Journal of Mathematics and Computing Science, 2021,
Volume 7, No 1, 8-14

Managing Fragmented Database in Distributed Database Environment

Ainul Azila Che Fauzi[1]*, Wan Fariza Wan Abdul Rahman[2], Auni Fauzi [3] and Florian Weigelt[4]

[1,2]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Kelantan, Bukit Ilmu, Machang, Kelantan, Malaysia
ainulazila@uitm.edu.my, wfariza@uitm.edu.my
[3]Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu, 21030, Kuala Terengganu, Terengganu, Malaysia
ainul.auni@gmail.com
[4]Media Designer, Bavaria, Germany
florian-weigelt.com

Abstract: One of the mechanisms for managing data is replication since it improves data access and reliability. However, the amount of various data grows rapidly since technology is widely available at a low-cost. Problem arises when the transactions only work with a part of the database, not as a whole.This paper will discuss about a technique called database fragmentation. Fragmentation in distributed database is very useful in terms of usage, reliability and efficiency. Handling fragmented database replication becomes a challenging issue to administrators since the distributed database is scattered into split replica partitions or fragments. We address how to build a reliable system by using a distributed database fragmentation algorithm. The result shows that managing database replication with fragmentation will help a transaction to work with the specific part of the database rather than the entire database.

Keywords: Distributed Systems, Distributed Database, Centralized Database, Database Fragmentation

1 Introduction

Organizations critically need to supply recent data to users who may be geographically remote, while at the same time handle a volume request of data. Data replication is a way to deal with this problem since it provides users with fast, local access to shared data and protects availability of applications because alternate data access options exist (1, 2, 3). Distributed database replication involves the process of copying and maintaining database objects in multiple databases that make up a Distributed Database Systems (DDS) (4, 5).

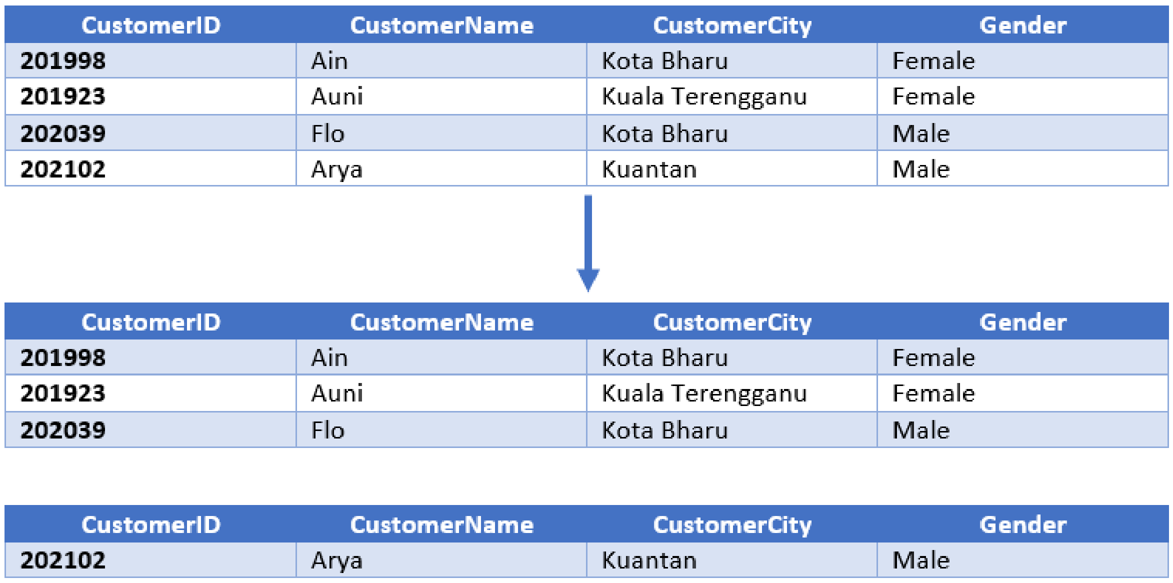
In case of database failures, the total system of centralized databases comes to a halt. However, in distributed systems, when a component fails, the functioning of the system continues may be at a reduced performance. Hence distributed systems are more reliable. However, a distributed Database Management Systems (DBMS) that hides the distributed nature from the user, works more complex than a centralized DBMS. The fact that data can be replicated also adds an extra level of complexity to the distributed DBMS. If the software does not handle data replication adequately, there wi1l be degradation in availability, reliability and performance compared to the centralized system. In addition, increased in complexity means the procurement and maintenance costs for a distributed DBMS also to be higher than centralized DBMS. Hence, the storage and communication cost are some of the important issues to be addressed in order to allow distributed users to efficiently and safely access data from many different sites.

