Information Engineering Express
International Institute of Applied Informatics
2020, Vol. 6, No. 2, 37 – 52

New Data Structure for Many-to-Many Relations to Reduce Data Size, Recording Time, and Search Time

Tsuneo Kuwabara *

Abstract

With the aim of reducing data recording and search times, this paper proposes a new data structure that reduces data sizes for data in which two item types have many-to-many relations. The proposed data structure newly introduces container variables related to many values of both items, and these container variables record many-to-many relations between them. The proposed data structure maintains data normalization and integrity and is independent of indexing methods conventionally used for relational databases, allowing simultaneous use of both. Simulation results show that data sizes and recording times using the proposed data structure are smaller than those using conventional structures. Search times in the proposed method are much shorter than those under conventional methods when searching data in main memory, such as in the key–value systems often used for data searches in NoSQL, whereas search times under both methods are equivalent when searching data in a relational database. The reduced data size by the proposed data structure might show good performance in other data processing tasks, such as data transmission to other nodes or making copies.
Keywords: Data Structure, Reducing Data Size, Reducing Search Time, Reducing Recording Time

1 Introduction

In today's big data era, the accumulation and utilization of extremely large datasets enables diverse value creation. Reducing data sizes, recording times, and search times is thus increasingly important.

The most well-known method for improving search times is the indexing method used in relational databases (RDBs) [1,2]. The researches to reduce search time are concentrated to how to apply the indexing method, for example [3]. The indexing method, however, cannot reduce the data size and the data recording time, or rather, tends to increase them. The proposed method in this paper is independent on the indexing method.

Recently, some NoSQL databases such as key–value systems use data in main memory to improve data processing speeds [4,5]. The proposed method is not directly concerned with the No SQL, but might be promising for data search on main memory as well No SQL.

* Kanagawa University, Kanagawa, Japan

Graph database such as Neo4j, one of the No-SQLs, are also developed to fasten search time for complexed searches by shorten the search path compared with RDB[4, 5]. But the graph databases are not directly related to the proposed method. The graph databases cannot reduce data size itself dramatically like the proposed method.

This paper proposes a new data structure for reducing the size of data in which two item types have many-to-many relations, thereby reducing data recording and search times [6–11]. The proposed data structure newly introduces container variables related to many values of both items, and these container variables record many-to-many relations between them. This paper newly shows algorithms of extracting the container variables and data recording in detail using flow chart. This paper also newly shows the simulation results of data searching times of the proposed method and conventional methods using randomly arranged data on main memory, and suggest that search time using data in main memory is approximately proportional to the total number of records commonly to proposed and conventional methods.

The new data structure maintains normalization and integrity. It is also independent of the indexing method, allowing simultaneous use of both.

Simulation results show that data sizes and recording times using the proposed data structure are smaller than when using a conventional data structure. Search times under the proposed method are much shorter than those under the conventional method when searching data in main memory, whereas search times under both methods are equivalent when searching data in an RDB.

2 Principle

Figure 1 shows an overview of the proposed data structure. We assume two items, A and B, having different types and a many-to-many relation between them. In Figure 1, value *a* of item A is related to values 1–1,000 of item B, value *b* of item A is related to values 501–1,500 of item B, and value *c* of item A is related to values 1,001–2,000 of item B. In a conventional data structure like those commonly used in an RDB, these relations are recorded as shown in table (1) in Figure 1. In this example, the conventional data structure would produce 3,000 records.

The proposed method instead records these relations as shown in tables (2)–(5) in Figure 1. Here, item B values in table (3) and item A values in table (4) are related via newly introduced container variables. These container variables relate multiple values of items A and B. For example, values *a* and *b* of item A are related to values 501–1,000 of item B via container variable *x*, whereas values *b* and *c* of item A are related to values 1001–1500 of item B via container variable *y*. Values of items A and B, which cannot be related via these container variables, are recorded in tables (2) and (5). For example, value *a* of item A is related to values 1–500 of item B in table (2), whereas value *c* of item A is related to values 1,501–2,000 of item B in table (5). Using the proposed data structure, the number of records is reduced from 3,000 in the conventional data structure to 2,004 among tables (2)–(5). Tables (2) and (5) are actually combined into one table in the database.

Generally, the reduction in record count, D, when using the proposed data structure in place of the conventional structure is

