

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : Ainul Azila Binti Che Fauzi

Date of Birth : 21st April 1985

Title : Binary Vote Assignment on Grid Quorum with Association
Rule in Distributed Database Systems

Academic Session : Sem 2 2017/2018

I declare that this thesis is classified as:

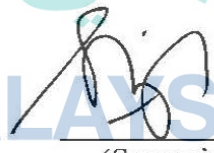
- CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)*
- RESTRICTED (Contains restricted information as specified by the organization where research was done)*
- OPEN ACCESS I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:


(Student's Signature)


(Supervisor's Signature)

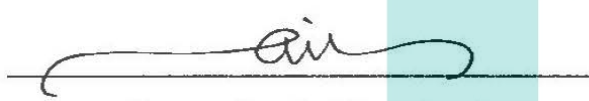
ASSOC. PROF. DR. NORAZIAH BINTI AHMAD
ASSOCIATE PROFESSOR
FACULTY OF COMPUTER SYSTEMS &
SOFTWARE ENGINEERING
UNIVERSITI MALAYSIA PAHANG
LEBUHRAYA TUN RAZAK, 26300 GAMBANG,
KUANTAN, PAHANG
TEL: 09-549 2121 FAX: 09-549 2144

850421036038
Date: 25 July 2018

Assoc. Prof. Dr. Noraziah Ahmad
Date: 25 July 2018

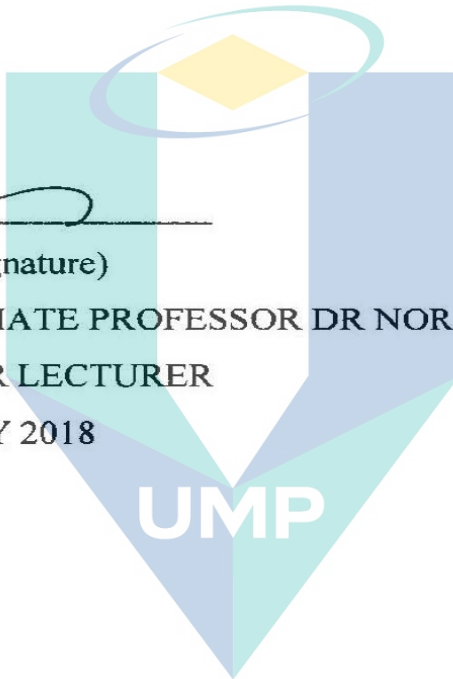
SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Doctor of Philosophy in Computer Science



(Supervisor's Signature)

Full Name : ASSOCIATE PROFESSOR DR NORAZIAH AHMAD
Position : SENIOR LECTURER
Date : 25 JULY 2018

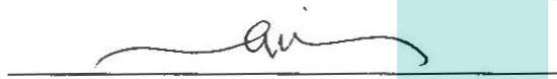


اونيورسيتي ملايسيا قهغ

UNIVERSITI MALAYSIA PAHANG

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.



(Student's Signature)

Full Name : AINUL AZILA BINTI CHE FAUZI

ID Number : PCC13001

Date : 25 JULY 2018



UMP

اونيورسيتي ملايسيا قهغ

UNIVERSITI MALAYSIA PAHANG

BINARY VOTE ASSIGNMENT ON GRID QUORUM REPLICATION
TECHNIQUE WITH ASSOCIATION RULE



AINUL AZILA CHE FAUZI

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Doctor of Philosophy

اونيورسيتي ملايسيا قهق

UNIVERSITI MALAYSIA PAHANG
UNIVERSITI MALAYSIA PAHANG

AUGUST 2018

PERMINTAAN 250219	
UNIVERSITI MALAYSIA PAHANG	
No. Pendaftaran	REKOD
120615	15001P
Tarikh	1500
16 JAN 2019	1500

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my supervisor Assoc. Prof. Dr. Noraziah Ahmad for the continuous support of my study and research, for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my study.

I express my deepest thanks to my family, especially to my father, Che Fauzi Bin Che Mohamood; my mother, Azizah Binti Yusoff; my sisters, Ainul Liyana and Ainul Auni and my brother, Ahmad Shahril Danial; for their patience and moral support throughout my life. Special thanks to my little angels, Aryan and Aryana for always brighten up my days and give me strength when I was feeling down. I couldn't do this without all of you.

I also want to thank my fellow mates in Faculty of Computer System and Software Engineering UMP for the stimulating discussions. Last but not least, I want to thank all best friends in UMP for the sleepless nights we were working together and for all the fun we have had in these few years.

The logo of Universiti Malaysia Pahang (UMP) is a large, downward-pointing arrow. The arrow is divided into four quadrants: top-left is light blue, top-right is light green, bottom-left is light purple, and bottom-right is light teal. The letters 'UMP' are written in white, bold, sans-serif font across the center of the arrow.

