International Journal of Networked and Distributed Computing, Vol. 3, No. 2 (April 2015), 119-127

Incremental Data Migration for Multi-Database Systems Based on MySQL with SPIDER Storage Engines

Naoyuki Miyamoto
Graduate school of engineering, University of Fukui,
3-9-1 Bunkyo, Fukui City, Fukui Prefecture 910-8507, Japan

Ken Higuchi
Graduate school of engineering, University of Fukui,
3-9-1 Bunkyo, Fukui City, Fukui Prefecture 910-8507, Japan
E-mail: higuchi@u-fukui.ac.jp

Tatsuo Tsuji
Graduate school of engineering, University of Fukui,
3-9-1 Bunkyo, Fukui City, Fukui Prefecture 910-8507, Japan
E-mail: tsuji@u-fukui.ac.jp

Abstract

In this paper, an incremental data migration technique is evaluated in the multi-database system based on MySQL with SPIDER storage engine and the improvement of the turn-around times of other operations is proved in this systems. In this method, a large data migration is divided into small data migrations and other operations are inserted between these small data migrations. This technique is easy to implement in the multi-database system.

Keywords: distributed database system, multi-database system, data migration, incremental data migration.

1. Introduction

Nowadays, database systems are used frequently in every field because the costs of introduction and running of the database system became low. What is more, network systems are being developed as well as database systems. Based on these backgrounds, a demand for sharing existent databases system grows. However, it is difficult to stop existent databases system to reorganize into one large database because database systems usually are used as mission-critical system such as the management system for goods in stock. Thus, an integrated system that consists of many existent databases is necessary, such as the multi-database system. Thereby, users can access to any data subset in

Published by Atlantis Press

all existent databases though the multi-database system and this operation can be processed in parallel.

On the other hand, there are some problems in the distributed database system including the multi-database system. One of the problems is performance decrement from load imbalance among individual databases. Thus, it is important to reorganize data partition in multi-database systems.

Reorganization of data partition includes data migration. Because data migration includes large amount data deletion and data insertion, it has a bad influence on processes for other queries in the multi-database system. There are some suggestions improving influence caused by data migration.[1,2,3,4,5] These suggestions are based on snapshot function, however, all database systems don't have this function because the capability to make snapshot is quite special. Then, not all multi-databases system can use these techniques for data migration.

One way to solve this problem is incremental data migration[9]. This method was originally based on incremental on-line reorganization scheme[8] for distributed index system[6,7]. It is only needs transaction function which implemented in many database systems. This method is based on typical operation such as insertion and deletion, and does not use snapshot function. Thus, this method is implemented easily in any multi-database system, but the research in Ref. 9 evaluated only on the set of independent database systems, not on the proper multi-database system with respect to the distributed transaction. Thus, it is necessary to evaluate the incremental data migration technique on the proper multi-database system.

In this paper, we implement the multi-database system based on MySQL with SPIDER storage engine, and evaluate the incremental data migration technique on it. The experimental result proves the improvement of turn-around times of other operations.

