Incremental Data Migration for Multi-Database Systems

Incremental Data Migration for Multi-Database Systems

Ken HIGUCHI
Graduate school of Engineering
University of Fukui
Fukui, Japan
higuchi@u-fukui.ac.jp

Wenqian WANG
AISIN AW CO., LTD.
Anjo, Japan
wen@pear.fuis.u-fukui.ac.jp

Tatsuo TSUJI
Graduate school of Engineering
University of Fukui
Fukui, Japan
tsuji@u-fukui.ac.jp

Abstract— **Nowadays, database systems are one of the most popular and essential software programs in computer systems. Many computers are installed a database system and users maybe want to use these database systems as one system. The multi-database system is one of the solutions to this request. The multi-database system is a kind of the distributed database system. It is a cluster of independent database systems. As the distributed database system, the multi-database system has some problems. One of these problems is data migration among individual database systems in the multi-database system. Many reorganization techniques for distributed database systems already proposed. But these techniques are not always adaptive to the data migration in the multi-database system. In order to overcome this problem, we adapt the incremental scheme to the data migration in the multi-database system. In our new scheme, a large data migration operation is divided into small ones, and other operations are inserted between them. The experimental result proves the improvement of the turn-around times of other operations.**

Keywords- multi-database, data migration, incremental data migration

I. INTRODUCTION

Nowadays, database systems are one of the most popular and essential software programs in computer system. Because the introduction cost and the running cost of the database system become to be low, many personal computers can be installed and use the database system. It is natural that users want to access large data in many database systems that are running on independent computers. One of its solutions is using the multi-database system. The multi-database system is a kind of distributed database systems. It is a cluster of independent database systems. But it does not have a centralized data management method for whole data in the multi-database system. Each data is stored and managed by an independent database system running on one computer. While it is necessary that each database system has all facilities of the database system, it is not necessary that the multi-database system has them. In extreme cases, some multi-database system only has the access method to data. Then, the multi-database system can be constructed from existing database systems. Therefore, the construction of the multi-database system is easier than that of the centralized controlled distributed database system.

By using the multi-database system, user can access to data in plural database systems. These data is larger than that in a single database system, and user can benefit from it. But the multi-database system has some problems that are the same as the distributed database system. One of these problems is data migration. In the distributed database, in order to keep the load balancing or to reorganize to the data partitioning, the data migration among individual database systems is necessary. It is caused by change of the tendency of retrieval queries, insertion of a large amount of data, a global strategy for the database, and so on. The distributed database can use its concurrency control to the data migration [1][2][3][4][5]. But these techniques are not always adaptive to the data migration in the multi-database system (discussed in next section).

One way to solve the data migration in the multi-database system is an incremental data migration scheme based on the incremental online reorganization scheme in [8]. Because it does not use any snapshots or versions of data, it can be adapted to the data migration in the multi-database system.

In this paper, we adapt the incremental scheme to the data migration in the multi-database system. In our new scheme, one data migration operation is divided into small ones, and other operations are inserted between them. The experimental result proves the improvement of turn-around times of other operations.

II. DATA MIGRATION IN MULTI-DATABASE SYSTEMS

In this paper, we only consider isolation and consistency to the multi-database system. Because each individual database system is managed according to its specification, its isolation and consistency are kept by that individual database system. Furthermore, we assume that the isolation level of the multi-database system is SERIALIZABLE.

As mentioned in Sec. I, many reorganization techniques are already proposed [1][2][3][4][5]. These typical techniques use the snapshot function to target data of the data migration (Fig. 1-(a)). By using the snapshots, other queries can be processed while processing the data migration (Fig. 2). These techniques can be adapted to the data migration of the centralized controlled distributed database system. But these techniques are not always adaptive to the data migration in the multi-database system, because these functions are not always implemented in the individual database system in the multi-database system. Then, in the multi-database system, other technique to the data migration is necessary.

One way to solve the data migration in the multi-database system is the classical data migration technique that uses the locking function, the insertion function, and the deleting function of the individual database system in the multi-database system (Fig. 1-(b)). In this scheme, queries that use the target data of the data migration cannot be processed until the data migration is completed (Fig 3). Then, turn-around times of other queries using this scheme are more degraded than that using the data migration scheme using the snapshots. On the other hand, the turn-around time of the data migration operation using this scheme is shorter than that using the data migration scheme using snapshots. It can be adapted to almost all of multi-database systems. Then, we consider only this classical data migration technique.

1. lock queries that use the target data
2. make a snapshot for the target data
3. unlock that queries
4. move the target data (or the snapshot)
5. lock queries that use the target data
6. delete the snapshot (or the target data)
7. unlock that queries

(a) The data migration algorithm using snapshots.