UMP

اونيورسيتي مليسيا قهغ

UNIVERSITI MALAYSIA PAHANG

ABSTRAK

Salah satu cabaran terbesar yang perlu dihadapi oleh pengguna pada hari ini adalah dalam penambahbaikan dalam sistem pengurusan data. Organisasi perlu menyediakan data terkini kepada pengguna yang mungkin berada pada jarak yang jauh secara geografi dan untuk mengendalikan jumlah permintaan data dibahagikan di beberapa laman dalam persekitaran teragih. Oleh itu, penyimpanan, kesediaan, dan konsistensi data adalah isu-isu penting yang perlu diberi perhatian untuk membolehkan akses data yang cekap dan selamat dari laman web yang berbeza. Salah satu cara untuk menangani cabaran ini dengan berkesan adalah dengan menggunakan teknik replikasi. Replikasi adalah teknik yang berguna untuk sistem pangkalan data teragih. Melalui teknik ini, data boleh diakses dari pelbagai lokasi. Oleh itu, ia meningkatkan ketersediaan data dan akses kepada pengguna. Apabila satu pelayan gagal untuk melakukan transaksi, pengguna masih boleh mengakses data yang sama di pelayan lain. Teknik-teknik seperti Read-One-Write-All (ROWA), Hierarchical Replication Scheme (HRS) dan Branch Replication Scheme (BRS) adalah teknik popular yang digunakan untuk replikasi dan pengurusan data. Walau bagaimanapun, teknik ini mempunyai kelemahan dari segi kos komunikasi, iaitu jumlah pelayan replikasi yang diperlukan untuk menyalin data. Selain itu, teknik-teknik ini juga tidak mempertimbangkan perkaitan antara data semasa proses pembahagian data. Pengetahuan mengenai perkaitan antara data dapat diekstrak dari data yang lepas dengan menggunakan teknik-teknik dalam bidang perlombongan data. Tanpa strategi yang tepat, replikasi boleh menyebabkan masa untuk melengkapkan satu transaksi menjadi tinggi. Dalam penyelidikan ini, Binary Vote Assignment on Grid Quorum with Association (BVAGQ-AR) dicadangkan bagi mengendalikan replikasi untuk data terbahagi dalam persekitaran pangkalan data teragih dengan menggunakan kos komunikasi dan masa memproses yang rendah untuk satu transaksi. Ciri-ciri utama BVAGQ-AR ialah teknik ini menggabungkan teknik replikasi dan teknik perlombongan data yang membolehkan pengekstrakan pengetahuan yang bermakna dari set data yang besar. Teknik BVAGQ-AR terdiri daripada langkah-langkah berikut. Langkah pertama ialah menggunakan teknik perlombongan data dengan melaksanakan algoritma Apriori dari Association Rules. Ia digunakan untuk mencari perkaitan antara data. Untuk langkah kedua, pangkalan data dipecahkan berdasarkan keputusan analisis perlombongan data. Teknik ini diuji untuk memastikan replikasi data dapat dilakukan dengan berkesan dalam masa yang sama dapat menjimatkan kos. Kemudian, pangkalan data yang dihasilkan selepas proses pembahagian data diletakkan di pelayan yang berkenaan. Akhir sekali, selepas proses perletakan data, setiap pelayan mempunyai fail pangkalan data tersendiri dan bersedia untuk menjalankan sebarang transaksi dan proses replikasi. Akhirnya, keputusan eksperimen menunjukkan bahawa BVAGQ-AR dapat memastikan konsistensi data dengan kos komunikasi dan masa pemprosesan untuk transaksi terendah berbanding dengan teknik BCSA, PRA, ROWA, HRS and BRS.

ABSTRACT

One of the biggest challenges that data grids users have to face today relates to the improvement of the data management. Organizations need to provide current data to users who may be geographically remote and to handle a volume of requests of data distributed around multiple sites in distributed environment. Therefore, the storage, availability, and consistency are important issues to be addressed to allow efficient and safe data access from many different sites. One way to effectively cope with these challenges is to rely on the replication technique. Replication is a useful technique for distributed database systems. Through this technique, a data can be accessed from multiple locations. Thus, replication increases data availability and accessibility to users. When one site fails, user still can access the same data at another site. Techniques such as Read-One-Write-All (ROWA), Hierarchical Replication Scheme (HRS) and Branch Replication Scheme (BRS) are the popular techniques being used for replication and data management. However, these techniques have its weaknesses in terms of communication costs that is the total replication servers needed to replicate the data. Furthermore, these techniques also do not consider the correlation between data during the fragmentation process. The knowledge about data correlation can be extracted from historical data using techniques of the data mining field. Without proper strategies, replication increases job execution time. In this research, the some-data-to-some-sites scheme called Binary Vote Assignment on Grid Quorum with Association (BVAGQ-AR) is proposed to manage replication for meaningful fragmented data in distributed database environment with low communication cost and processing time for a transaction. The main feature of BVAGQ-AR is that the technique integrates replication and data mining technique allowing meaningful extraction of knowledge from large data sets. Performance of the BVAGQ-AR technique comprised the following steps. First step is mining the data by using Apriori algorithm from Association Rules. It is used to discover the correlation between data. For the second step, the database is fragmented based on the data mining analysis results. This technique is executed to make sure data replication can be effectively done while saving cost. Then, the databases that are resulted after the fragmentation process are allocated at their assigned sites. Finally, after allocation process, each site has a database file and ready for any transaction and replication process. Finally, the result of the experiments shows that BVAGQ-AR can preserve the data consistency with the lowest communication cost and processing time for a transaction as compared to BCSA, PRA, ROWA, HRS and BRS.