2 Overview Database Fragmentation

Distributed database is a logically interrelated collection of shared data physically distributed over a computer network. Database can be decomposed into numbers of fragments. Fragmentation plays an important role in distributed database systems because usually, applications work with only some of the relations rather than entire of it (6, 7). In data distribution, it is better to work with subsets of relations as the unit of distribution. The other benefit from fragmentation is efficiency. Data is stored close to where it is most frequently used and for data that is not needed, it is not stored. By using fragmentation, a transaction can be divided into several subqueries that operate on fragments. So, it will increase the degrees of parallelism. Besides, it is also good for security as data not required for local applications is not stored. So, it will not be available to unauthorized users. There are mainly three type of fragmentation called horizontal, vertical and mixed.

A Horizontal Fragmentation

Horizontal fragmentation groups together the tuples in a relation that are used by the important transactions (7, 9). A horizontal fragment is produced by specifying a predicate that performs a restriction on the tuples in the relation. It is defined using the *Selection* operation of relational algebra. Given a relation *R*, a horizontal fragment is defined as $\sigma_p (R)$, where p is a predicate based on one or more attributes of the relation.



CustomerID	CustomerName	CustomerCity	Gender
201998	Ain	Kota Bharu	Female
201923	Auni	Kuala Terengganu	Female
202039	Flo	Kota Bharu	Male
202102	Arya	Kuantan	Male

CustomerID	CustomerName	CustomerCity	Gender
201998	Ain	Kota Bharu	Female
201923	Auni	Kuala Terengganu	Female
202039	Flo	Kota Bharu	Male

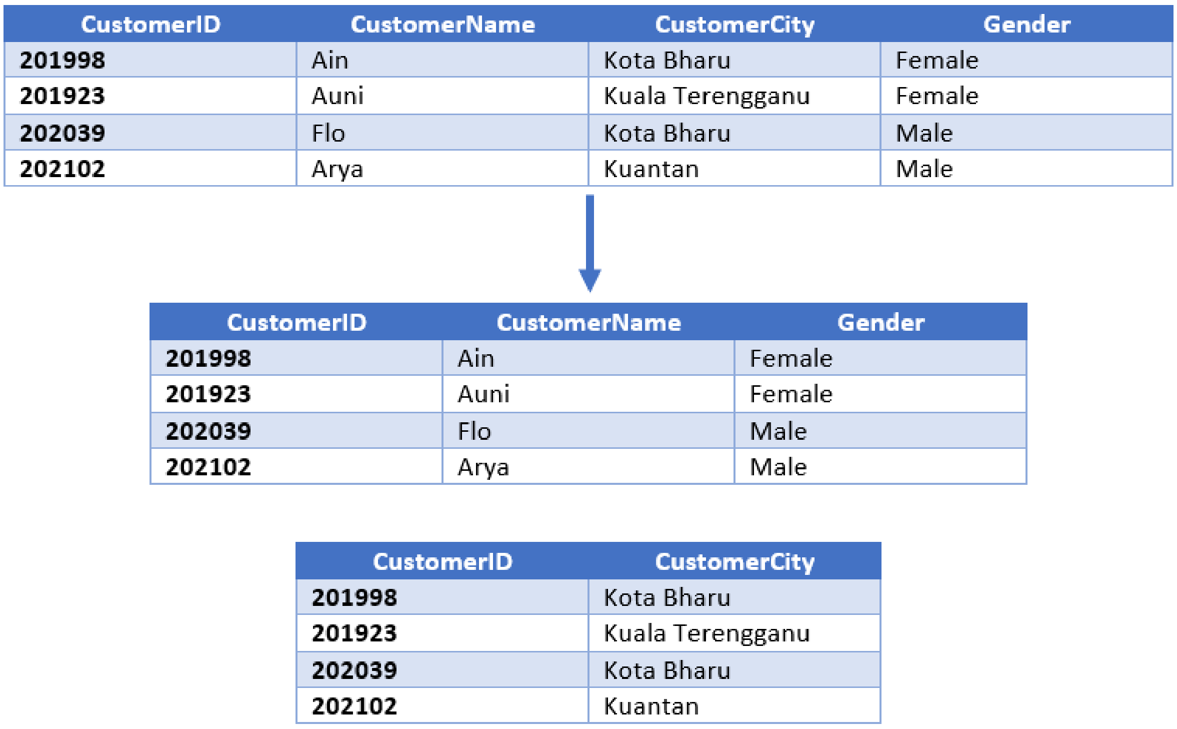
CustomerID	CustomerName	CustomerCity	Gender
202102	Arya	Kuantan	Male