1. lock the queries that use the target data
2. move the target data using insertion and deletion
3. unlock that queries

(b) The data migration algorithm without a snapshot.

Figure. 1 data migration algorithms for the multi-database system

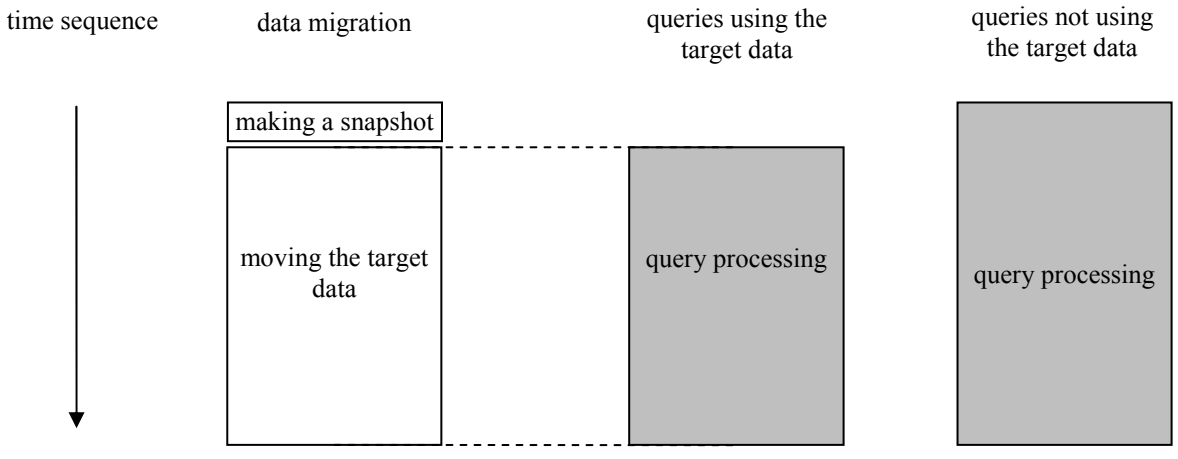

Figure. 2 a data migration operation and other operations in the data migration using snapshots

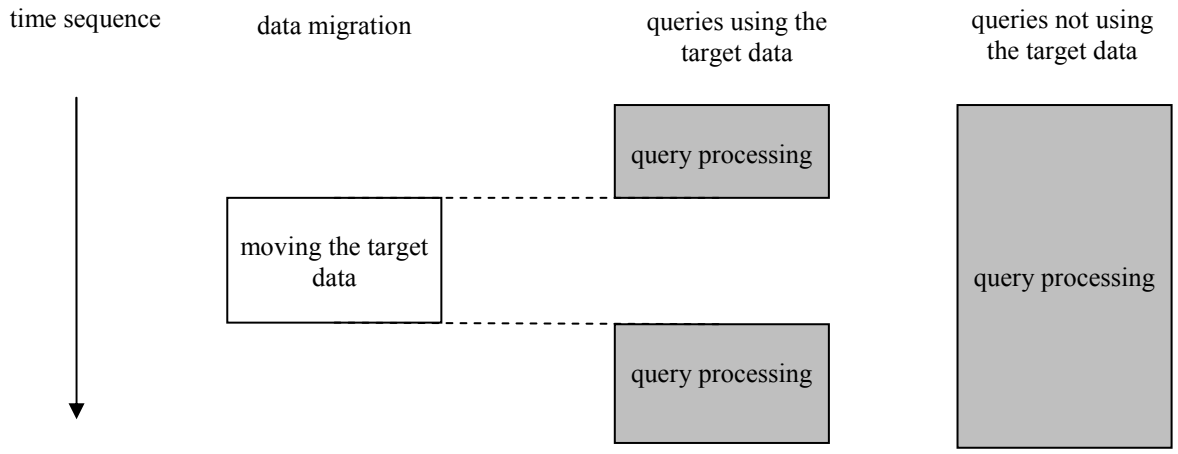

Figure. 3 a data migration operation and other operations in the data migration scheme without snapshot

III. INCREMENTAL DATA MAIGRATION

In [8], we proposed an incremental on-line reorganization scheme for distributed index systems. In this scheme, a large reorganization operation is divided into small reorganization operations and other queries are inserted to these small reorganization operations. By this insertion of other queries, turn-around times of these queries are improved. This incremental scheme can be adapted to the data migration in the multi-database system because the data migration is included in the reorganization on the distributed database system. Fig. 4 shows an overview of the incremental data migration scheme. Fig. 5 shows the procedure of one step of the incremental data migration. The incremental scheme repeats this step until all target data are moved. Then, the number of these locking operations is increased. But by dividing the data migration operation, these locking areas are expected to become small and the total cost of these locking operations is expected to be not much different. Furthermore, the inserted queries can be started early and these turn-around times can be improved.