UNIVERSITI MALAYSIA PAHANG

DECLARA IO

TITLE PAGE

ACKNOWLEDGEMENTS

iv

ABSTRAK

v

ABSTRACT

vi

TABLE OF CONTENT

vii

LIST OF TABLES

xi

LIST OF FIGURES

xii

LIST OF ABBREVIATIONS

xiii

CHAPTER 1 INTRODUCTION

1

1.1 Introduction

1

1.2 Problem Statement

3

1.3 Objectives of Research

5

1.4 Scopes of Research

5

1.5 Organization of Thesis

6

CHAPTER 2 LITERATURE REVIEW

7

2.1 Introduction

7

2.2 Theoretical Background

7

2.2.1 Data Grid

7

2.2.2 Data Mining in Grid

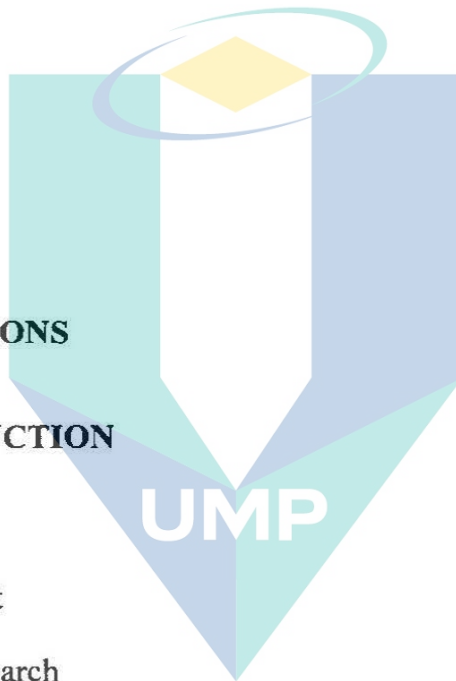
9

2.2.3 Data Mining Types

11

2.2.4 Mining Frequent Patterns in Association Rules

14



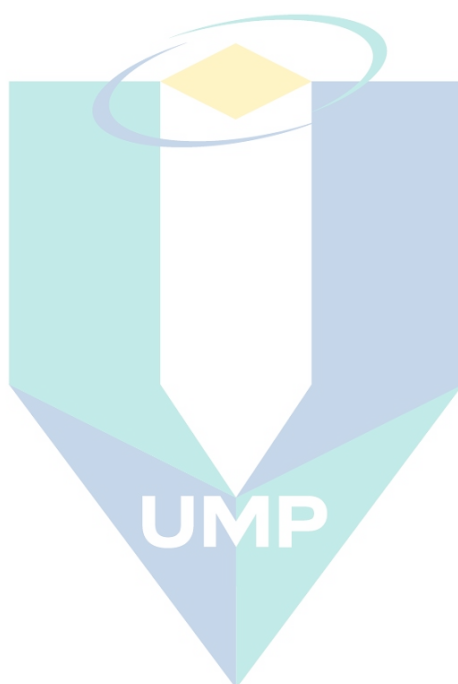
اونيورسيتي ملپسبا قهق

UNIVERSITI MALAYSIA PAHANG

2.2.5	Database Fragmentation	15
2.3	Literature Review	18
2.3.1	General Data Replication in Grid	19
2.3.2	Data Replication with Data Mining Techniques	20
2.3.3	Data Replication without Data Mining Techniques	26
2.4	Discussion	32
2.5	Summary	38
CHAPTER 3 METHODOLOGY		39
3.1	Introduction	39
3.2	BVAGQ-AR Overview	39
3.3	BVAGQ-AR Transaction Manager (BTM)	43
3.3.1	Data Mining in BVAGQ-AR	45
3.3.2	Data Replication in BVAGQ-AR	54
3.4	The Coordinating Algorithm for Primary Replica	57
3.4.1	Initial Lock	57
3.4.2	Propagate Lock	57
3.4.3	Primary Replica Processing	62
3.5	The Cooperative Algorithm for Neighbour Replica	65
3.6	Replica Management	66
3.6.1	Case 1 - A Set Of Transaction V_η Request to Update Instant b at One Site $i \in S(B)$	67
3.6.2	Case 2 - Different Sets of Transactions Where V_η and V_ψ Request Instant b at Different Site $i \in S(B)$	67
3.6.3	Case 3 - Different Set of Transactions V_η and V_ψ Request to Update Instant b at Same Site $i \in S(B)$	68

3.6.4	Case 4 - A Set of Transaction V_{η} Request to Update an Unavailable Instant c , at Site $i \in S(B)$	69
3.7	Illustration Examples	70
3.7.1	Case 1 - Database Mining Management	71
3.7.2	Case 2 - Database Replication Transaction Management	72
3.8	Correctness	75
3.9	Summary	76
CHAPTER 4 RESULTS AND DISCUSSION		77
4.1	Introduction	77
4.2	Hardware and Software Specifications	77
4.3	BVAGQ-AR Experimental Result	79
4.3.1	Experiment 1: Mining S to Identify J_I and the $S(B)^e$ for S'	81
4.3.2	Experiment 2: A Transaction V_{η} Request to Update Instant e at Site E	84
4.3.3	Experiment 3: Different Sets of Transactions, V_{η} and V_{ψ} Request Instant e at Different Site	89
4.3.4	Experiment 4: : Different Sets of Transactions, V_{η} and V_{ψ} Request Instant e at the Same Site	94
4.3.5	Experiment 5: A set of Transaction V_{η} Request to Update an Unavailable Instant, i at a Site	99
4.4	Result and Discussion	103
4.4.1	Validity Threats	103
4.4.2	Communication Cost Comparison	104
4.4.3	Replication Job Execution Time Comparison	105
4.5	Summary	108

CHAPTER 5 CONCLUSION	109
5.1 Introduction	109
5.2 Conclusion	109
5.3 Contributions to Knowledge	111
5.4 Future Work	111
REFERENCES	113