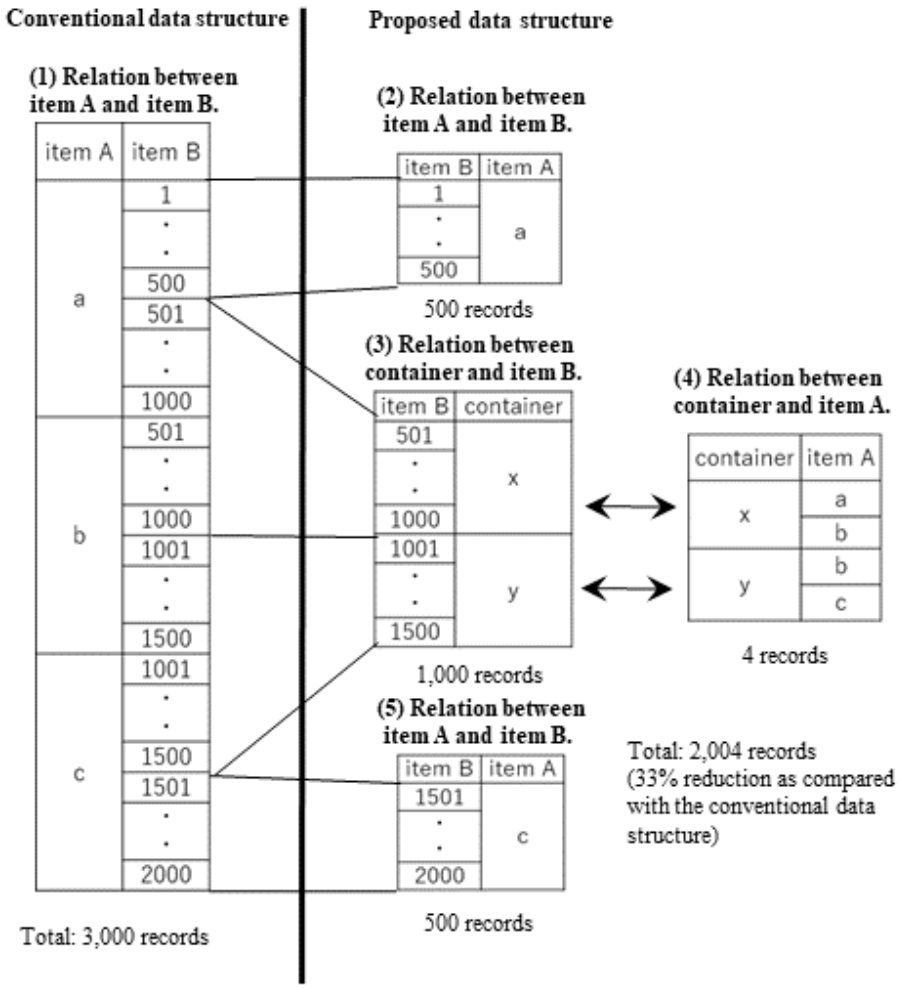

Figure 1: Overview of the proposed data structure.

$$D = \sum_{i=1}^{n} \{M_i \cdot N_i - (M_i + N_i)\}, \qquad (1)$$

where i is the container number, n is the number of containers, M_i is the number of item A values related to container i, and N_i is the number of item B values related to container i.

Data in a conventional data structure and those in the proposed structure can be exchanged without loss or redundancy. The proposed structure thus can maintain data integrity and normalization.

3 Typical Application

This section describes a purchase records database as a representative example, comparing the proposed data structure with the conventional structure including many-to-many relations.

Figure 2 shows the conventional data structure commonly used in an RDB. There are three tables:

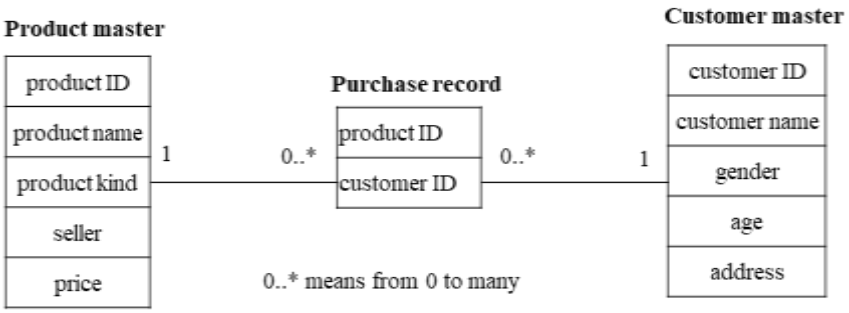

Figure 2: Conventional data structure including many-to-many relations.

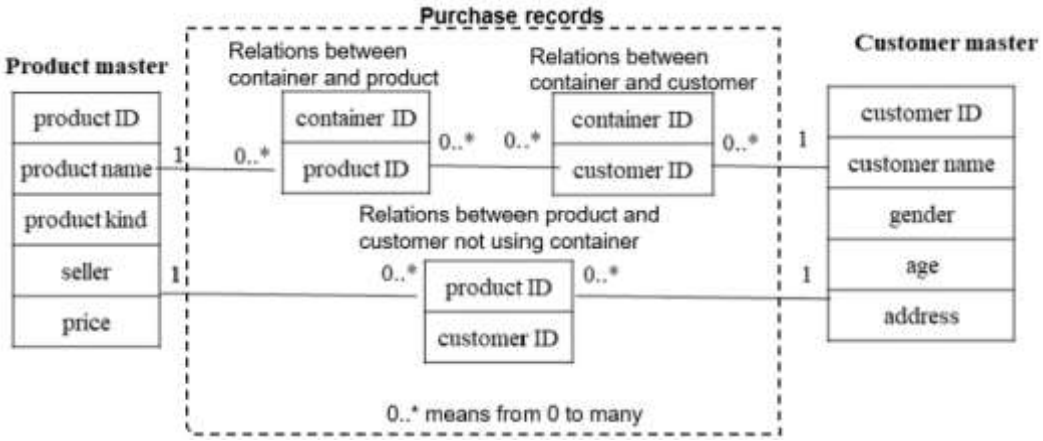

Figure 3: Proposed data structure including many-to-many relations.

"product master," "customer master," and "purchase records." The purchase records table will normally have the most records. For example, if the product master table has 1,000 records and the customer master table has 100,000 records, the purchase records table may have up to 100,000,000 records.

Figure 3 shows the proposed data structure. The product master table and the customer master table in Figure 3 are the same as those in the conventional data structure in Figure 2. The purchase record table is replaced by three tables: one for relations between containers and products, one for relations between containers and customers, and one for relations between products and customers not related to containers. These respectively correspond to table (4), table (3), and the combined tables (2) and (5) in Figure 1.