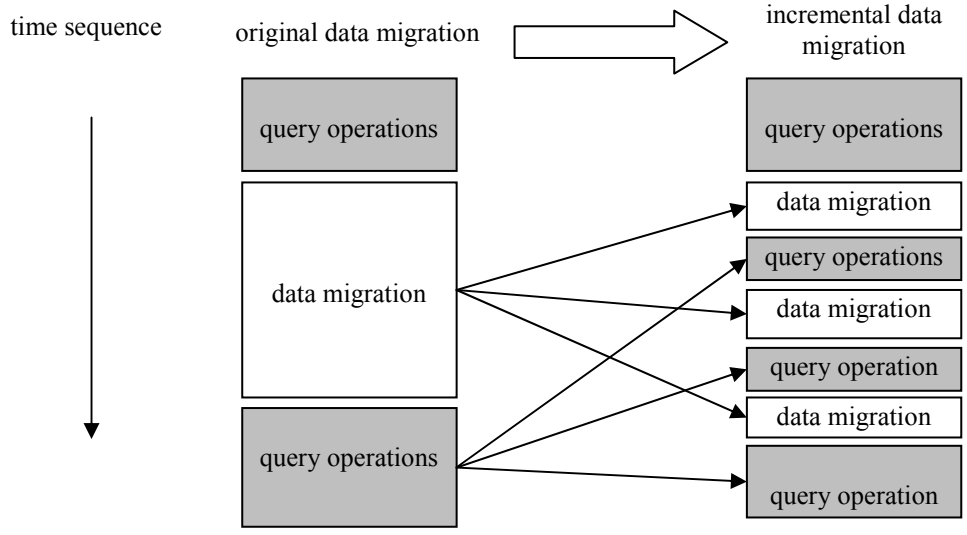

Figure. 4 the incremental scheme for the data migration in the multi-database system

1. get an exclusive lock to the target data in the source database system
2. get some locks in the destination database system if necessary
3. transfer the target data
4. insert the target data to the destination database system
5. delete the target data in the source database system
4. unlock (according to 1 and 2)

Figure. 5 one step of the incremental data migration in the multi-database (from source database system to destination database system)

IV. EXPERIMENTS

To show the improvement by our incremental data migration scheme, we evaluate its performance by experiments on a cluster of database systems using MySQL[9]. The multi-database system used in the experiment is an elementally one. It can process only retrievals and data migrations using MySQL API.

A. Experiment Environment

Experiment conditions are listed below:

(a) The number of database servers is 4 (db1, db2, db3, db4). Each server is running on an individual node (Table 1).
(b) The number of tables in one server is 4 (t1, t2, t3, t4).
(c) Only one node controls all data migration. But plural operations can be processed in parallel if possible.
(d) Our scheme is evaluated in following two conditions. In each data migration, target records are moved between two tables that have same name.
  (Exp.1) Each table stores 2,500,000 records. In each table, 500,000 records are moved to another table in another node.
  (Exp.2) 4 tables in db1 store 2,000,000 records respectively and other tables store 1,000,000 records respectively. In each table of db1, 750,000 records are moved to other tables in db2, db3, and db4 in order to equalize the numbers of records in one table.
  In each 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.
(e) The number of clients for queries is 16 and each client requests queries repeatedly. Each query has to be access to only one table in some node and the client changes the target node in order.

Here, a set of successive 4 queries is called a cycle. In one cycle, query processing accesses 4 tables that are stored by separate database systems. One cycle is not one transaction but it uses all database servers. Then, one cycle is not a query to the multi-database system but it can evaluate the performance of our scheme as the multi-database system.

TABLE I. SECIDICATION OF NODES

the number of nodes	21
CPU	Intel Corei5-650 (3.2GHz)
OS	Fedra 14
network	1 G bps Ethernet
database system	MySQL 5.5.14

In Exp.1, the numbers of records stored in one table are equal to each other before the data migration and after the data migration. Exp. 1 can evaluate our scheme without the influence of the improvement of load-balance. In Exp.2, before the data migration, db1 stores records whose amount is double of other database servers. After the data migration, the numbers of records stored in one table are equal to each other. Then Exp. 2 can evaluate our scheme in the data migration that improves load-balance.

In both conditions, experiments are executed ten times and the results of these experiments are averages of them.

B. Results of Experiment

Fig. 6 and Fig. 7 show results of Exp1. and Exp2 respectively. Vertical axes are the total execution time (sec) and horizontal axes are 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 time of that cycle (the difference between the total execution times of two successive cycles is the turn-round time).