اونيورسيتي مليسيا قهغ

UNIVERSITI MALAYSIA PAHANG

LIST OF TABLES

Table 2.1	Comparative table of replication strategies with data mining	33
Table 2.2	Comparative table of replication strategies in real time environment	37
Table 3.1	Transactional data from MyGrants	47
Table 3.2	Database with binary variable	51
Table 3.3	Target set	56
Table 3.4	BVAGQ-AR Primary-Neighbours Grid Coordination	70
Table 3.5	The BVAGQ-AR PNGC for $S(B_x) = \{E, B, D, F, H\}$.	71
Table 3.6	An example of how BVAGQ-AR handle database mining	71
Table 3.7	An example of how BVAGQ-AR handle concurrent transactions	74
Table 4.1	Server component specifications	78
Table 4.2	System development tools specifications	78
Table 4.3	BVAGQ-AR Grid Coordination	80
Table 4.4	Mining S to Identify J_I and the $S(B)^e$ for S'	82
Table 4.5	Experimental result for one transaction at one site	85
Table 4.6	Experimental result for two transactions update same data at two sites	90
Table 4.7	Experimental result for two transactions at the same server	95
Table 4.8	Experimental result for transaction request to update unavailable data	100
Table 4.9	Communication cost comparison	104
Table 4.10	BVAGQ - AR Improvement in terms of communication cost (%)	105
Table 4.11	Comparison of job execution time for the minimum number of replication servers	106
Table 4.12	Comparison of job execution time for the maximum number of replication servers	107
Table 4.13	BVAGQ - AR Improvement in terms of job execution time (%)	107

LIST OF FIGURES

Figure 2.1	Horizontal fragmentation	16
Figure 2.2	Vertical fragmentation	17
Figure 2.3	Hybrid fragmentation	18
Figure 2.4	PDDRA architecture	23
Figure 2.5	The framework of ROWA-MSTS	27
Figure 2.6	All replicas in HRS update data	28
Figure 2.7	All replicas in BRS update data	29
Figure 2.8	Data Replication in BVAG	31
Figure 3.1	Replica selection strategies based on data mining techniques	40
Figure 3.2	BVAGQ-AR methodology illustration	43
Figure 3.3	Set of frequent 1-itemsets	48
Figure 3.4	Set of frequent 2-itemsets	48
Figure 3.5	Set of frequent 1-itemsets	49
Figure 3.6	Generating frequent itemsets using the Apriori algorithm	52
Figure 3.7	Examples of data replication in BVAGQ-AR	59
Figure 3.8	An assignment B for data file k where $S(B_k) = \{ E, B, D, F, H \}$	61
Figure 3.9	One set of transaction at one site	67
Figure 3.10	Two set of transactions at two sites	67
Figure 3.11	Two set of transactions at one site	68
Figure 3.12	Two set of transactions at two sites	69
Figure 3.13	An example of BVAGQ-AR transaction processing	72
Figure 4.1	Five replication servers connected to each other	80
Figure 4.2	Execution time for Experiment 1	81
Figure 4.3	Time diagram for Mining S to Identify J_l and the $S(B)^e$ for S'	83
Figure 4.4	Execution time for Experiment 2	84
Figure 4.5	Time diagram when a set of transaction V_η request to update instants at site E	88
Figure 4.6	Execution time for Experiment 3	89
Figure 4.7	Time diagram when different sets of transactions, V_η and V_ψ request instant e at different site.	93
Figure 4.8	Execution time for Experiment 4	94
Figure 4.9	Time diagram when different sets of transactions, V_η and V_ψ request instant e at the same site	98

LIST OF ABBREVIATIONS

ADW	Administrator of Data Warehouse
AUFM	Attribute Usage Frequency Matrix (AUFM)
BRS	Branch Replication Scheme
BSCA	Based on Support and Confidence Dynamic Replication Algorithm
BVAGQ-AR	Binary Vote Assignment in Grid Quorum with Association Rule
DDBMS	Distributed Database Management System
DDS	Distributed Database System
GIG	Global Information Grid
GUI	Graphical User Interface
HRS	Hierarchical Replication Scheme
IBM	International Business Machines
I/O	Input/Output
J&J	Johnson & Johnson
LAN	Local Area Network
NASA	National Aeronautics and Space Administration
NRG	Neighbour Replication on Grid
NTM	NRG Transaction Manager
PDDRA	Pre-Fetching Based Dynamic Data Replication Algorithm
PRA	Pre-Fetching Based Dynamic Replication Algorithm
ROWA	Read-One-Write-All
ROWA- MSTs	Read-One-Write-All Monitoring Synchronization Transactions Systems
RSCA	Replication Strategy Based on Clustering Analysis
RSCP	Replication Strategy Based on Maximal Frequent Correlated Pattern
WAN	Wide Area Network

CHAPTER 1

INTRODUCTION

1.1 Introduction

One of the biggest challenges that data grid users have to face today relates to the improvement of the data management. Organizations need to provide current data to users who may be geographically remote and manage a volume of requests for data distributed around multiple sites in distributed environment. Therefore, the storage of data, their availability and consistency are important issues to be addressed in order to allow efficient and safe access data distributed to and from users around many different sites. One way to effectively cope with these issues is to rely on the replication technique. The main aims of replication are to manage large volumes of data in a distributed manner, expedite data access, reduce access latency and increase data availability (Milani and Navimipour, 2016; Wang et al., 2017).

While data availability is increased because data are stored in more than one site, most of the existing replication strategies *overlook the possible correlation that may occur* among different data files in Distributed Database System (DDS). The knowledge about data correlation can be extracted from historical data using techniques of the data mining field. Data mining techniques have been proven to be a powerful tool in facilitating the extraction of meaningful knowledge from large data sets (Han et al., 2011; Zaki and Meira, 2014). In this respect, mining grid data is an interesting research field which aims at analyzing grid systems with data mining techniques in order to discover new meaningful knowledge to efficiently enhance grid systems in many areas (Sánchez et al., 2008). Nevertheless, only few works have used data mining techniques

to explore file correlations in data grid. Therefore, the present study was initiated on the basis of the paucity of published works in this area of research and its potential benefits.