As this example shows, the proposed data structure can be partially applied to data having many-to-many relations, such as a relation between product and customer, which presumably has huge

amounts of data.

4 Algorithm

4.1 Extracting Container Variables

The first step in applying the proposed data structure is to extract container variables relating many values of two items. These two items correspond to items A and B in the example shown

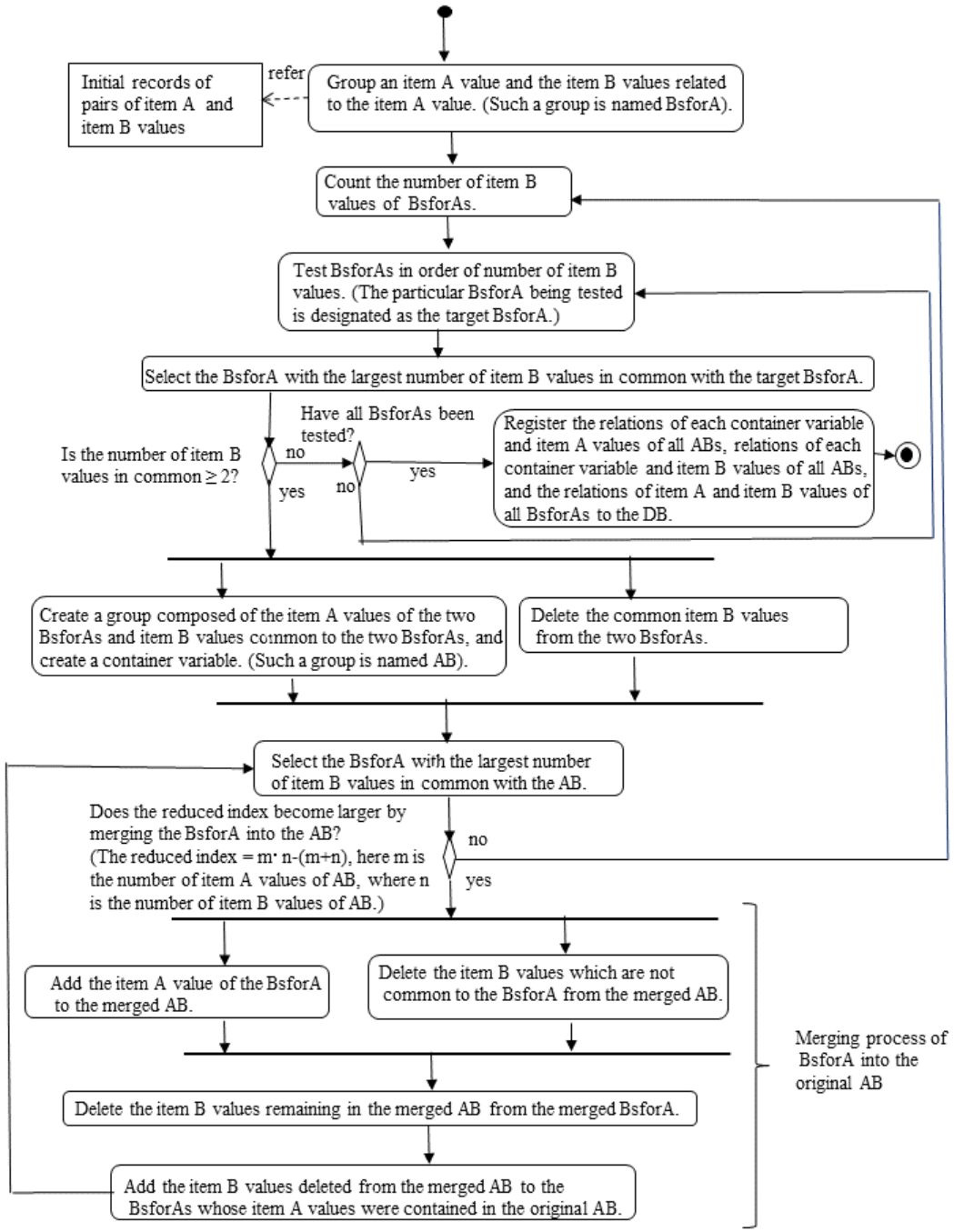

Figure 4: Algorithm for extracting container variables in the proposed method.

in Figure 1, or to the product and customer IDs in the example in Figures 2 and 3.

Figure 4 shows an algorithm for extracting container variables. Relations between container variables and item A values, those between container variables and item B values, and those between item A and item B values not recorded via container variables are stored in a database (DB).
In this algorithm, all pairs of item A and item B values in the initial data are divided into groups having an item A value and item B values related to the item A value. This group is named BsforA in Figure 4.

All the combinations of any 2 BsforAs are tested whether the container variables are extracted or not. The number of the tests is k(k-1)/2, here, k is the number of BsforAs, that is the number of kinds of item A values. If there exist some container variables which should be extracted, they certainly extracted with the algorithm shown in Figure 4 because a container variable must be related to more than 2 item B values common for two BsforAs at least. When the container variable are extracted, the item A values of the 2 BsforAs and the common itemB values are merged one group named AB with the container variable.

