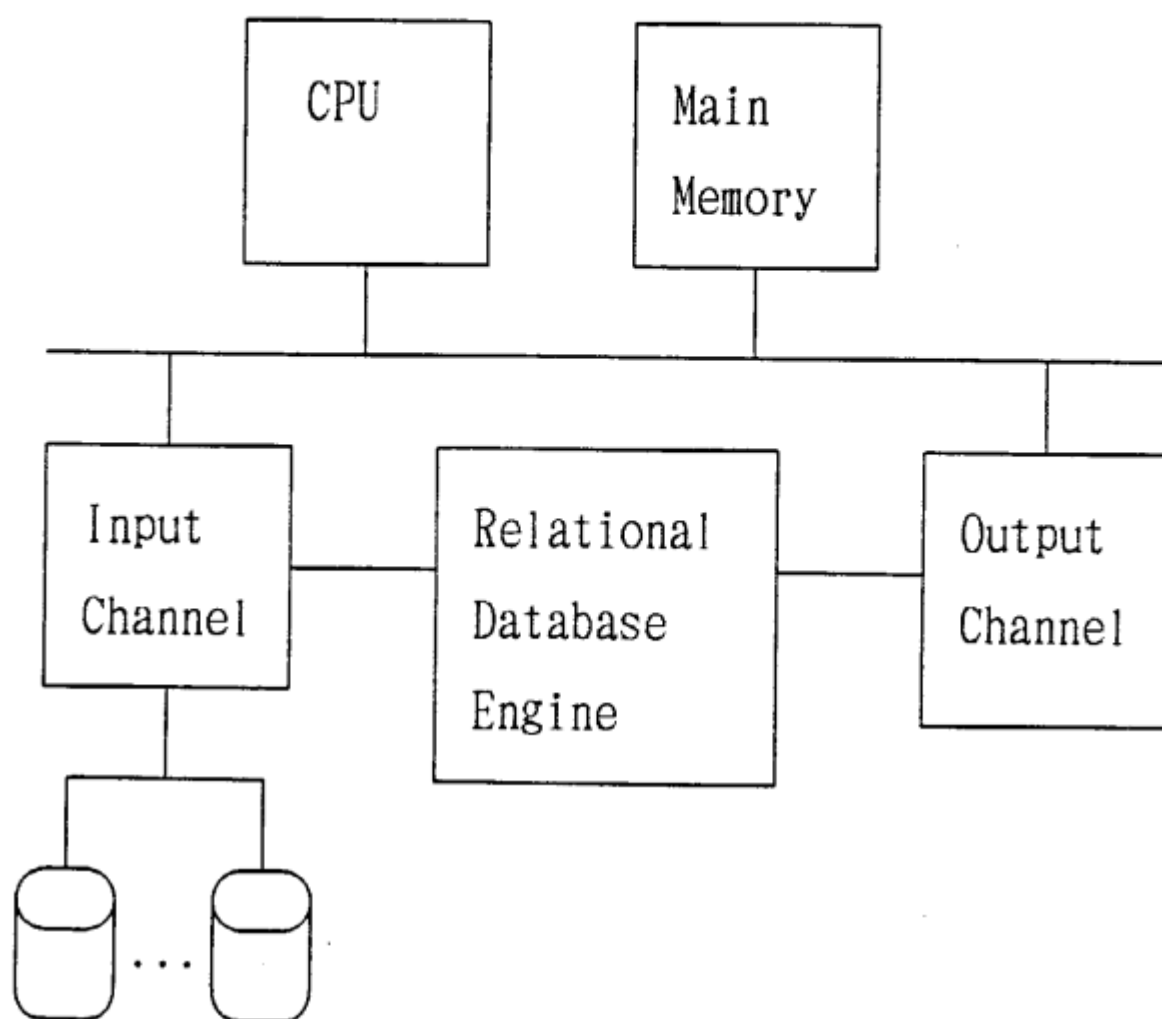
## ACKNOWLEDGEMENTS

The authors 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. The authors also thank the RDBE development members for the implementation and the performance measurement of the RDBE.

## REFERENCES

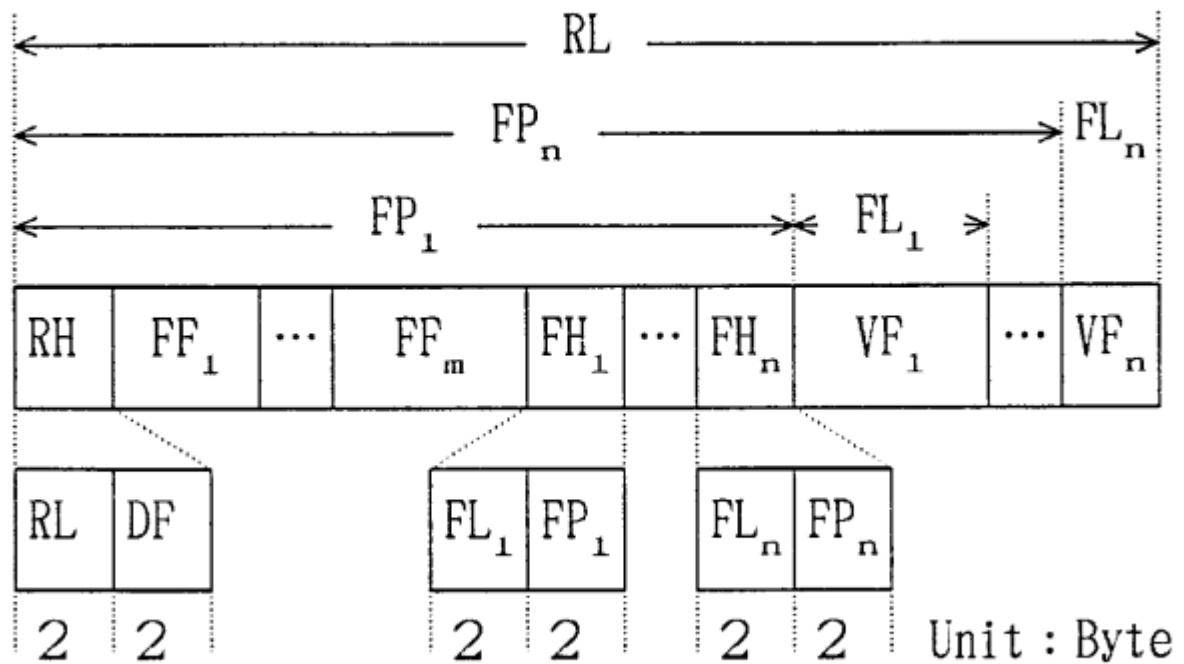
- [Hayami 86] Hayami, H., and Inoue, U., "Design and Implementation of Search Processor", IPS Japan Technical Report, DB-51-2, Jan. 1986
- [Kakuta 85] Kakuta, T., Miyazaki, N., Shibayama, S., Yokota, H., and Murakami, K., "The Design and Implementation of Relational Database Machine Delta", Proc. of the International Workshop on Database Machine '85, Mar. 1985
- [Kitsuregawa 83] Kitsuregawa, M., Fushimi, S., Kuwabara, K., Tanaka, H., and Moto-oka, T., "An Organization of Pipeline Merge Sorter", Trans. IECE Japan (Section J) , Vol.J66-D, No.3 , pp.332-339, Mar. 1983, in Japanese
- [Morita 86] Morita, Y., Yokota, H., Nishida, K., and Itoh, H., "Retrieval-By-