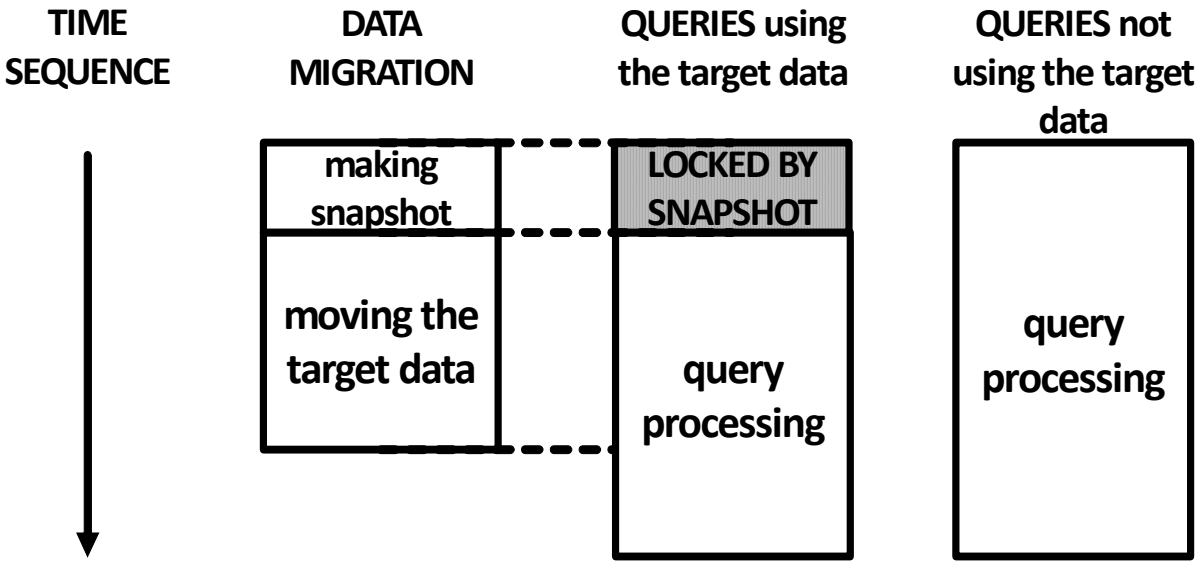

Fig. 1. a data migration operation and other operations in the data migration scheme with making snapshot

2. Data Migration in Multi-Database Systems

Multi-database system is united several database systems on network without physical reorganization. Users can access to the data in the multi-database system as one database system. It is one kind of distributed database systems, but it is different from typical distributed database systems that each component databases systems are able to provide their own service while operating as integrated database system.

When we use a multi-database system, it is necessary to reorganize data partition in the multi-database system in several purposes as mentioned in Sec. 1. Many reorganization techniques are already proposed.[1,2,3,4,5] These typical techniques use the snapshot function. However, the time to make snapshot is not short, what is worth, all database systems don't have this function. Therefore, it is not easy for multi-database systems to implement snapshot function. Then, another technique is necessary in multi-database systems. Fig. 1. shows the data migration operation using the snapshot function.

One of these solutions is classical data migration technique which only uses typical transaction function. It only uses the operation of insertion and deletion. However, in order to execute data migration safely, it is necessary to avoid conflicts with other processes by using exclusive access control, such as the locking table. Thus, as Fig. 2. shows, by executing the data migration operation, the turn-around times of other queries are more degraded than that without the data migration operation. So it is important to execute data migration more effectively.

3. Incremental Data Migration

In Ref. 9, the incremental on-line reorganization scheme for distributed index system[8] is adapted to the data migration operation in multi-database systems. This solution is named incremental data migration. Fig. 3. shows an overview of incremental data migration. This method divides one large data migration into small data migrations. By inserting other queries between small data migrations, the turn-around times of other queries are improved. However, some small data migrations are repeated incrementally until all target data are moved. In this method, the number of exclusive control including locking operation is increased. But by dividing the data migration operation, these locking areas are expected to become small and the increase of the total cost of these locking operations is expected to be not so much. Furthermore, the inserted queries can be started early and turn-around times can be improved. However, incremental data migration in multi-database system causes other problems. In multi-database system, these data migrations can be considered as data migrations between component tables in local databases. When some small data migrations have been finished but other small data migrations are not started