When a container variable is extracted from the two BsforAs, all other BsforAs are tested whether they can be merged to the AB to increase the reduction in record count, *D,* in Equation (1). The BsforAs are tested one by one in order of the number of item B values of the BsforAs common to the item B values of the AB. This procedure is reasonable because the more the number of the common item B values is, the bigger the D becomes. However, it is possible that the above procedure may miss merging chance when merging one BsforA to the AB cannot increase the D and merging multiple BsforAs to the AB at the same time can increase the D. In that case, the D does not become maximum with this algorithm. But even in that case, the container variables extracted with this algorithm certainly reduce the total data size.

Figure 5 shows an example conversion using this algorithm from the conventional data structure to the proposed data structure. In this example, six container variables are extracted, and the data size is reduced to 58% of that of the original conventional structure.

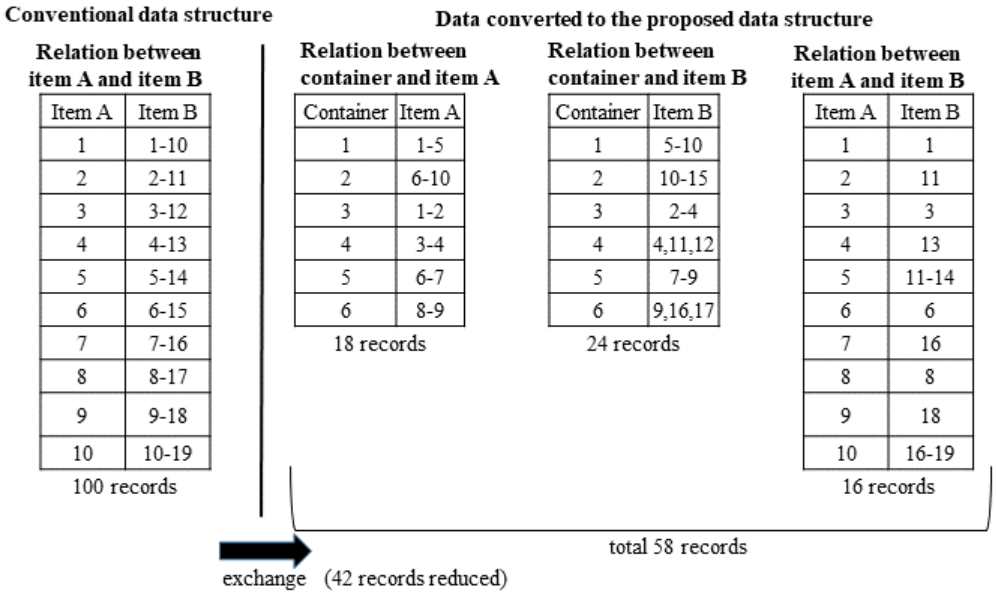

Figure 5: Extracting container variables using the proposed algorithm.