Database technology has become important in most business organizations. Distributed Database System (DDS) has become more affordable and useful driven by the advances in telecommunications. A DDS normally consists of a number of separate yet interrelated databases located at different geographic sites which can communicate through a network. Usually, the system is managed by a Distributed Database Management System (DDBMS). Distributed databases reduce cost and increase performance and availability; however, the design of DDBMS is complicated. In order to make this process feasible, the database distribution process is divided into two steps, namely, fragmentation and allocation (Hossein et al., 2015).

DDS is distributed into separate partitions or fragments. Fragmentation attempts to split data into fragments, which should be allocated to sites over the network in the allocation stage. A single database needs to be divided into two or more pieces and the combination of the pieces yield the original database without any loss of information. Each piece that is produced after fragmentation is known as a database fragment. Fragmentation is very useful in terms of usage, reliability and efficiency of distributed databases. Fragmentation phase is the process of clustering the information accessed simultaneously by applications in fragments, while allocation phase refers to the process of distributing the generated fragments over the database system sites (Baiao et al., 2000, Pazos et al., 2014).

To fragment a database, two basic methods are commonly used, namely, vertical fragmentation and horizontal fragmentation. In addition to the vertical and horizontal fragmentation methods, it is also possible to execute mixed or hybrid fragmentation on a class by combining both techniques (Baiao et al., 2000, Hossein et al., 2015). In the object model, vertical fragmentation breaks the class logical structure (its attributes and methods) and distributes logical structure across the fragments, which will logically contain the same objects, but with different structures. On the other hand, horizontal fragmentation distributes class instances across the fragments, which will have exactly

the same structure but different contents. Thus, a horizontal fragment of a class contains a subset of the whole class extension (Baiao et al., 2000).

Each partition or fragment of a distributed database may be replicated (Jemal et al., 2014). Changes applied at one site are captured and stored locally before being forwarded to and applied at each of the remote locations. Additionally, expensive synchronization mechanisms are needed in order to maintain the consistency and integrity of the replicated data in distributed environments. Synchronous replication is the process of copying data over a storage area network or wide area network so there are multiple up to date copies of the data. It is primarily used for high-end transactional applications that need instant update. Synchronous replication can be categorized into several schemes, namely copying all data to all sites (full replication), all data to some sites, some data to all sites and some data to some sites. By contrast, asynchronous replication writes data to the primary storage first and then copies the data to the replica. Although the replication process may occur in near real time, replication on a scheduled basis, for example, every ten minutes, is more common.

1.2 Problem Statement

Single centralized database has low availability and reliability because if the database site goes down the whole system fails. DDS has high availability and reliability. However, DDS introduces high redundancy as more than one site is used and also creates low data consistency and data coherency as more than one replicated data need to be updated. Hence, this raises the following questions that warrant further investigation:

- i. How can high availability of data be achieved?
- ii. How can redundancy be reduced when there is an increase in the data storage capacity?
- iii. How are synchronization mechanisms executed in order to maintain the consistency of data when changes are made by transactions?

By storing multiple copies of data at several sites in the system, there would be increased data availability and accessibility to users despite site and communication failures. The mechanism is important to enable access to data whenever required. However, the storage capacity becomes an issue as multiple copies of data are replicated on different sites. In this regard, data organization tends to increase the data storage capacity. At the same time, expensive synchronization mechanisms would be needed to maintain the consistency and integrity of data when changes are made by the transactions. In addition, when replication is done without proper strategy, this process will increase the data redundancy and waste of space.

Examples of the simplest techniques that can be used are all-data-to-all sites replication scheme or simply known as, the Read-One-Write-All (ROWA) technique (Noraziah et al., 2010) and the Hierarchical Replication Scheme (HRS) (Pérez et al., 2010). Read operations on a data object are allowed to read any copy while write operations are required to write all copies of the data object. ROWA and HRS result in the imbalance of availability as well as the communication cost of read and write operations. The read operations have a high availability and low communication cost whereas the write operations have a low availability with higher communication cost. Meanwhile, voting techniques (VT) became popular because they are flexible and are easily implemented. VT have been applied to the primary cluster for managing replicated data. One weakness of these techniques is that writing an object is fairly expensive: a write quorum of copies must be larger than the majority of votes. In addition, dynamic quorum techniques have also been proposed to further increase availability in replicated databases. However, VT and dynamic quorum techniques do not address the issue of communication cost. Another technique called Tree Quorum (TQ) uses quorums that are obtained from a logical tree structure imposed on data copies. TQ has been proposed for persistent consistent distributed database commit in peer-to-peer network. Nonetheless, TQ also has some drawbacks. If majority of the copies in any level of the tree become unavailable, write operation cannot be executed. Several researchers have proposed logical structures on the set of copies in the database. To create intersecting quorums, the logical information has been deployed. Such technique that uses a logical structure, for example, the Grid Structure (GS) technique, executes operation with low communication costs while providing fault tolerance for

both read and write operations. However, this technique still requires for a larger number of copies to be made available to construct a quorum.