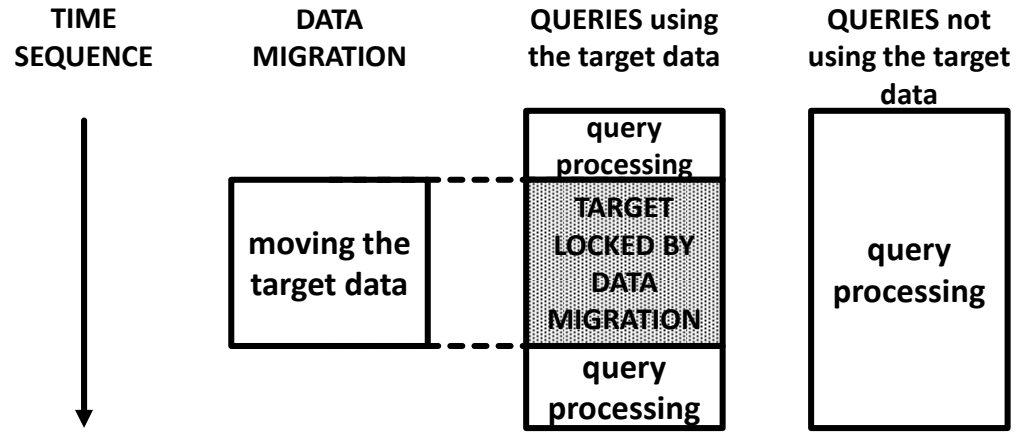

Fig. 2. a data migration operation and other operations in the data migration scheme *without* making snapshot.