- Unification Operation on a Relational Knowledge Base", Proc. 12th International Conference on Very Large Data Bases, pp.52-59, Aug. 1986
- [Sakai 84] Sakai, H., Iwata, K., Kamiya, S., Abe, M., Tanaka, A., Shibayama, S., and Murakami, K., "Design and Implementation of the Relational Database Engine", Proc. of International Conference on Fifth Generation Computer Systems 1984, Nov. 1984
- [Shibayama 84] Shibayama, S., Kakuta, T., Miyazaki, N., Yokota, H., and Murakami, K., "A Relational Database Machine with Large Semiconductor Disk and Hardware Relational Algebra Processor", New Generation Computing, Vol.2, No.2, Mar. 1984
- [Tanaka 80] Tanaka, Y., et. al., "Pipeline Searching and Sorting Modules as Components of a Database Computer", Proc. of IFIP Congress, pp.427-432, 1980
- [Todd 78] Todd, S., "Algorithm and Hardware for a Merge Sort Using Multiple Processors", IBM J. Res. Dev., 22, 1978
- [Valduriez 84] Valduriez, P., and Gardarin, G., "Join and Semijoin Algorithms for a Multiprocessor Database Machine", ACM Trans. Database Syst., Vol.9, No.1, pp.133-161, 1984
- [Yang 86] Yang, W., Kitsuregawa, M., and Takagi, M., "Pipeline Merge Sorter for Variable Length Record", IPS Japan Technical Report, CA-63-12, Nov. 1986, in Japanese
- [Yokota 86] Yokota, H., and Itoh, H., "A Model and Architecture for a Relational Knowledge Base", Proc. 13th International Symposium on Computer Architecture, pp.2-9, Jun. 1986



Database Data

Fig. 1 Configuration of the Database Machine



RH : Record Header

RL : Record Length

DF : Deleting Flag

FF : Fixed Length Field

FH : Variable Length Field Header

FL : Variable Length Field Length

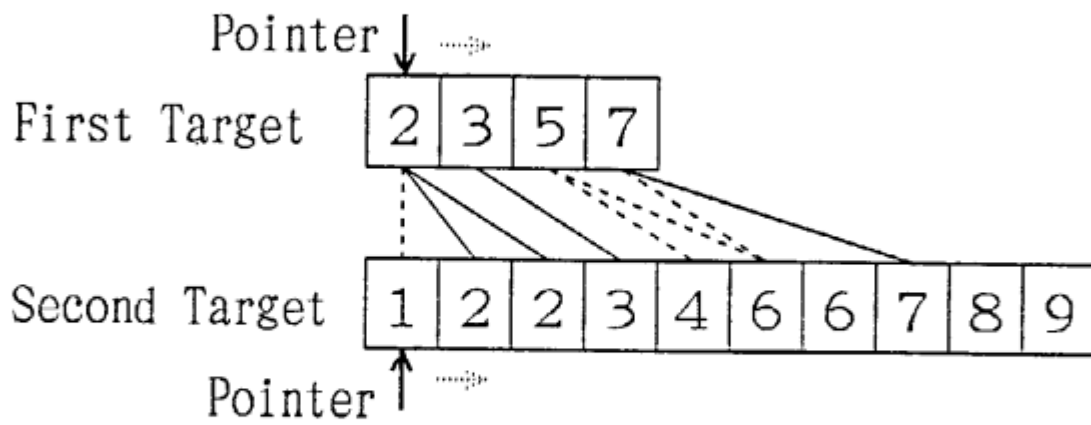
FP : Variable Length Field Position

VF : Variable Length Field Body

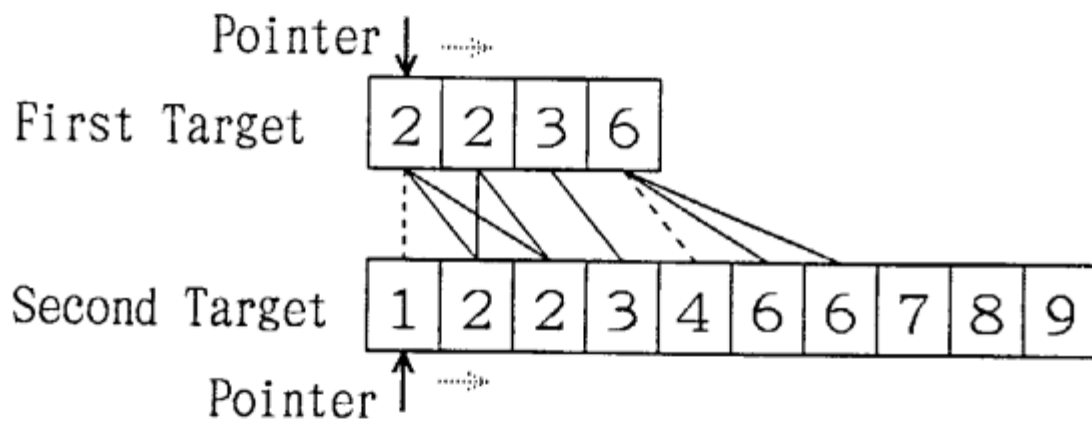
Fig. 2 Format of a Variable Length Record