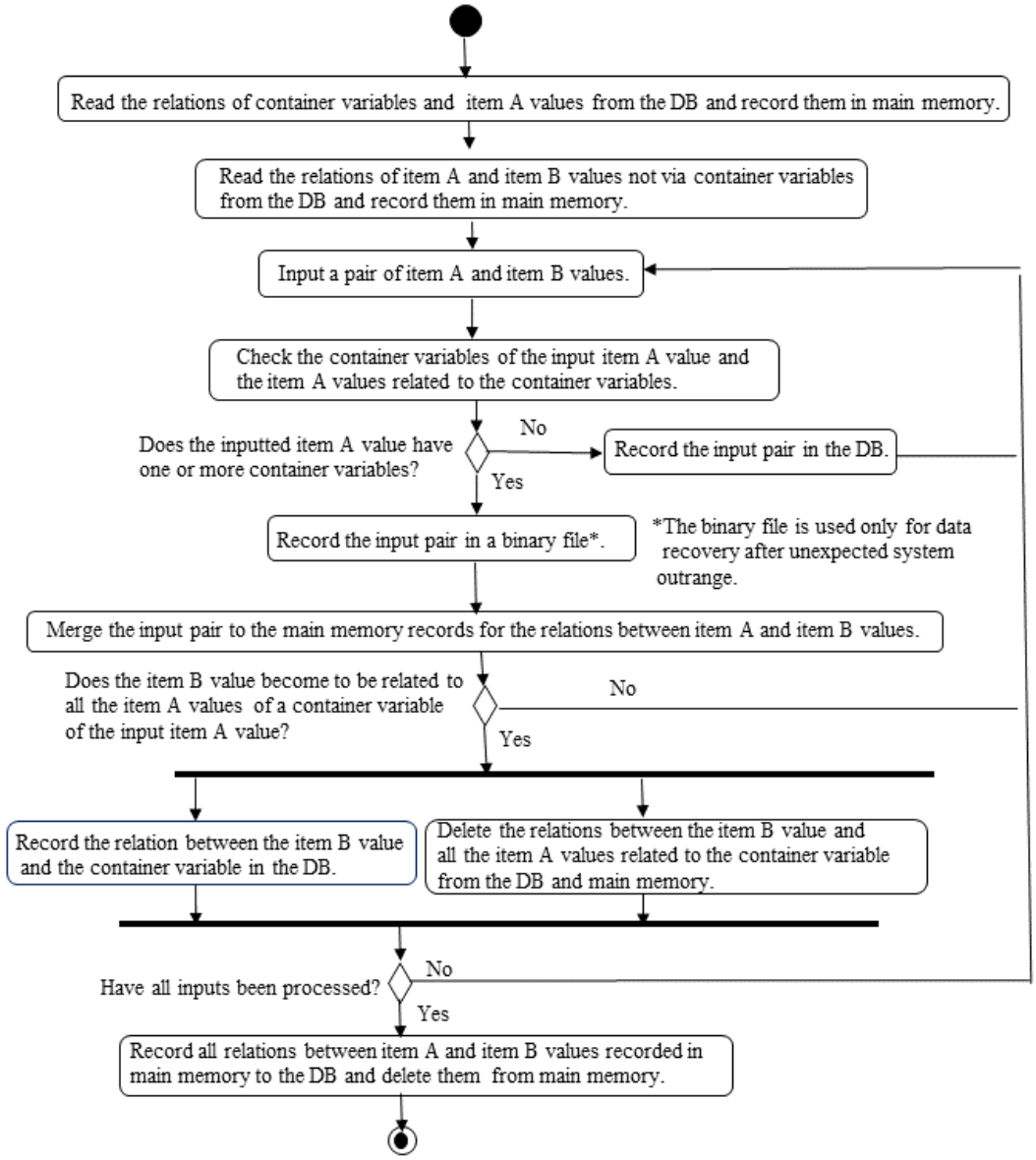

Figure 6: Algorithm for data recording in the proposed method.

This algorithm can be performed at early stages of data collection because it is difficult to dynamically modify the set of container variables over the entire period. If it is necessary to add or modify container variables, the algorithm can be periodically applied at appropriate intervals.

4.2 Data Recording

After extracting container variables using the algorithm described in section 4.1, input pairs of item A and item B values are recorded directly or using container variables.

Figure 6 shows an algorithm for recording data. If a newly input item A value is unrelated to any container variable, the input pair of item A and item B values is directly recorded in the database.

If instead a newly input item A value is related to a container variable, the input pair of item A and item B values is not immediately recorded in the database, but instead retained in main memory. When an input item B value becomes related to all item A values related to a container variable united with previously input data, only the relation between the container variable and the item B value is recorded in the database. Then, all input pairs of item A and item B values related to the container variable are deleted from the database and main memory. The number of database accesses is thus dramatically reduced, as is the data recording time. Simultaneously, all input data are recorded in a binary file, used only to recover data when the system is unexpectedly stopped.

4.3 Data Searching

The following assumes the data structure of tables (2)–(5) in Figure 1 to describe an algorithm for searching for item B values related to an input item A value.

1. Search the combined tables (2) and (5) for item B values related to the input item A value.
2. Search table (4) for container variables related to the input item A value.
3. If there is no such container value, then end.
4. If there exist such container variables, search table (3) for item B values related to the container variables, then end.

For an algorithm that instead searches for item A values related to an input item B value, exchange items A and B and tables (3) and (4) in the above steps.

5 Simulations

5.1 Using Regularly Arranged Data

Using regularly arranged sample data, the following compares data recording and search times when using the conventional data structure and when using the proposed data structure and algorithms.

Database table (5) in Figure 7 shows the data used in the simulation applying the conventional data structure. This simulation produced 10,000,000 pairs of item A and item B values, and tables (6) and (7) show the data used when applying the proposed data structure. In this case, there also exists a table for relations between item A and item B not related to container variables. However, no data are recorded in that table in this simulation.

Simulations were performed using a Java application, using JDBC to access the RDB where the data are stored. All RDB tables for both the conventional and proposed data structures are independently indexed for every column of each table.

The following tools were used in the simulation.

(1) Hardware
Fujitsu TX1320 M4
3.5 GHz Xeon E-2134 processer, 16 GHz main memory,
two 500 GHz, 7,200 rpm hard disk drives

(2) Software

OS: CentOS 7.6 1810
Java: Open JDK 1.8.0_211
DBMS: MySQL 5.1.72
JDBC: mysql-connecter-java-5.1.47

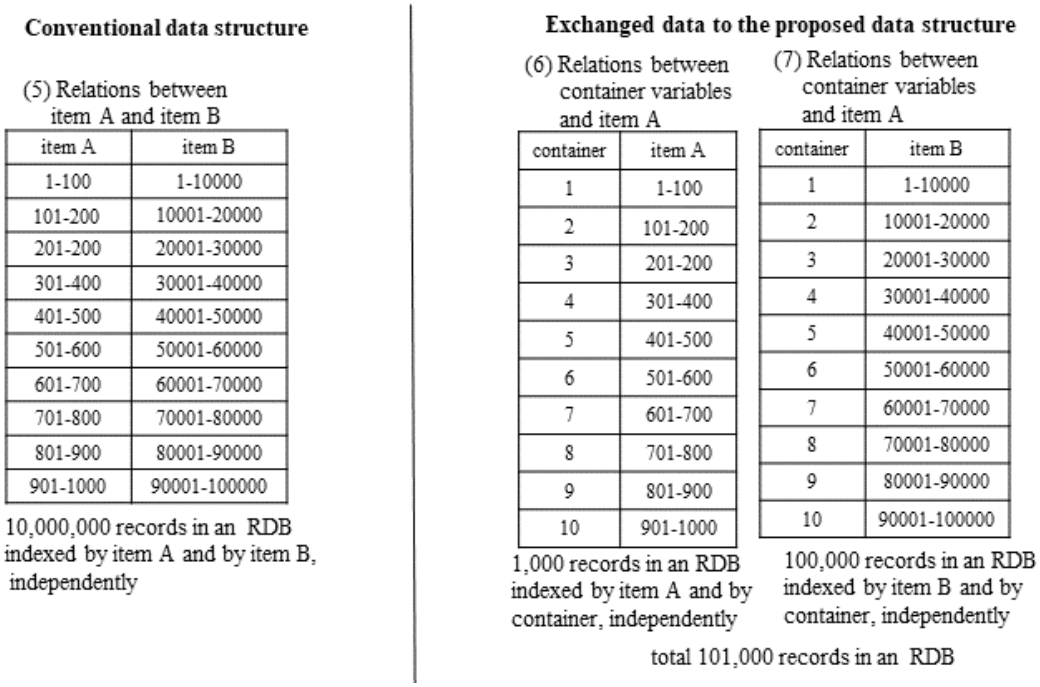

Figure 7: Regularly arranged data used for the simulation.