Published by Atlantis Press

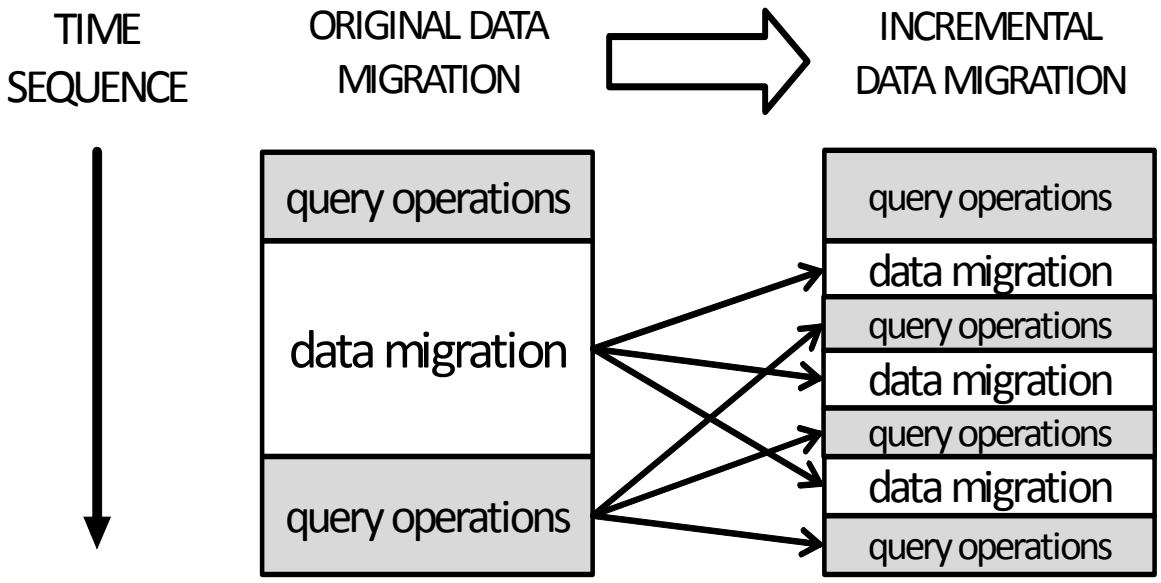

Fig. 3. the incremental scheme for the data migration

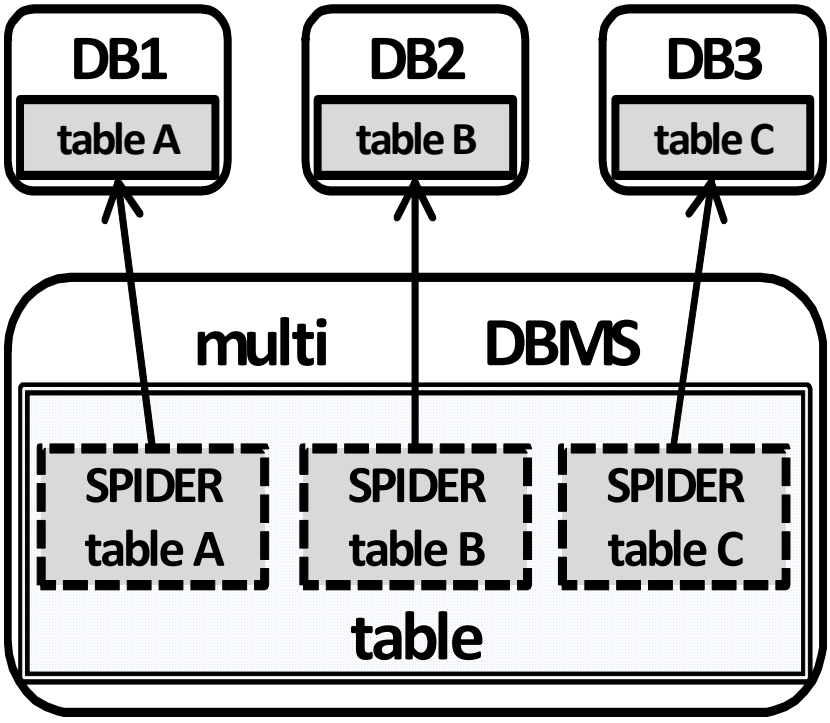

Fig. 4. a structure of multi-database system using the SPIDER storage engine..

yet, inserted queries are processed by using these halfway tables. Then, the correctness of these queries which use only one local component table is not ensured. These problems include foreign key constraint also. However, in multi-database system, since these component tables are regarded as one large table, these data migrations are regarded as internal data migration in one big table. Thus, by transforming the query only using component table into the query for multi-database system, these problems are ignored.

4. Multi-Database Based on MySQL with SPIDER Storage Engine

MySQL[10] is the relational database management system developed and provided by Oracle. One of the biggest features of MySQL is that users can choose various storage engines depending on their purposes. For example, a user can choose Memory storage engine if he wants to access database table rapidly.

The SPIDER storage engine[11] is one of the MySQL storage engines developed by Kentoku Shiba. It doesn't have its own data record, and only has reference records to other databases by using table links. In other word, the SPIDER storage engine is the collection of links such as symbolic links of UNIX file systems. The reference record is not only the link to the table in the same computer but also the link to the table in other computer. Users can accesses to tables in many computers as one table by using SQL statements without special descriptions. However, the SPIDER storage engine is not a default storage engine of MySQL, so, it is necessary to recompile the existing MySQL server.

In addition, the table defined by the SPIDER storage engine performs as the original table that is linked. For example, the table linked to the table defined by the InnoDB storage engine has the same ability as the InnoDB, which includes the record-level lock function and the transaction function. On the other hand, the table linked to the MyISAM table doesn't have row-level lock function and the transaction function. Furthermore, the SPIDER storage engine supports XA transaction internally. Then, it supports distributed transaction.

By using the SPIDER storage engine, the multi-database system is implemented easily. Fig. 4. shows a structure of multi-database using the SPIDER storage engine. This multi-database system has three links to tables on other database servers. In this multi-database, a query of the multi-database system is divided to sub-queries of DB1, DB2, and DB3. The each sub-query is processed in the corresponding database in parallel and the results of sub-queries are unified and processed in the multi-database system. In addition, the SPIDER storage engine ensures synchronization of update among linked tables on other database systems by using distributed transaction function internally. Thus, a user can update record keeping the consistency of multi-database systems. Furthermore, transactions on the multi-database system satisfy ACID property by SPIDER storage engine, not only for each database but also for the multi-database systems, that is, concurrency control capability is realized in this system. In this paper, transaction isolation level is set SERIARIZABLE.

5. Experiments

In this experiment, we implement a multi-database systems based on MySQL with SPIDER storage engine, and evaluate the improvement by using our incremental data migration in multi-database systems.