It should be noted that all of the replication techniques aforementioned are based on single file granularity. They are indeed confined to identify popular files based on file access patterns observed at application runtime. In many applications, data files may actually be correlated in terms of accesses and have to be considered together (Hamrouni et al., 2015). File correlations become an increasingly important consideration for performance enhancement in data grid. When the correlations between the data are not investigated efficiently, this condition will lead to high job processing time for a transaction. Without proper strategies, the execution time for the replication process will be high and the system will not work efficiently.

1.3 Research Objectives

The present research concentrates on the replication for fragmented database in distributed database environment. Binary Vote Assignment Grid Quorum with Association Rule (BVAGQ-AR) was proposed to produce higher reliability and availability in managing distributed database replication. Hence, the objectives of this research are as follows:

- i. To design and develop data replication algorithms in distributed database environment with low communication cost and processing time for a transaction.
- ii. To enhance the data consistency technique in objective 1 for synchronous replication.

1.4 Scope of Research

The scope of the current study was focused only on synchronous replication with non-failure cases. In addition, it also did not consider deletion or data replacement

during the allocation process. BVAGQ-AR Transaction Manager (BTM) was developed to integrate the replication process with transaction management in distributed database environment. The experiments were tested in five replicated servers because the maximum of replicated data objects in BVAGQ-AR technique was equal to 5.

1.5 Organization of the Thesis

This orientation of this thesis has been organized to give details of the theories, algorithms, procedures, observations, arguments, conclusions and recommendations in order to meet the objectives of the present study. Chapter 1 generally presents the background of the database replication, problem statement, objectives and scope of the research. This is followed by Chapter 2 which critically reviews the literature on database replication, database fragmentation concept, data grid concept, the data mining in grid, data mining tasks, namely, classification, clustering and association rule. In addition, details of mining frequent patterns in association rules are explained. Furthermore, data mining techniques in data replication and replication model are also reviewed which not only discuss the techniques but also include their drawbacks. Chapter 3 describes the algorithm of the proposed model while Chapter 4 explains the implementation of BVAGQ-AR algorithm and compares the performance with other replication techniques. The conclusions of the present research are summarized and presented in Chapter 5 which also includes recommendations for future research in this area.

اونيورسيتي مليسيا قهغ

UNIVERSITI MALAYSIA PAHANG

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter reviews database fragmentation, data grid, data mining in grid, data mining types and mining frequent patterns in association rules. Additionally, this chapter also examines data replication in grid and data replication techniques with and without data mining techniques. Last but not least, a discussion of the topics is presented in this chapter.

2.2 Theoretical Background

This section provides theoretical foundation of the present research. Key elements such as data grid, data mining in grid, data mining types, mining frequent patterns in association rules, and database fragmentation are discussed to shed some light on the theoretical background of the present study.

2.2.1 Data Grid

Currently, it is possible to utilize a large number of geographically distributed heterogeneous resources owned by different organizations as a result of the rapid development of networking technology and web. A wide range of developments on IT-based systems with the recent advances in cloud computing (Asghari, 2016; Chiregi, 2016; Milani, 2016; Navimipour, 2015; Ashouraie, 2015), grid computing (Navimipour, 2014; Soury, 2014), and peer-to-peer computing (Chiregi, 2016) have emerged. These technologies have facilitated data access and resource sharing (Milani, 2017; Milani,

2016). Grid Computing is defined as a type of parallel and distributed system that comprises the combination and collaborative uses of resources depending on the availability and the capability to satisfy the demands of researchers requiring a large amount of communication and computation power (Garg & Singh, 2017). The grid systems have been developed with the purpose of handling a huge amount of data sets and distributing them among several grid resources (Nagarajan & Mohamed, 2017). Jobs dependency can be based on the data storage. Placing replicas or files locally would be desirable in order to improve performance through reduction of file access time (Azari, 2018).

At present, the size of the data that need to be accessed on the data grid has already reached terabytes. Ensuring effective access to such huge and widely distributed data is a serious challenge to network and grid designers. Huge volumes of bandwidth can be consumed to transfer the file from the server to the client when a user generates a request for a file. Additionally, the latency involved could be significant if the sizes of the files involved are large (Satyha et al., 2006).

Data grid is a very suitable technique to process a large number of data (Zhou et al., 2008). Grid allows an organization to operate and manage distributed resources as a secure and flexible infrastructure because it can grow, shrink and change depending on current needs (Linesch et al., 2007). Furthermore, terabytes or petabytes volumes of data that are geographically scattered in storage resources are managed, controlled and shared in data grid environment (Beigrezaei et al., 2016; Ranganathan and Foster, 2001; Hoschek et al., 2000; Moore et al., 1999) to make sure users are able to share data which are located in different places, and other resources (Bsoul et al. 2011; Pérez et al., 2010).

Grid computing in general is derived from high-performance computing, supercomputing and cluster computing where several processors are connected through a high-speed interconnect in order to compute a mutual program (Stockinger, 2001). data grid deals with the efficient management, placement and replication of large amounts of data (Stockinger, 2001). Many data grid applications are being developed or proposed, such as DoD's Global Information Grid (GIG) for both business and military domains, NASA's Information Power Grids, GMES Health-Grid for medical services,

data grid for Federal Disaster Relief (Tu et al., 2010), Johnson & Johnson (J&J), China Grid, Amazon, Google, and eBay (Linesch et al., 2007). In addition, data grid applications also offer new and more powerful ways of working such as science portals, distributed computing for large-scale data analysis or collaborative work (Laura et al., 2009).

Dealing with data grid is not an easy job. Some issues must be considered while handling and managing the data. Grid allows users to keep a large number of replicas of data objects in terms of data management in order to allow for a high degree of availability, reliability and performance to make sure the grid meets the requirements of users and applications (Laura et al., 2009). Moreover, the size of data managed by data grid is endlessly increasing (Pérez et al., 2010).