Table 1: Results of simulations with regularly arranged data.

			Conventional data structure	Proposed data structure	Comparison with conventional
Data size (number of item A and item B pairs)			10,000,000	101,000	Excellent
Data recording time (s/10,000,000 records) (Here, A record corresponds to a pair of an item A and an item B.)			395	112	Good
Data searching time (s/10,000,000 outputs)	In RDB	Search for item B from input item A	11.0	10.6	Equivalent
		Search for item A from input item B	17.2	23.6	
	In main memory	Search for item B from input item A	11.4	0.15	Excellent
		Search for item A from input item B	13.2	0.21	

When producing the data for table (7) in Figure 7 under the proposed data structure, the relations between container variables and item A values shown in table (6) in Figure 7 are assumed to be already extracted. The relations between the container variables and item A values in table (6) can be extracted using the algorithm described in section 4.1, taking about 21 s with 1,000,000 pairs of item A and item B values, or about 3.6 s with 100,000 pairs.

After extracting the container variables, 10,000,000 pairs of item A and item B values were input and processed using the algorithm described in section 4.2. As a result, relations between container variables and item B values in table (7) in Figure 7 are recorded in the RDB.

In contrast, under the conventional data structure, these 10,000,000 pairs of item A and item B values are simply recorded in the RDB.

Data searching simulations were performed using two methods:

(1) Each data is directly searched for in the RDB.
(2) Each data is searched for in main memory.

In case (2), all data in each RDB table are set once to HashMap before searching. The HashMap is one of the Java APIs. Searches are performed on one or more HashMap instances under both the conventional and proposed data structures. In the simulations, data setting times from the RDB to HashMap are not added to the search times.

Table 1 shows the results of the simulations. Note that the results for data recording times under the proposed data structure include times for recording data to a binary file, as described in section 4.2.

As Table 1 shows, the results for the proposed data structure are much better than those for the conventional data structure in terms of data size and search times when using data in main memory, better in terms of data recording time, and equivalent in terms of data search times using data in an RDB. Data size reduction under the proposed data structure seems to be very effective when data in main memory are searched.

Then, search times using data in main memory are also simulated for two data amounts, 1,000,000 and 100,000 pairs, with the same relations of container variables and item A values but fewer relations of container variables and item B values.

Figure 8 shows the results, including the results for the 10,000,000 pairs in Table 1. As Figure 8 shows, search times under the proposed data structure and algorithm are far reduced as compared to those under the conventional method for all data amounts. Data searches in main memory are commonly used under NoSQL, such as key–value systems. Applications of the proposed data structure to NoSQL may be promising, but concrete development of such applications is left to future work.

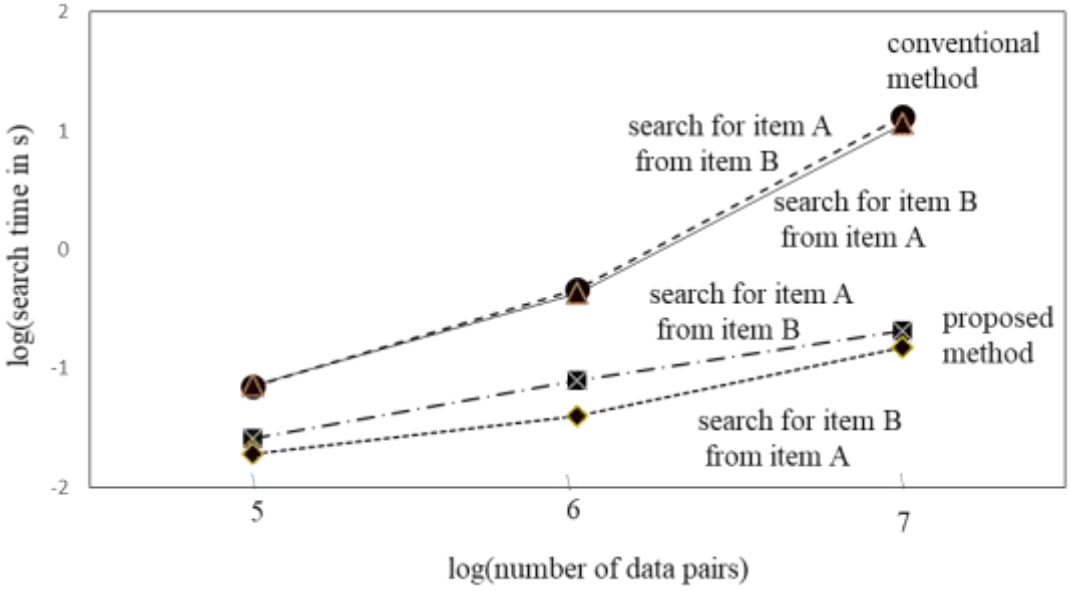

Figure 8: Search times using regularly arranged data in main memory.