Experiment conditions are listed below:

(i) The number of remote databases server is 4 and each database server has 4 tables.
(ii) Each table consists of 5 attributes (integer(4), character(20), double(8), double(8), double(8)) and stores 1,000,000 records.
(iii) The number of databases in the multi-database system is 4 and each database has table link to each

remote database server. Then, each database has 4 table links.

(iv) Only one node controls whole data migration. But plural operations can be processed in parallel if possible.

(v) Data migrations are executed 4 times in the multi-database system. Each data migration moves 500,000 records to other table cyclically in same multi-database system.

(vi) The number of clients for queries is 16 and each client requests queries repeatedly. Each query has to be access to tables in only one database system and the client change the target node in order.

(vii) The specification of each node is written in Table 1.

Here, a set of successive 4 queries is called a cycle. In one cycle, query processing accesses 4 tables in one multi-database system.

As shown in the Fig. 5., in this condition, the number of steps of the incremental data migration is 1, 5, and 10. Here, 1-step data migration is equal to the data migration without incremental scheme.

Table 1. specification of nodes

ITEM	VALUE
the number of nodes	21
CPU	Intel Core I 5 - 650 (3.2GHz)
OS	Fedora 14 (x86)
MEMORY	4GB
NETWORK	1G bps Ethenret
DATABASE SYSTEM	MySQL 5.5.14

5.1. *Results of Experiment*

Fig. 6. and Fig. 7. show the results of experiment. Horizontal axis is the cycle number. One curve is the average of the total execution times of 16 clients. Here the total execution time means the finish time of some cycle and not a turn-around times of that cycle. Fig. 6 shows the result to 30 cycles and Fig. 7. is the magnification of Fig. 6.

In this result, the curves of the 1-step data migration have one small step between 2nd cycle and 4the cycle. It is clearer in Fig.7. It caused by competition among queries and the large data migration which costs long time. On the other hand, the curves of the 5-steps data