(a) Intersection and Difference



(b) Equal Join



Comparison — Matched  
 ..... Unmatched

Fig. 4 Stream Processing of RA Operations



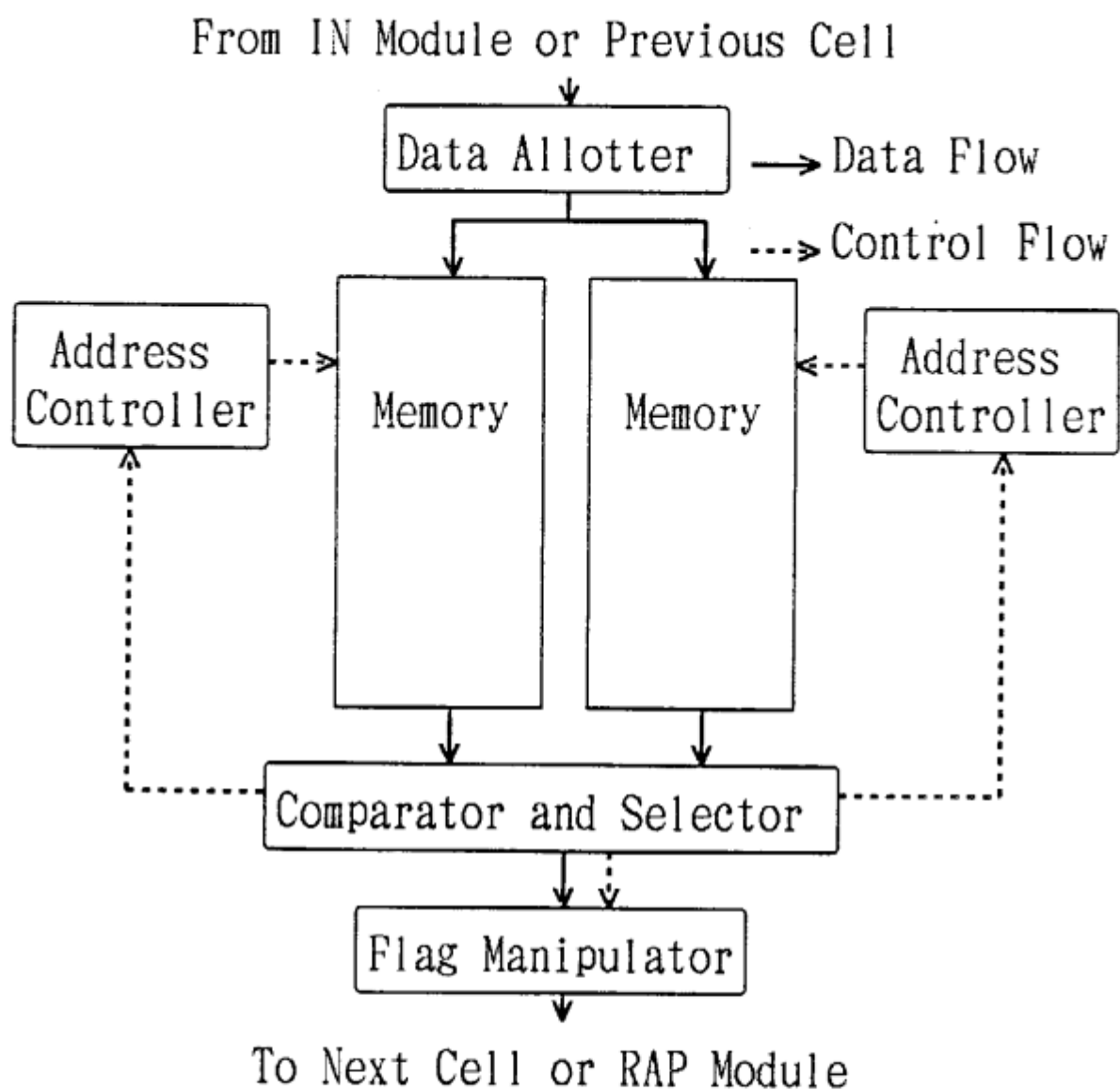


Fig. 5 Configuration of a Sorting Cell

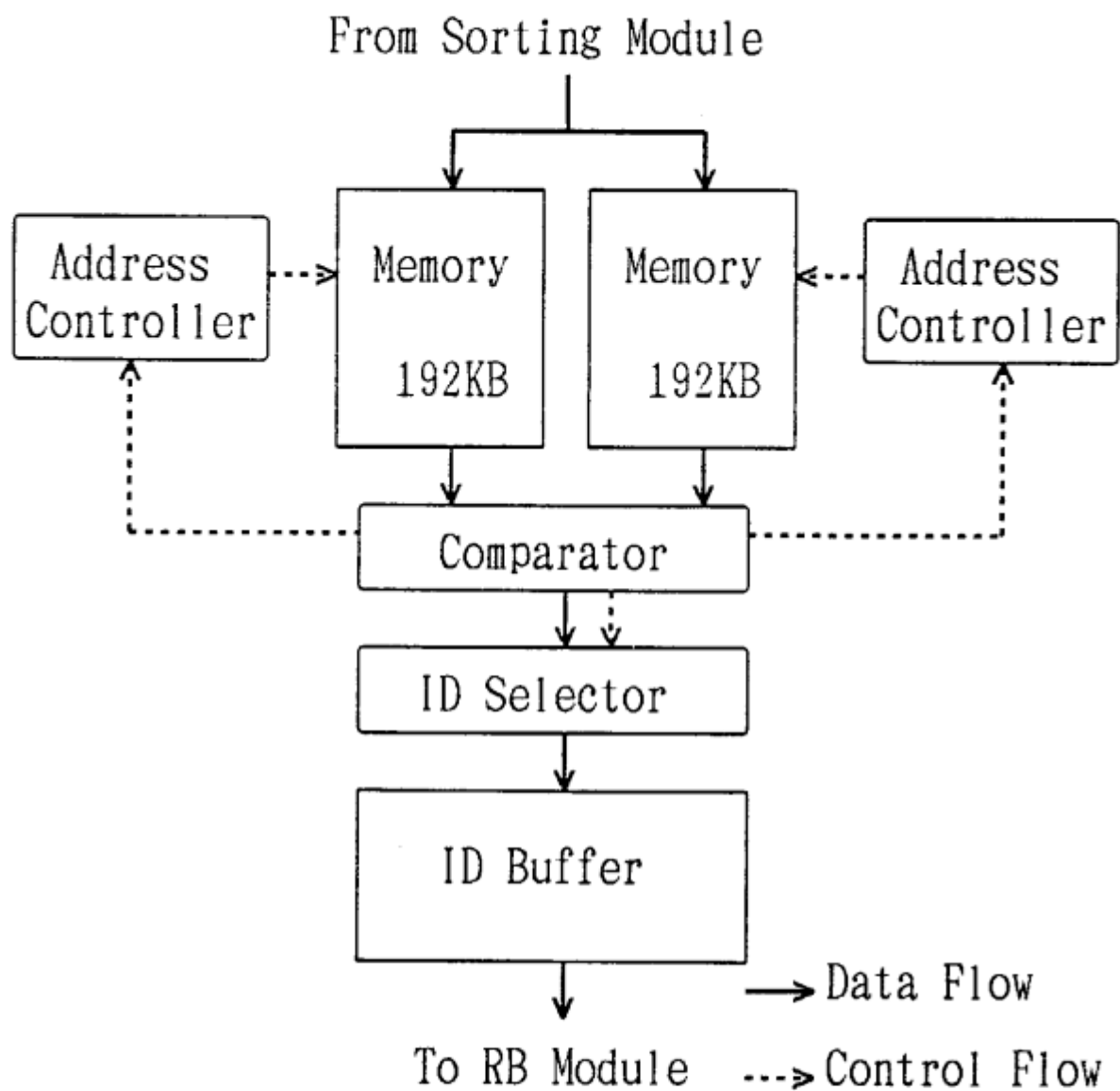