In data grid, a large amount of bandwidth could be spent to send the file from the server to the client when a user requests a file and the delay or response time involved could be high (Charrada et al., 2010; Bsoul et al., 2011). Besides that, preserving local copies of data on each accessing site is expensive in terms of cost while storing all data in a centralized storage is unreasonable due to remote access latency (Shorfuzzaman et al., 2011; Sashi & Thamani, 2010a,b; Shorfuzzaman et al., 2010). This may turn out to be a bottleneck in the Internet during the process of accessing the files in the grid. Hence, due to the high latency of the Wide Area Network (WAN), the core issue is to design the strategy for efficient data access with significantly low time complexity in data grid research (Zhao et al., 2008). Additionally, in order to manage the data grid, there are some other several issues that must be considered. For example, failures or malicious attacks during execution, fault tolerance, and scalability of data. These issues can be solved by using the replication techniques (Naseera & Murthy, 2009; Charrada et al., 2010; Ounelli et al., 2010; Mistarihi et al., 2009; Zhou et al., 2008; Bsoul et al., 2011; Pérez et al., 2010; Shorfuzzaman et al., 2011; Shorfuzzaman et al., 2010; Zhou et al., 2010).

2.2.2 Data Mining in Grid

It has often been said that we are living in *the information age*. As a matter of fact, we are now actually living in *the data age* as terabytes or petabytes of data are

transferred into computer networks, the World Wide Web (WWW), and numerous data storage devices every day. This explosive growth of data volume emerges a result of the rapid development of powerful data collection and storage tools. Moreover, businesses around the world generate massive data sets, including sales transactions, stock trading records, product descriptions, sales promotions, company profiles and performance, and customer feedback (Han et al., 2012).

This explosively growing, widely available, and gigantic body of data makes this time is truly the data age. One of the biggest challenges that data grid users have to face today is the enhancement of the data management. In this regard, the understanding of the system features, user behaviour, or frequent patterns can help in contributing to both efficient management and better performance (Hamrouni et al., 2015). Powerful and handy tools are critically needed to discover valuable information from the tremendous volumes of data and to renovate the data into systematized knowledge. This obligation has led to the existence of data mining (Han et al., 2012).

Data mining techniques have been proven to be a powerful tool in facilitating the extraction of meaningful knowledge, hidden patterns, associations or anomalies from large data sets (Hamrouni et al., 2015). Hence, by mining historical grid data using the techniques from the data mining field, valuable knowledge can be mined. The grid systems management may able to improve in many areas using the knowledge discovered from the data mining because that knowledge can enhance data replication strategies. Replication is one way to effectively address the challenges in improving data management in data grid. Many surveys were conducted and approaches have been proposed (Hamrouni et al., 2015; Mokadem and Hameurlain, 2015; Kingsy et al., 2014; Dayyani et al., 2013, Amjad et al., 2012). Nevertheless, most of existing replication techniques are based on single file granularity. Such techniques have indeed failed to recognize popular files based on file access patterns and have not taken into account file correlations or file access patterns (Hamrouni et al., 2015). However, in many applications, data files may be correlated in terms of accesses and have to be considered together.

In some cases, knowledge about file correlations such as groups of files that are always requested together as well as file access patterns must be considered in the

replication process. Hence, the present research was driven by the idea that hidden patterns and associations discovered by data mining techniques would be used to identify groups of correlated files always requested together by users, to discover the knowledge that could be used for data fragmentation and data allocation in order to enhance the data replication techniques.

2.2.3 Data Mining Types

Due to the wide variety of data mining techniques and different types of information and forms of data presentation, it is necessary to define the limits of the applicability and relevance of certain methods according to the provided data and the objectives that need to be achieved (Vadim, 2018). Data mining can be classified into two types, namely, the predictive and the descriptive (Gullo, 2015; Fayyad et al., 1996). Predictive data mining involves the development of a useful model, particularly one that can be used to predict future behaviours. These include classification and prediction tasks. These tasks are executed by deriving some models that describe their data by a set of data objects of which class label is known in order to predict the class of objects of which class label is unknown. By contrast, descriptive data mining describes data in a clear, efficient, and effective form. An example of descriptive tasks is data characterization. The objective of descriptive data mining is to summarize the general characteristics of a target class of data and data discrimination as well as association-rule discovery (Gullo, 2015).

2.2.3.1 Classification

The classification task uses a collection of data as the input known as the training set. Each data comprise a set of attributes; one of the attributes represents the class of the data. The aim is to discover a model for the class attribute as a function of the values of the other attributes. Consequently, the model is used to predict the class attribute of the neglected original data.

For example, let us assume that there is a group of records that represent the position held by the teachers in a school and each record has the following attributes: (i) teacher's name, (ii) position, (iii) years of service in the school, and (iv) the class attribute, which is a Boolean attribute that specifies whether or not the teacher holds the position of a homeroom teacher. Additionally, by further assuming that the input collection contains the following records: ⟨Shahril, head teacher, 2, no⟩, ⟨Liyana, senior assistant, 7, yes⟩, ⟨Fauzi, senior assistant, 2, yes⟩, ⟨Auni, assistant coach, 7, yes⟩, a classification algorithm would possibly find a model as the following set of rules:

“IF position = senior assistant, OR years > 4 THEN homeroom=yes”.

Therefore, given a new record ⟨Azizah, head teacher assistant, 6, yes/no⟩, the model would predict the missing class value as ⟨yes⟩.