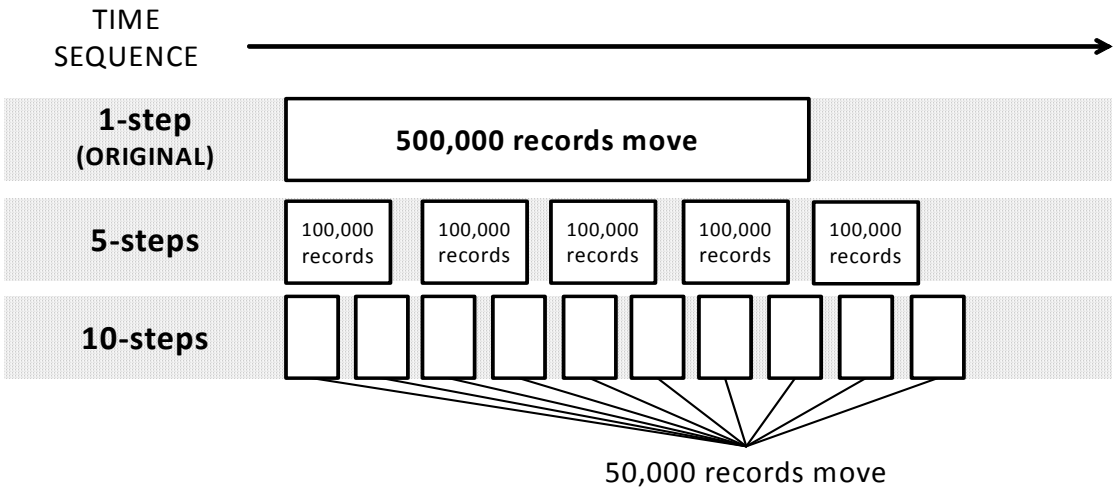

Fig. 5. the incremental data migration in this experiment

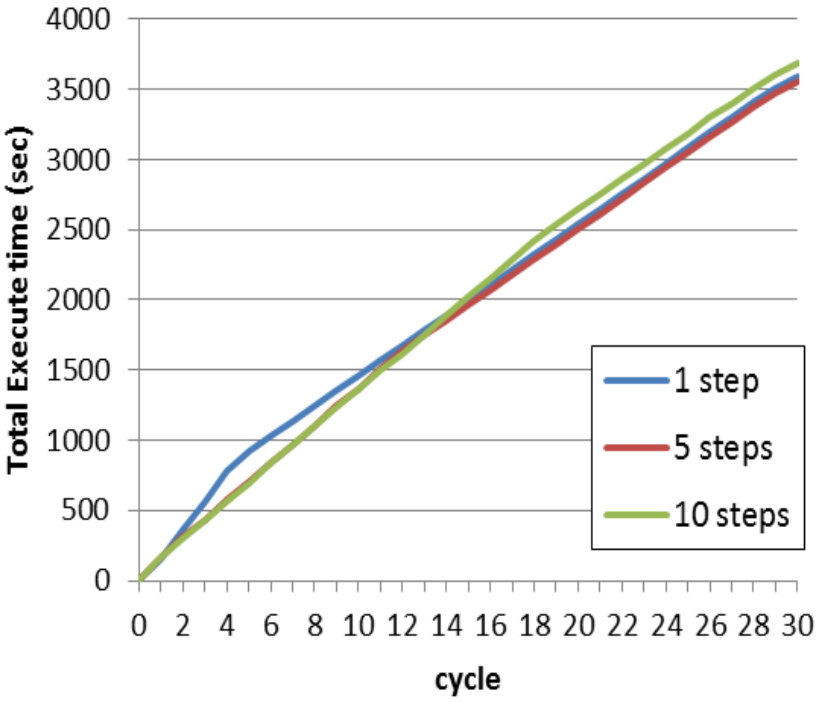

Fig. 6. result of the experiment

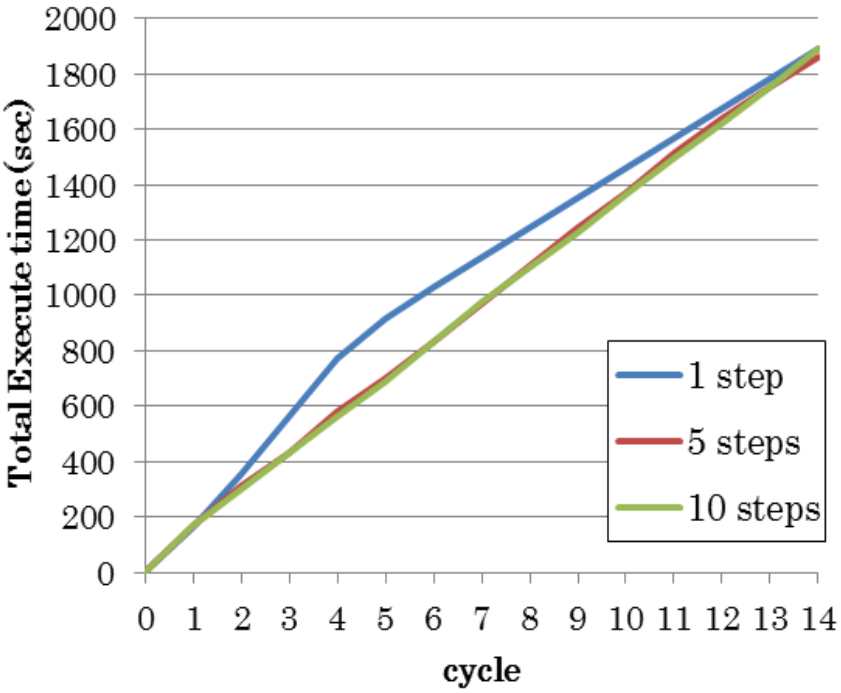

Fig. 7. magnification of Fig. 6.