Fig. 6 Configuration of the RAP Module

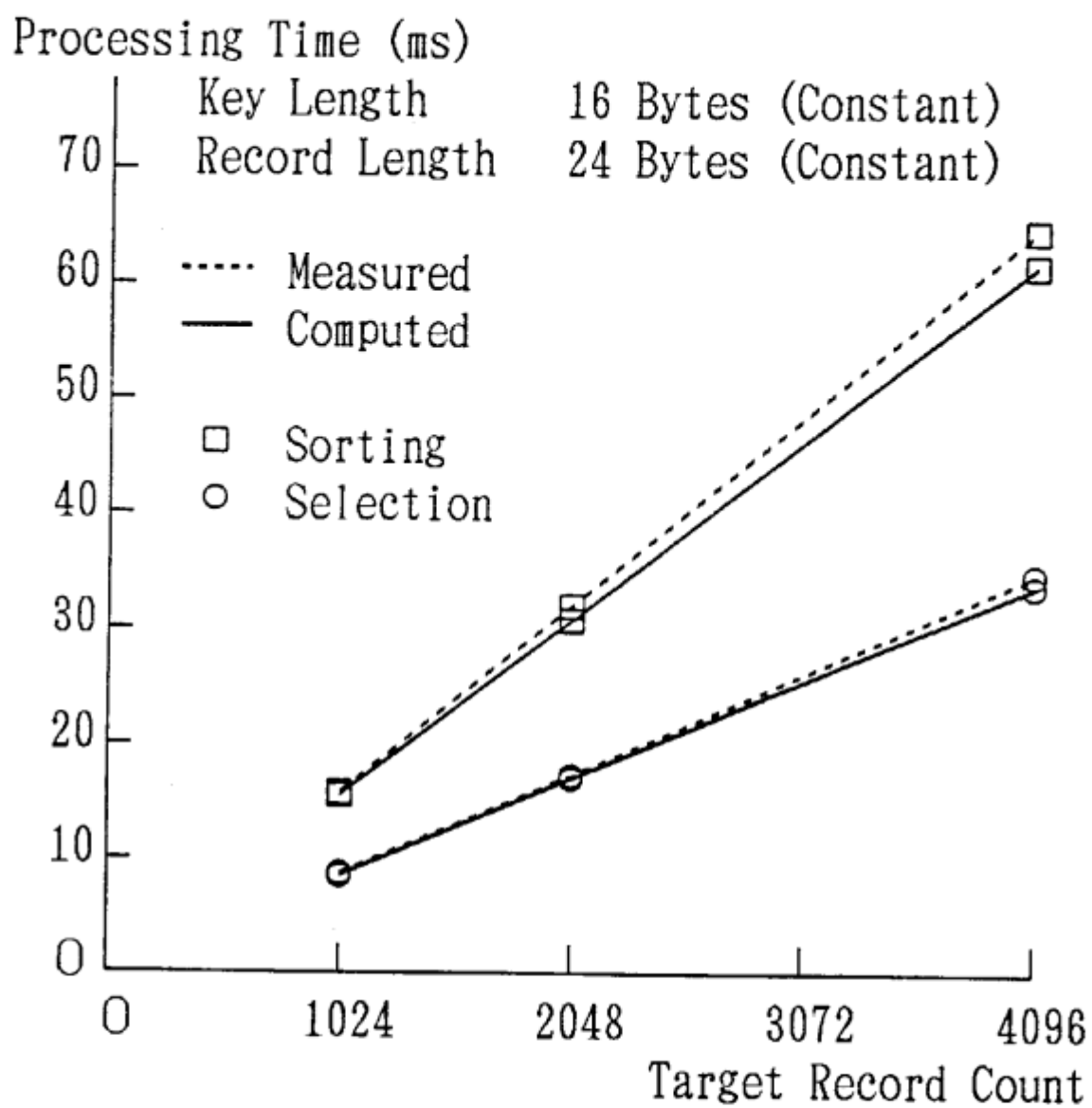


Fig. 7(a) Target Record Count Characteristic  
in Sorting and Selection

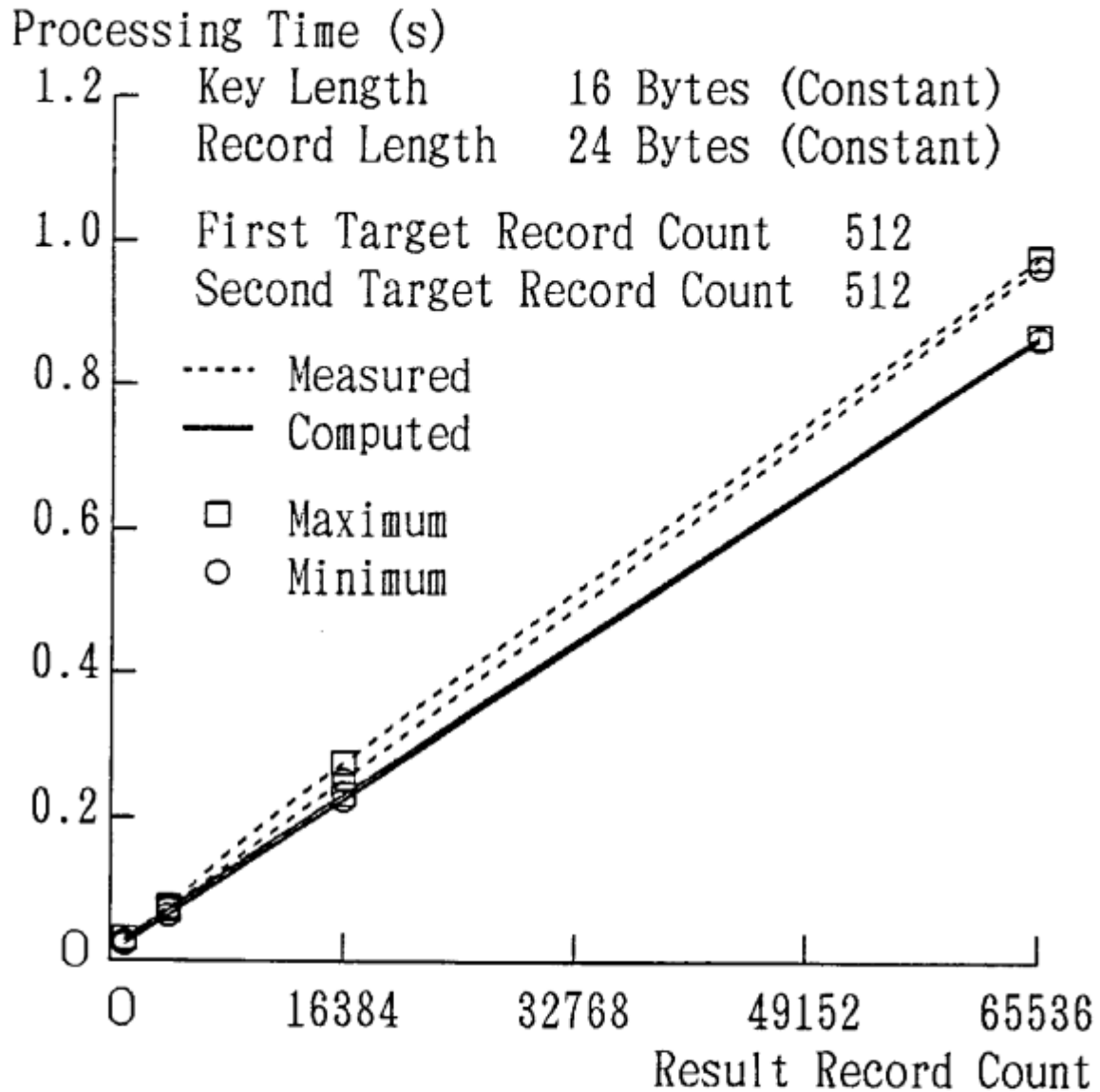


Fig. 7(b) Result record count Characteristic  
in Equal Join