In both two experiments, the curves of the 1-step data migration have one clear step. It is caused by the data migration operation and many execution times of other queries in this step are degraded by it. On the other hand, the curves of the 5-steps data migration and the 10-steps data migration are smooth and lower than that of 1-step data migration. In each client, curve of total execution time has 5 or 10 steps but timings of these steps are different to each other. Then, curves in Fig. 5 and Fig 6 become to be smooth. On comparison between the 5-steps data migration and the 10-steps data migration, total execution times of queries in the 10-steps data migration are faster than that in the 5-steps data migration in Exp.1. In Exp. 2, the total execution time of the last query in the 10-steps data migration is equal to that in the 5-steps data migration. It is because that the source database server of the data migration in Exp.2 is only one, many tables are not locked, and many queries can be processed. On the other hand, in Exp.1, many divided data migration operations are processed in parallel and many tables are locked. Because the duration of locking for one step in the 5-steps data migration is longer than that in the 10-steps data migration, the 10-steps data migration is more efficient than the 5-steps data migration. The deference between curves of the 1-step data migration and curves of the 5-steps data migration is caused by the same reason.

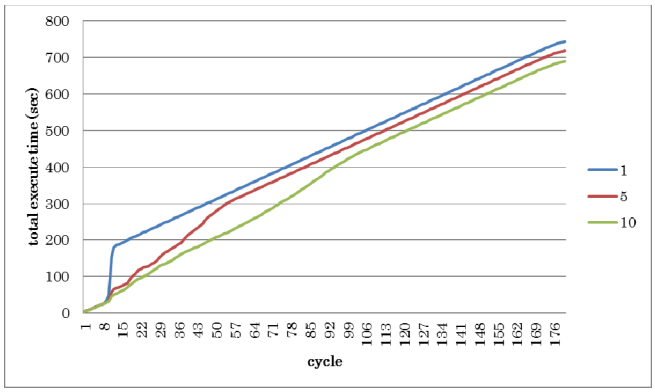

Figure. 6 results of Exp.1

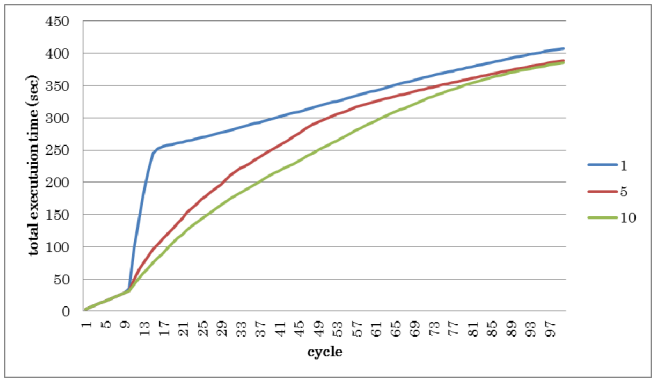

Figure. 7 results of Exp.2

V. CONCLUSIONS

We adapted the incremental scheme to the data migration in the multi-database system. By experimental results, our incremental scheme is effective for the improvement of execution time. But queries of our experiments are not corrective queries for the multi-database system. For future works, we need to evaluate the performance of our scheme in more real situations.

REFERENCES

[1] Salzberg, B., and Dimock, A., Principles of Transaction-Based On-line Reorganization, Proc. of 18th International Conf. on Very Large Data Bases, pp. 511-520, 1992..

[2] Achyutuni, K., Omiecinski, E., and Navathe, S., Two techniques for on-line index modification in shared nothing parallel database, Proc. of the 1996 ACM SIGMOD International Conf. on Management of Data, pp. 124-136, 1996.

[3] Omiecinski, E., Concurrent File Reorganization: Clustering, Conversion and Maintenance, Data Engineering Bulletin, 19, 2, pp 25-32, 1996.

[4] Zou, C. and Salzberg, B., Safely and Efficiently Updating References During On-line Reorganization, Proc. of 24th International Conf. on Very Large Data Bases, pp. 512-522, 1998.

[5] Lakhamraju, M. K., Rastogi, R., Seshari, S., and Sudarshan, S., On-line Reorganization in Object databases, Proc. of the 2000 ACM SIGMOD International Conf. on Management of Data, pp.58-69, 2000.

[6] Higuchi, K. and Tsuji, T., and Hochin, T., Distributed Index System for Complex Objects with On-line Modification, IPSJ Trans. of Databases, 43, SIG12(TOD16), pp.64--79, 2002.

[7] Higuchi, K. and Tsuji, T., On-line Reorganization for Distributed Index System for Complex Objects, IPSJ Trans. of Database, 45, SIG10(TOD23), pp 1-17,2004

[8] Higuchi, K., Nomura, T, and Tsuji, T., Incremental reorganization for distributed index system, Systems Modeling and Simulation, Proceedings of Theory and Applications Asia Simulation Conference 2006, pp 223-227, 2006.

[9] www.mysql.com