migration and the 10-steps data migration are smooth and lower than that of 1-step data migration. Total execution time of queries in the 5-steps data migration is a little shorter than that in 1-step data migration, but that in the 10-steps data migration is a little longer than that in the 1-steps data migration as queries are processed. Table 2. shows the performance gain at 4th cycle, which is ratio to 1-step performance. Table 3. shows the performance gain at 30th cycle. These results mean the response time of the query inserted between data migrations is clearly improved. On the other hand, the response time of the query after the whole data migration is degraded in 10-step data migration. Then, 5-step data migration is the best solution in this experiment because total performance gain is a little better than that of 1-step data migration.

In these circumstances, by dividing data migration, turn-around times of other queries are surely improved. If a data migration is divided appropriately (i.e. appropriate number of division), some good effect is expected. But if a data migration is divided inappropriately (i.e. too many number of division), negative effect is expected because of the increase of the number of exclusive control executions including locking operations. The appropriate number of division depends of the cost of exclusive control and the amount of data migration..

6. Conclusions

We evaluated the incremental data migration technique on the multi-database system based on MySQL with SPIDER storage engine. From the experimental result, our incremental data migration technique is effective for the improvement of the execution time (except for some situation). Because this multi-database system ensures serializability, the execution time in this situation is longer than that in Ref. 9, which doesn't ensure serializability. On other hand, we proved the effectiveness of incremental data migration in this system.

For future works, evaluation of the performance of our scheme in more real situations is necessary.

Table 2. performance gain at 4th cycle

number of step	power
1-step	1.000000
5-steps	1.334172
10-steps	1.376289

Table 3. performance gain at 30th cycle

number of step	power
1-step	1.000000
5-steps	1.010725
10-steps	0.975545

References

1. B. Salzberg, and A. Dimock, Principles of Transaction-Based On-line Reorganization, in *Proc. 18th International Conf. on Very Large Data Bases* (1992), pp. 511-520.
2. K. Achyutuni, E. Omiecinski, and S. Navathe, Two techniques for on-line index modification in shared nothing parallel database, in *Proc. the 1996 ACM SIGMOD International Conf. on Management of Data* (1996), pp. 124-136.
3. E. Omiecinski, Concurrent File Reorganization: Clustering, Conversion and Maintenance, *Data Engineering Bulletin*, **19(**2) (1996), pp. 25-32.
4. C. Zou and B. Salzberg, Safely and Efficiently Updating References During On-line Reorganization, in *Proc. 24th International Conf. on Very Large Data Bases* (1998), pp. 512-522.
5. M. K. Lakhamraju, R. Rastogi, S. Seshari, and S. Sudarshan, On-line Reorganization in Object databases, in Proc. *the 2000 ACM SIGMOD International Conf. on Management of Data* (2000), pp.58-69.
6. K. Higuchi and T. Tsuji, and T. Hochin, Distributed Index System for Complex Objects with On-line Modification, *IPSJ Trans. of Databases*, **43**, SIG12 (TOD16) (2002), pp.64-79.
7. K. Higuchi and T. Tsuji, On-line Reorganization for Distributed Index System for Complex Objects, *IPSJ Trans. of Database*, **45**, SIG10 (TOD23) (2004), pp. 1-17.

8. K. Higuchi, T. Nomura, and T. Tsuji, Incremental reorganization for distributed index system, Systems Modeling and Simulation, in Proc. *Theory and Applications Asia Simulation Conference* (2006), pp.223-227.
9. K. Higuchi, W. Wang, and T. Tsuji, Incremental Data Migration for Multi-Database Systems, in Proc. *13th ACID international Conference on Software Engineering, Artificial Intelligence, Networking and parallel/Distributed Computing* (2012), pp. 716-720.
10. Oracle Corp., MySQL.com, *http://www.mysql.com/*
11. K. Shiba, SpiderForMySQL.com, *http://spiderformysql.com/*