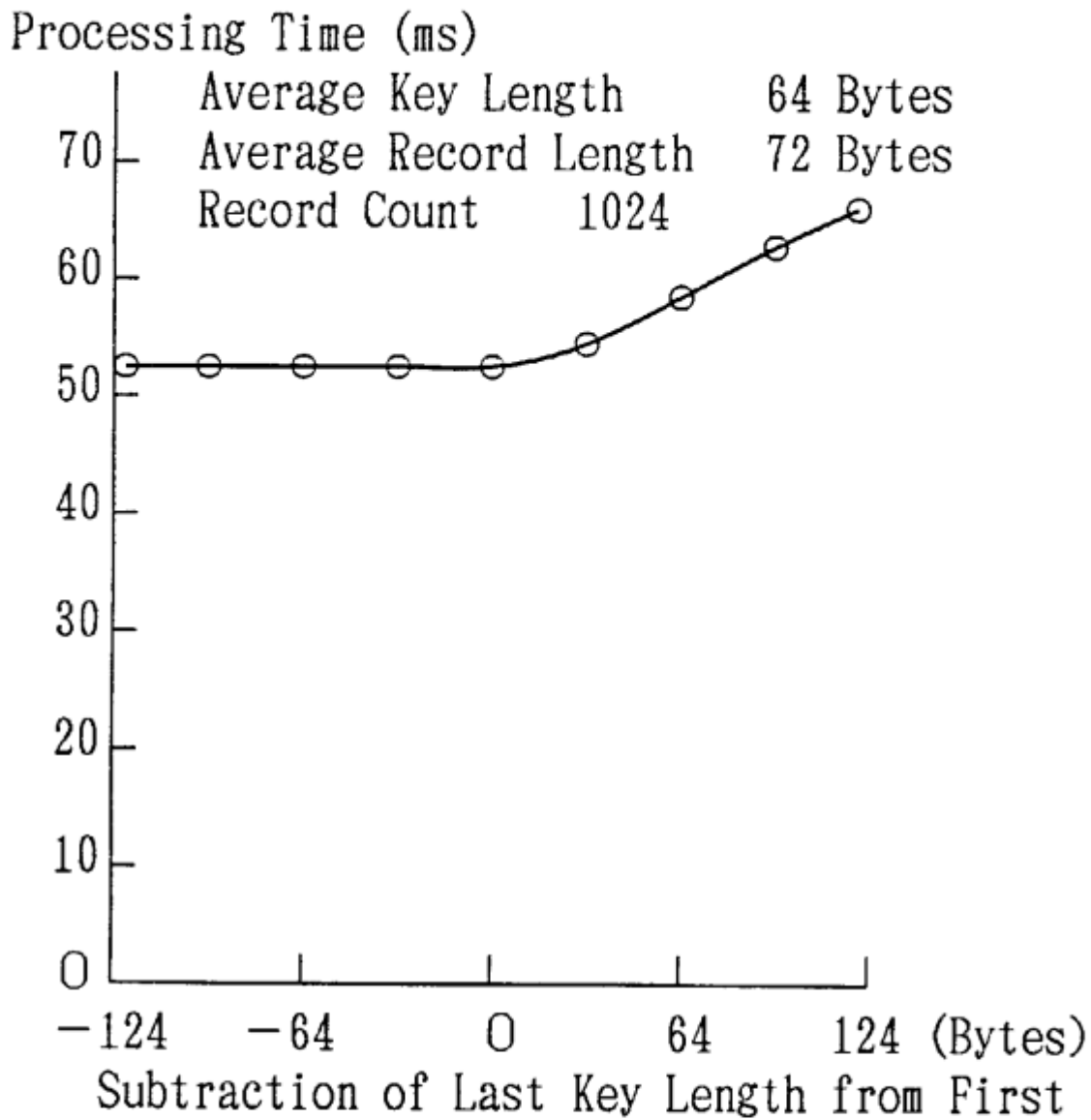






Fig. 8 Target Record Length Distribution  
Characteristic in Sorting

- ① Input to RDBE
- ② Output from Second to Last Sorting Cell
- ③ Output from Last Sorting Cell
- ④ Output from RDBE

|----| Whole Records  
 |----| Only Keys  
 ▲ Middle of Target Data