5.2 Using Random Data

This section describes simulations with randomly arranged item A and item B pairs.
Item A is randomly selected from 1 to 1000, while item B is randomly selected from 1 to 4000. These pairs are selected so as to exclude pairs where item A and item B values are the same. The simulations are performed for cases of 3,000,000 pairs (Case 1), 2,000,000 pairs (Case 2), and 1,000,000 pairs (Case 3).

The selected data pairs are once recorded in an RDB table, which can be directly used for the conventional data structure. Relations between container variables and item A values, those between container variables and item B values, and those between item A and item B values not via container variables are obtained from the table using the algorithms described section 4.1. In these simulations, there also exist data regarding relations between item A and item B values not related to container variables but directly connected. Under the proposed method, these three relations are recorded in the three RDB tables.

Table 2 shows numbers of records of these three relations in each case under the proposed method, along with total numbers of records in each case and reduction rates as compared with the conventional method. The reduction rate R is

$$R = 1 - \frac{P}{C}, \qquad (2)$$

where P is the number of total records under the proposed method, and C is the number of records under the conventional method.

As Table 2 shows, the reduction rate *R* when using the proposed method increases with the number of item A and item B value pairs.

In actual situations, data will likely be biased rather than random. For example, many users access popular sites, while few access unpopular sites. The *R* in Eq. (2) can thus become bigger, because some container variables may be related to many item A and item B values.

We next measure search times in each case under both the proposed and conventional methods using data in main memory. The procedures for measuring search times is as described in section 5.1.

As Figure 9 shows, search times under the proposed method are smaller than those under the conventional method in every case. These results show that the proposed method is efficient even when using randomly arranged data in main memory.

Figure 10 shows search times as a function of total number of records for both the proposed and conventional methods. These results show that under both methods, search times using data in main memory is approximately proportional to the total number of records in the search, suggesting that these search times are simply dependent on the total data size. As the proposed method can reduce the total number of records as compared with the conventional method, search times using data in main memory become shorter.

Table 2: Number of records in simulations with random data.

Number of item A and item B pairs*	Number of records under the proposed method				Reduction rate *R* by Eq. (2)
	Container and item A	Container and item B	Item A and item B not via container	Total	
1,000,000	31,260	483,422	33,111	547,793	0.45
2,000,000	32,655	891,126	33,294	957,075	0.52
3,000,000	33,547	961,134	33,150	1,027,831	0.66

*Equal to the number of records under the conventional method.

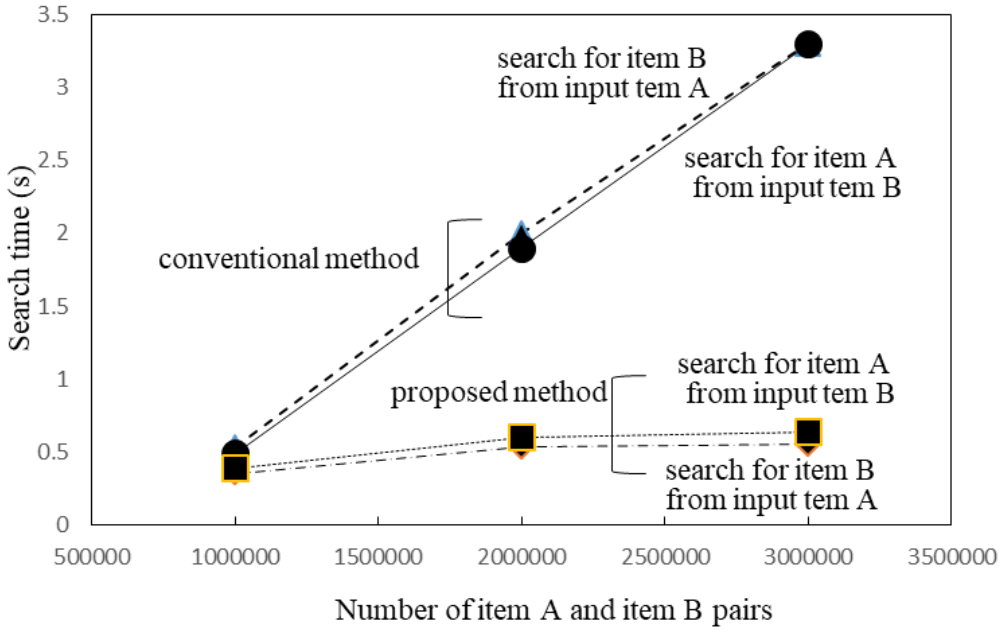

Figure 9: Search times as a function of the number of item A and item B value pairs using data in main memory with random data.