Figure 1: Horizontal Fragmentation

Figure 1 shows the illustration of horizontal fragmentation. Horizontal fragmentation involves forming a subset of the global relation by selecting tuples based on the value of one or more attributes which are also called scan attributes. The type of horizontal fragmentation is known as *Primary* if the scan attributes are contained in the relation being fragmented. However if the scan attributes are contained in a relation which owns the relation being fragmented, the type of horizontal fragmentation is called *Secondary*. The benefits of horizontal fragmentation are extensive when fragments can be formed such that query or update transactions for a relation at a site is largely localized to the fragment at that site (7, 9). During horizontal fragmentation, rules to ensure completeness, reconstruction and disjointness must be followed (7, 9, 10, 11). Horizontal fragmentation is most effective if the locality of reference can be established by using site-based attributes for fragmentation. It improves response time, availability and concurrency of transactions.

B Vertical Fragmentation

Vertical fragmentation groups together the attributes in a relation that are used jointly by the important transactions (7). A vertical fragment is defined using the *Projection* operation of the relational algebra. Given a relation *R*, a vertical fragmentation is defined as $\Pi a_1, \ldots, a_n (R)$, where $a_1, \ldots, a_n$ are attributes of the relation *R*, where *n* is equal to the number of the attributes.



CustomerID	CustomerName	CustomerCity	Gender
201998	Ain	Kota Bharu	Female
201923	Auni	Kuala Terengganu	Female
202039	Flo	Kota Bharu	Male
202102	Arya	Kuantan	Male

CustomerID	CustomerName	Gender
201998	Ain	Female
201923	Auni	Female
202039	Flo	Male
202102	Arya	Male