 Record Headers  
 Key Field Headers  
 Key Fields  
 Other

(a) Case of Constant Key and Record Lengths

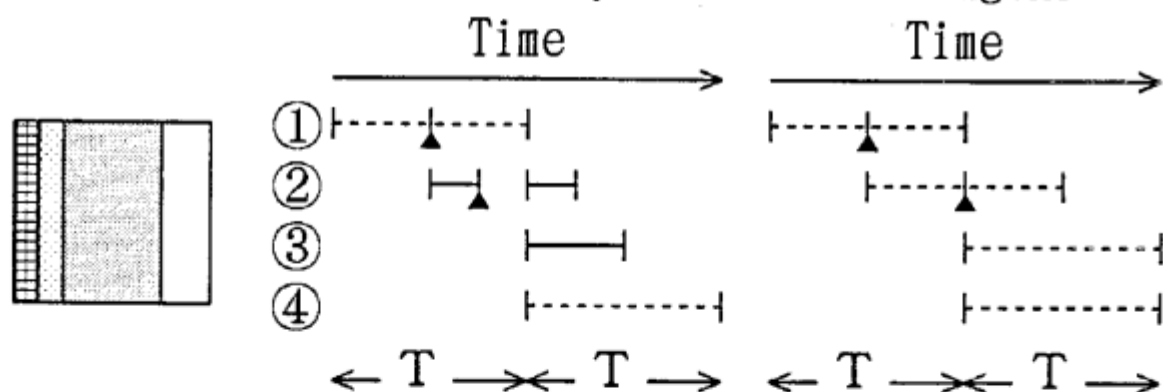


Fig. 9 Outline of Data Flow in Sorting

(b) Case 1

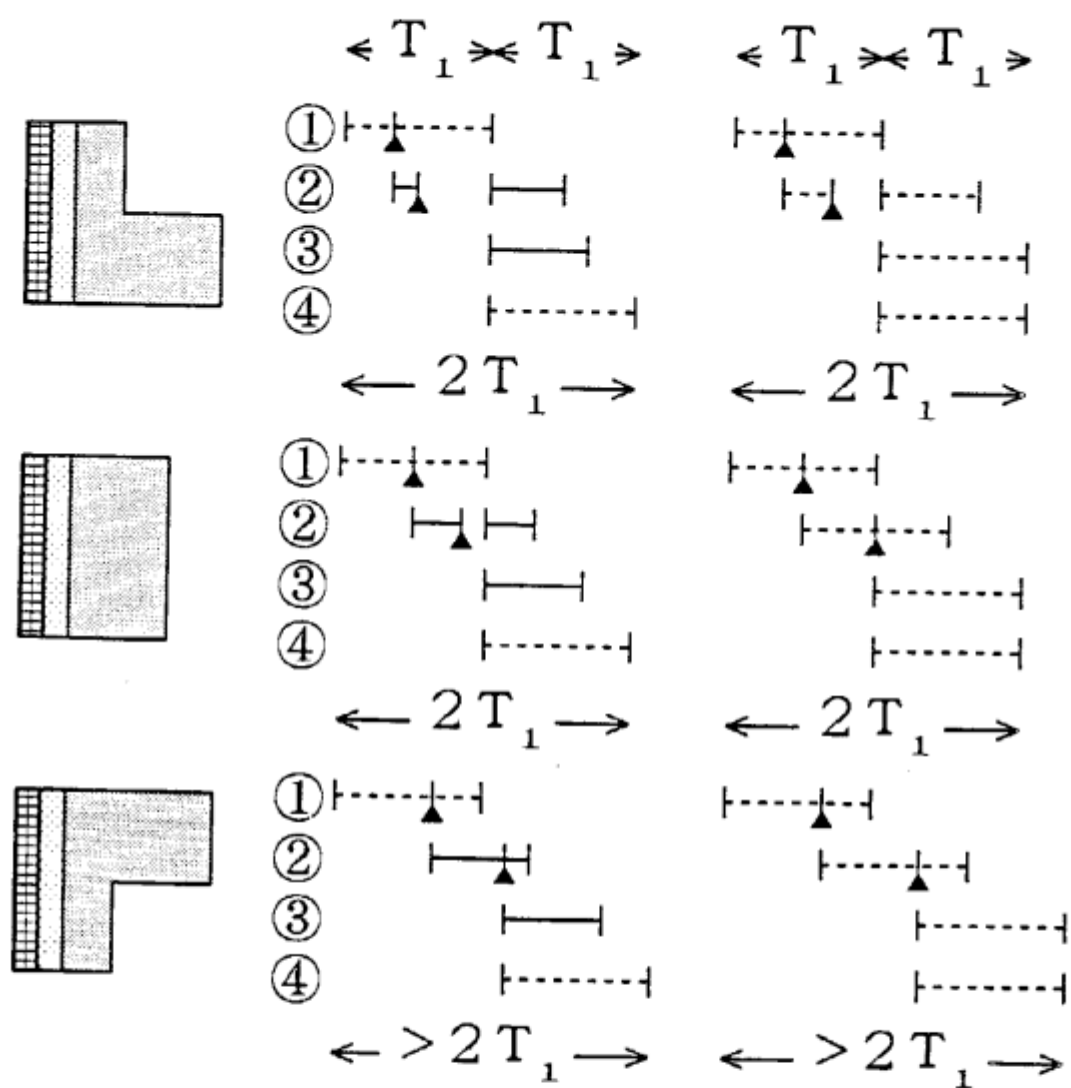


Fig. 9 (Continued)