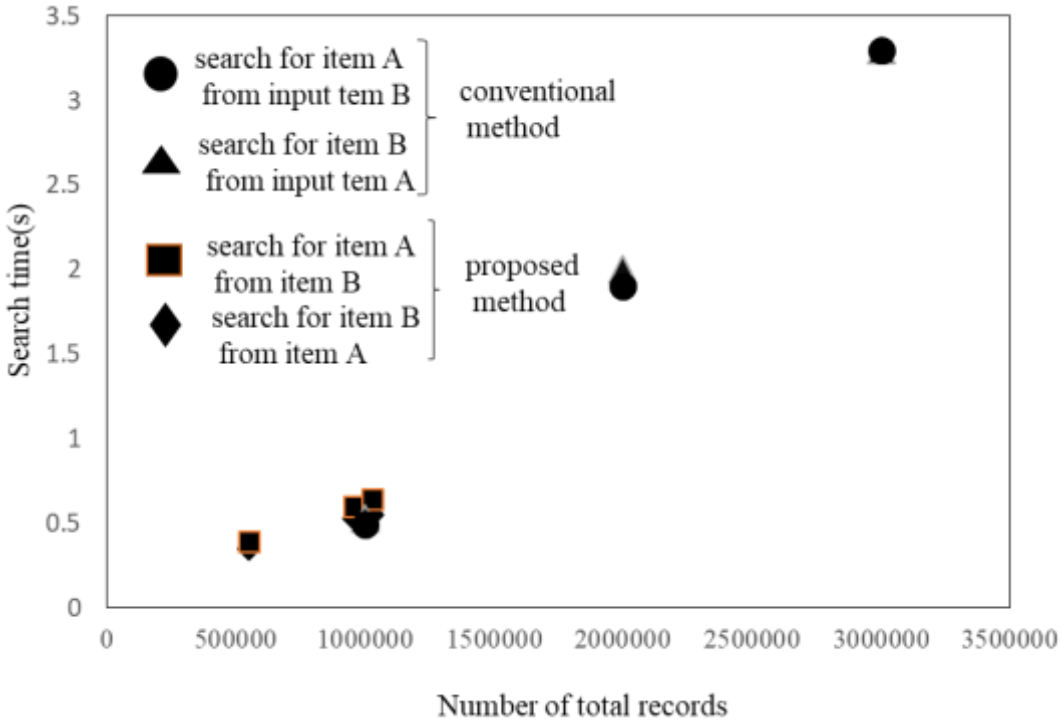

Figure 10: Search times as a function of total record number using data in main memory for both methods with random data.

5 Conclusions

This paper proposed a new data structure for applying many-to-many relations between two item types that reduces data size, recording times, and search times. In the proposed data structure, container variables are newly introduced to connect multiple values of two items. This paper also proposed algorithms for extracting container variables and for recording and searching data under the proposed data structure. Simulations using regularly arranged sample data showed that the proposed data structure is far superior to the conventional data structure in terms of data size and searching data in main memory, superior in terms of data recording, and equivalent in terms of searching data in an RDB. Simulation results on randomly arranged data showed that data sizes and search times when using data in main memory became smaller under the proposed method as compared to the conventional method. These results also suggested that search times using data in main memory are approximately proportional to the total data size.

The reduced data size under the proposed method might show good performance in other data processing tasks, such as data transmission to other nodes or making copies. The proposed method might also show promise for application to processing data in main memory, which is often performed in NoSQL approaches such as key–value systems.

Future research will investigate applications of the proposed data structure to practical situations. One goal for applying the proposed data structure will be inclusion of the proposed algorithms as functions or template programs in a database management system, especially for NoSQL, which processes data in main memory.

6 References

[1] Rajesh, N. ,"Database Management Systems (Second Edition)". pp 51-67, PHI Learning Private Limited, New Delhi,(2012)

[2] Joy L. Starks, Philip J. Pratt, Mary Z. Last," Concepts of Database Management (Ninth edition)" , pp 138-141, Cengage, Boston (2018)

[3] ISLAM A. K. M. Tauhidul, PRAMANIK Sakti, ZHU Qiang,"The BINDS-Tree: A Space-Partitioning Based Indexing Scheme for Box Queries in Non-Ordered Discrete Data Spaces", IEICE Transactions on Information and Systems, E102.D(4), 745-758, (2019)

[4] Motohashi, S., Kawano, T., Tsurumi, T., "Basic Knowledge of NOSQL", pp154-158, pp171-174, RICTELWCOM, Tokyo (2012)

[5] Watanabe, T., et.al. ,"A guide to NoSQL database for an enterprise-level RDB engineer", pp 120-131, pp420-461, SHUWA SYSTEM, Tokyo (2016)

[6] Kuwabara, T. ," Information Search Device, Program for Search, Database-Updating Device, and Program for Updating Database", Japan Patent No. 6269884, Kanagawa University (patentee), Filed May 19, 2017, Issued Jan. 12, 2018. (2018)

[7] Kuwabara, T., ” New data structures to reduce searching time on databases”, Proceedings of the IEICE General Conference 2018 (Tokyo, Japan, March 20–23, 2018). The Institute of Electronics, Information and Communication Engineers, Tokyo, Japan, D-4-7, 28.(2018)

[8] Kuwabara, T. ,” Information Search Device, Program for Search, Method for Updating Database, Database-Updating Device, and Program for Updating Database”, PCT Patent Application No. PCT/JP2018/018419, Kanagawa University (applicant), Filed May 11, 2018. (2018)

[9] Kuwabara, T. ,”Data Structure, Information Search Device, Method for Updating Database, Database-Updating Device, and Program for Updating Database”, Japan Patent Application No. 2018-090308, Kanagawa University (applicant), Filed May 9, 2018. (2018)

[10] Kuwabara, T. ,” New data structures to reduce data size and search time”, Proceedings of FIT2018 (Fukuoka, Japan, Sep. 19–21, 2018). The Institute of Electronics, Information and Communication Engineers, Tokyo, japan, and Information Processing Society of Japan, Tokyo, Japan, CD-001, No. 2, 1-4. (2018)

[11] Kuwabara, T., ”New Data Structure for Many-to-Many Relations to Reduce Data Size, Recording Time, and Search Time”, 2020 9th International Congress on Advanced Applied Informatics, pp355-360, IIAI AAI (2020)