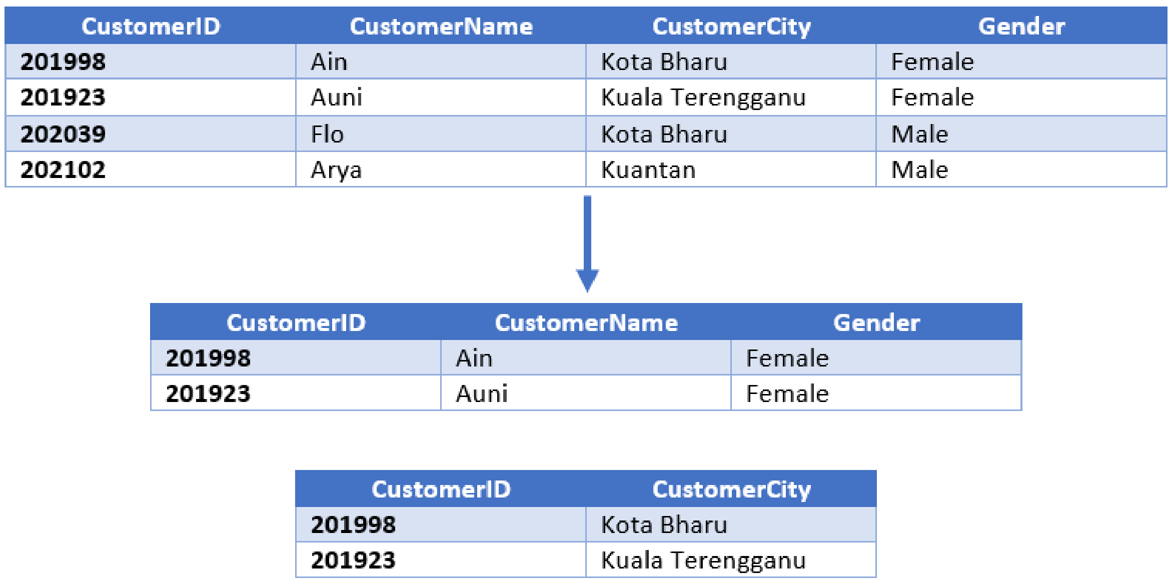
CustomerID	CustomerCity
201998	Kota Bharu
201923	Kuala Terengganu
202039	Kota Bharu
202102	Kuantan

Figure 2: Vertical Fragmentation

Figure 2 shows the illustration of vertical fragmentation. Vertical fragmentation involves dividing the attributes of a relation into groups and then projecting the relation over each group. The attributes should be no overlapping across the groups with the exception of primary key attributes or tuple ID substantial when fragments can be formed such that query or update transactions for a relation at a site are largely localized to the fragment at that site is the benefit of vertical fragmentation. Vertical fragmentation comes generally with side-effects and requires a detailed analysis for establishing its benefits such as improving response time, availability and concurrency of transactions. The additional overhead of implicit joins will have to be incurred for queries spanning multiple fragments. The use of the implicit joins is practical when different applications located at multiple sites share data and to merge the two partitioning approaches to result in mixed partitioning (7, 9). All the updates are executed on the master copy while other copies which are called snapshots are refreshed occasionally. Queries are routed to a local copy, if available. This practise improves availability and concurrency of query operations and it does not degrade response time for update operations. Vertical fragmentation requires consistency analysis at the application level and it is most suitable for networks with high delays and low reliability.

C Hybrid Fragmentation

For some applications horizontal or vertical fragmentation of a database schema by itself is insufficient to adequately distribute the data. Instead, mixed or hybrid fragmentation is required.



CustomerID	CustomerName	CustomerCity	Gender
201998	Ain	Kota Bharu	Female
201923	Auni	Kuala Terengganu	Female
202039	Flo	Kota Bharu	Male
202102	Arya	Kuantan	Male

CustomerID	CustomerName	Gender
201998	Ain	Female
201923	Auni	Female

CustomerID	CustomerCity
201998	Kota Bharu
201923	Kuala Terengganu

Figure 3: Hybrid Fragmentation

Figure 3 shows the illustration of hybrid fragmentation. A hybrid fragment is defined using the Selection and Projection operations of the relational algebra. Given a relation R, a mixed fragment is defined as σp (Πa1, . . . , an(R)) or Π a1, . . . , an(σp (R)) where p is a predicate based on one or more attributes of R and a1, . . . , an are attributes of R.