(c) Case 2

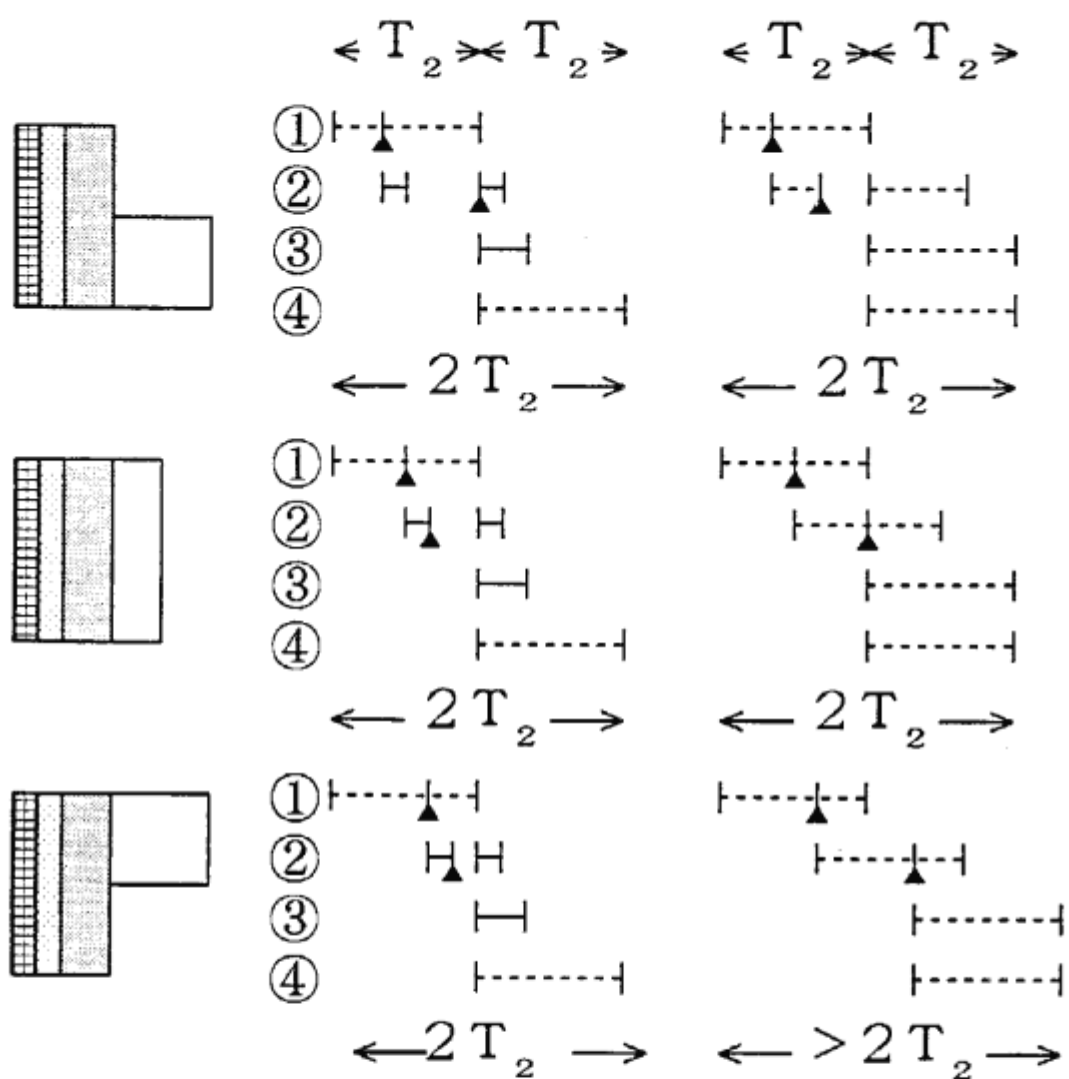


Fig. 9 (Continued)