3 Database Fragmentation in Distributed Database Systems

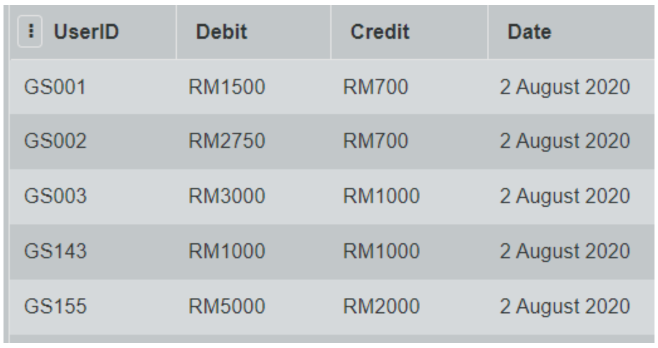
In this section, we proposed database fragmentation in distributed database systems. The following notations are defined:

i. S is relation in database.
ii. x is an instant which will be modified
iii. y is an instant which will not be modified
iv. S_1 is a vertical fragmented relation with instant x.
v. S_2 is a vertical fragmented relation without instant x.
vi. Pk is a primary key.
vii. Pk,x is a primary key with data x.
viii. Pk,y is a primary with data y, where $y \neq x$
ix. $S_{1_{(Pk,x)}}$ and $S_{1_{(Pk,y)}}$ are a horizontal fragmentation relation.

In distributed database systems, each primary replica copies other database to its replicas. The fragmentation process starts with relation S fragmented into S_1 and S_2 using vertical fragmentation. Next, the fragmented relation, S_1 will be fragmented again but this time using horizontal fragmentation into $S_{1_{(Pk,x)}}$ and $S_{1_{(Pk,y)}}$ leaving only instant for primary key and data that need to be updated.

4 Distributed Database Fragmentation Example Case Study

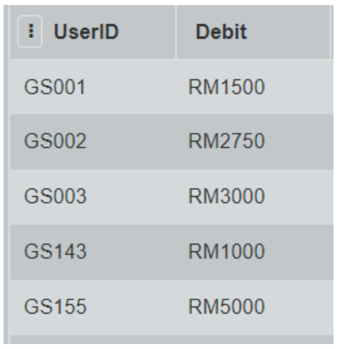
Let's say, Transaction 1 (T_1), requests to update data *Debit* for user with *UserID: GS001* from *RM1500* to *RM3000*.



UserID	Debit	Credit	Date
GS001	RM1500	RM700	2 August 2020
GS002	RM2750	RM700	2 August 2020
GS003	RM3000	RM1000	2 August 2020
GS143	RM1000	RM1000	2 August 2020
GS155	RM5000	RM2000	2 August 2020

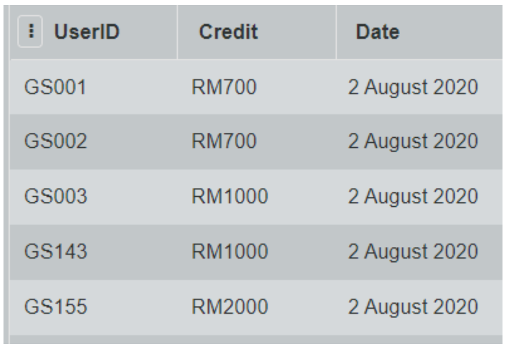
Figure 4: Relation *S*

Relation S fragmented into S_1 and S_2 using vertical fragmentation as shown in Table 3 and Table 4.
S_1 = UserID (primary key), Debit
S_2 = UserID (primary key), Credit, Date



UserID	Debit
GS001	RM1500
GS002	RM2750
GS003	RM3000
GS143	RM1000
GS155	RM5000

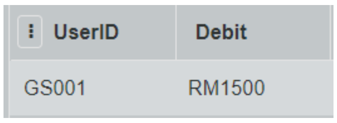
Figure 5: Relation S1



UserID	Credit	Date
GS001	RM700	2 August 2020
GS002	RM700	2 August 2020
GS003	RM1000	2 August 2020
GS143	RM1000	2 August 2020
GS155	RM2000	2 August 2020