(d) Case 3

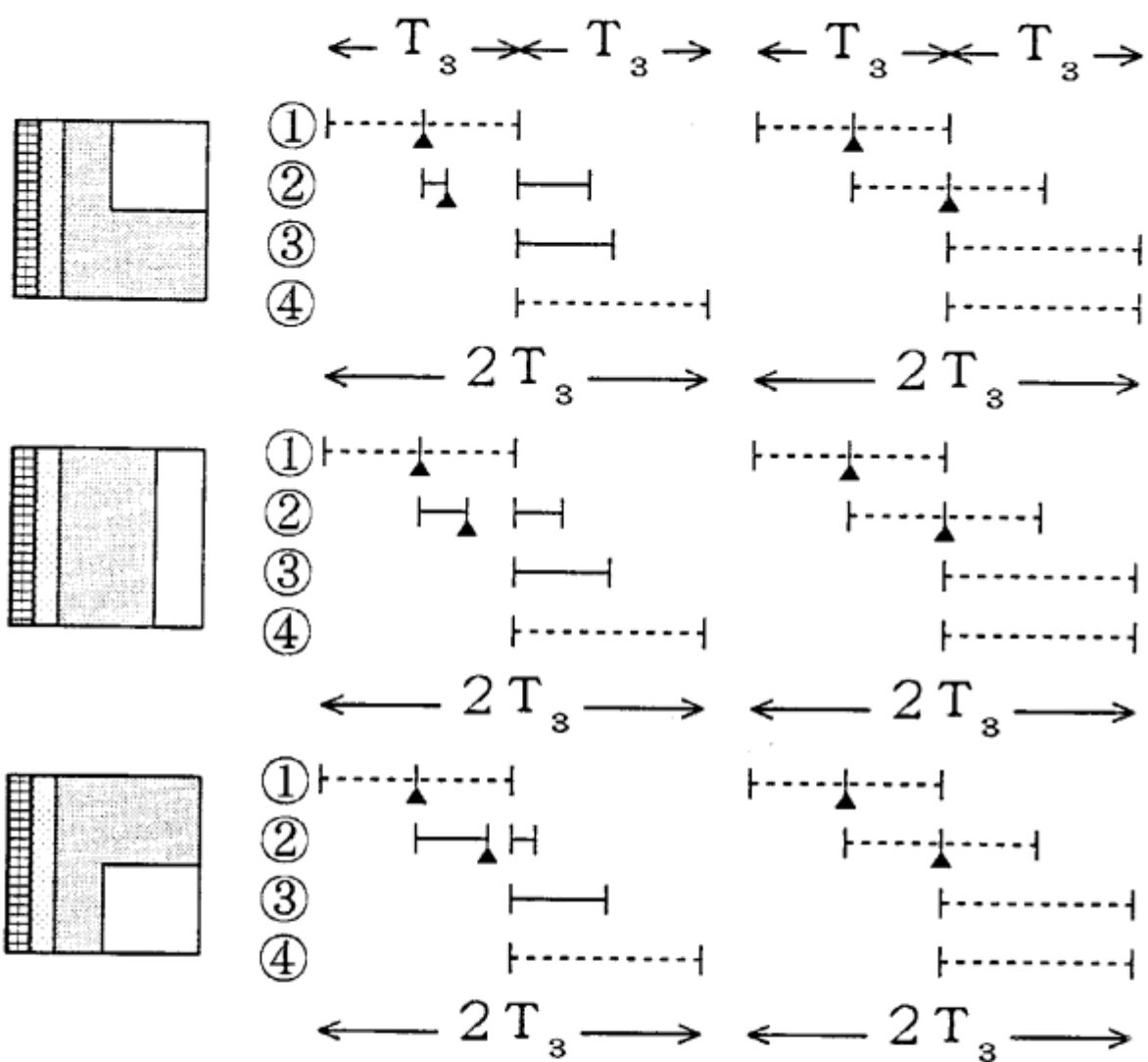


Fig. 9 (Continued)

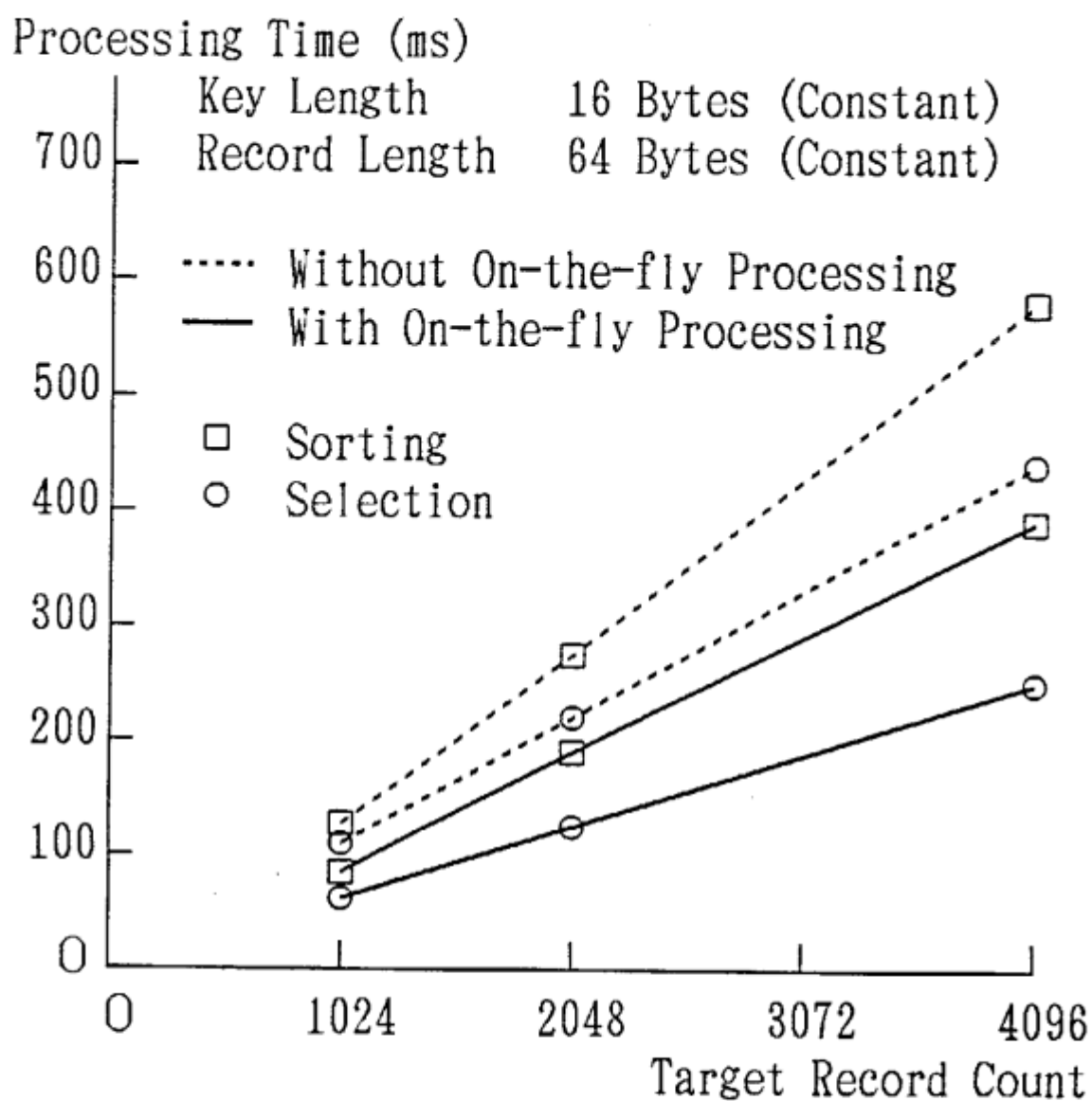


Fig. 10 With and Without On-the-fly Processing  
in Sorting and Selection