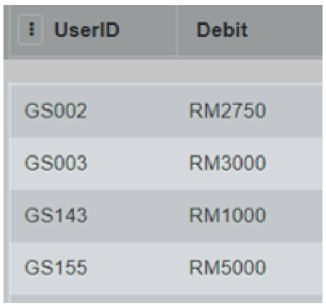
Figure 6: Relation S2

Relation S_1 then fragmented using horizontal fragmentation into $S_{1_{(Pk,x)}}$ and $S_{1_{(Pk,y)}}$. After the fragmentation, T_1 will update data *x* and only will replicate $S_{1_{(Pk,x)}}$ to the other replicas.



UserID	Debit
GS001	RM1500

Figure 7: Relation $S_{1_{(Pk,x)}}$



UserID	Debit
GS002	RM2750
GS003	RM3000
GS143	RM1000
GS155	RM5000

Figure 8: Relation $S_{1_{(Pk,y)}}$

5 Conclusion

Handling fragmented database replication is very important in order to preserve the data reliability with low communication cost. Therefore, a database fragmentation in distributed database systems has been proposed to manage the fragmented database replication. From the example case, it shows that the fragmentation is able to reduce the size of the table to smaller subsets and make them available with less network access time. Furthermore, fragmentation may help to reduce storage space because this data will be distributed evenly among replica servers instead of saving all data to all servers.

References

[1] A. Noraziah, A. A. C. Fauzi, S. H. S. A. Ubaidillah, B. Alkazemi and J. B. Odili, “BVAGQ-AR for Fragmented Database Replication Management”, *IEEE Access.* vol. 9, pp. 56168-56177, 2021.

[2] B. A. Milani and N. J. Navimipour, “A Comprehensive Review of The Data Replication Techniques in The Cloud Environments: Major Trends and Future Directions", *Journal of Network and Computer Applications*. vol. 64, pp. 229-238, 2016.

[3] J. Wang, H. Wu and R. Wang, “A New Reliability Model in Replication-Based Big Data Storage Systems”, *Journal of Parallel and Distributed Computing*. vol. 108, pp. 14-27, 2017.

[4] A.A.C. Fauzi, A. Noraziah, A. Amer and T. Herawan, “Managing Fragmented Database Replication for Mygrants Using Binary Vote Assignment on Cloud Quorum”, *Applied Mechanics and Materials.* vol. 490, pp. 1342-1346, 2014.

[5] K. Ren, Z. Li and C. Wang, “LBDRP: A Low-bandwidth Data Replication Protocol on Journal-based Application”, *Computer Engineering and Technology (ICCET).* pp. 89 - 92, 2010.

[6] A. Noraziah, A.A.C. Fauzi, W.M.W. Mohd, T.Herawan and Z. Abdullah, “Managing MyGRANTS Fragmented Database Using Binary Vote Assignment Grid Quorum with Association Rule (BVAGQ-AR) Replication Model”, *Lecture Notes in Electrical Engineering, Springer, Singapore*, vol 520, 2019.

[7] T. Connolly, and C. Begg. *Database System A Practical Approach to Design, Implementation and Management, International Edition, Sixth Edition* : Pearson Education, 2015.

[8] A.M. Tamhankar, and S. Ram, “Database fragmentation and allocation: an integrated methodology and case study”, In *Proceeding of IEEE Transactions Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 28 (3), pp. 288 – 305, 1998.

[9] S. Bhar, and K. Barker, “Static allocation in distributed objectbase systems: a graphical approach”, *6th Int. Conf. Information Systems Data Management*, pp. 92–114, 1995.

[10] S. Navathe, S. Ceri, G. Wiederhold, and J. Dou, “Vertical Partitioning Algorithms for Database Design”, *ACM Transaction Database System*. vol. 9, pp. 680–710, 1984.

[11] D. Sacca, and G. Wiederhold, “Database partitioning in a cluster of processors”, *ACM Transaction Database System.* vol. 10, pp. 29–56, 1985.