2.2.3.2 Clustering

The objectives of clustering is to discover a finite set groups of objects in a set of data objects (i.e., clusters) to ensure that the objects in the same cluster are alike or similar while the objects belonging to different clusters are different or dissimilar. The degrees of similarity and dissimilarity among the data objects are decided and weighed according to a proximity measure.

Clustering approaches usually define a specific objective function to be optimized to correctly define clusters that are compact and well-separated from each other. Because this theory normally causes computational problems that is hard to be optimally solved for large-scale inputs (the so-called NP-hard problems), any particular clustering technique should define the corresponding approximation algorithms to find good estimations of the optimal solution.

Past studies in the literatures had employed different clustering approaches and algorithms which differed from each other in terms of the optimization standard, the

resolution principle, and the computation of the distance between the input objects (Gullo, 2015; Aggarwal & Reddy, 2014). Nevertheless, clustering is divided into two main categories, namely, partitional and hierarchical. In general, partitional-clustering algorithms compute a single partition of the input dataset. A considerable number of partitional algorithms exploit the *relocation* scheme in which the objects are iteratively re-assigned to the clusters until a stop criterion has been met. Such scheme is used as the foundation of the popular *K-Means* algorithm (Gullo, 2015; MacQueen, 1967).

2.2.3.3 Association Rule

Given a set of records, for example, transactions, with each of them holding a number of items from a certain collection. Producing dependency rules is the objective of the association rule. These rules are used to predict the occurrence of an item based on occurrences of other items. For example, if for marketing reasons, the management of a store is keen to understand the best way to expose items to customers in order to increase purchases, an analysis of the purchasing history from the past can be made in order to find out association rules like $\{uniform, bag\} \rightarrow \{stationeries\}$, which informally show that when customers buy a uniform and a bag, it is very likely that they would also buy stationery as well. Such a rule can be used in numerous ways. For instance, uniforms and bags can be used to improve the sales of stationery such as by placing the cameras and tripods close to the stationery or putting uniforms in a promotional package that accompanies the bags.

An initial step, namely, a frequent pattern mining, is usually required by association rule algorithms to correspond to another classical data mining task (Han et al., 2007), of which the main goal is to discover the subsets of items that frequently occur together in a set of transactions. For instance, the above example association rule $\{uniform, bag\} \rightarrow \{stationery\}$ is derived from the initial discovery that uniforms, bags and stationery frequently appear together in the purchasing data log. Apriori algorithm is the most recognized association rule mining algorithm (Gullo, 2015).

2.2.4 Mining Frequent Patterns in Association Rules

Patterns or itemsets that frequently appear in a data set are called frequent patterns. For example, a set of items that appear frequently together in a transaction such as tripod and camera. This data set is called a frequent itemset. Discovery of frequent patterns plays a significant part in mining associations, data correlations, and many other exciting relationships among the data.

Frequent pattern mining in association rules seeks for recurrent relationships in a given data set. Frequent itemset mining specifies the discovery of associations and correlations between items in large transactional or relational data sets. As a result of the enormous amount of data constantly being collected and stored, many industries have been drawn to mining the patterns exhibited by their databases. The finding of interesting correlation between massive amounts of business transaction records can help the decision-making processes such as catalogue design, cross-marketing, and client behaviour analysis.

The basic concepts of data mining association rules are called support and confidence. These concepts show the practicality and certainty in data discovery rules.

Rule 1: $A \Rightarrow B$ set up in transaction D , it has support s , where P is percent of $A \cup B$ in transaction D , it is the $P(A \cup B)$ where A and B are itemsets which $A \neq B$. So support is defined as:

$$\text{support}(A \Rightarrow B) = P(A \cup B) \quad 2.1$$

Each discovery mode should be denoted by a certainty measure of its efficiency or reliability, so rule 2 is:

Rule 2: $A \Rightarrow B$ has confidence c , it is percent both A and B in transaction D . It is conditional probability $P(A | B)$, so the certainty measure confidence is defined as:

$$\text{confidence}(A \Rightarrow B) = P(A | B) \quad 2.2$$

If rule 1 and rule 2 meet the specified minimum support and confidence, then the rules are the strong association rules.

Rule 3: it is strong association rule, if $support \geq min\ support$ and $confidence \geq min\ confidence$. The min support is minimum support, and min confidence is minimum confidence.

2.2.5 Database Fragmentation

Fragmentation in distributed database is very useful because usually applications work with only some of relations rather than in entirety (Connolly, 2015). Other benefits of database fragmentation are as follows:

- i. Easy usage of data: Fragmentation makes most frequently accessed set of data near to the users. Hence, these data can be accessed easily as and when required by them.
- ii. Efficiency: Fragmentation increases the efficiency of the query by reducing the size of the table to smaller subset and making it available with less network access time.
- iii. Reliability: Fragmentation increases the reliability of fetching the data. If the users located at different locations are accessing the single database, then there will be a huge network load. This will not guarantee that correct records are fetched and returned to the users. Accessing the fragment of data in the nearest database will reduce the risk of data loss and accuracy of data.
- iv. Balanced Storage: A system usually has enormous data. These data consume huge storage spaces in servers. By using fragmentation, these data will be distributed evenly among the databases in distributed database.

There are three types of fragmentation, namely, horizontal, vertical and hybrid (Connolly, 2015). Horizontal fragments are subsets of tuples, vertical fragments are

subsets of attributes and hybrid fragments are the combination of horizontal and vertical fragments.

2.2.5.1 Horizontal Fragmentation

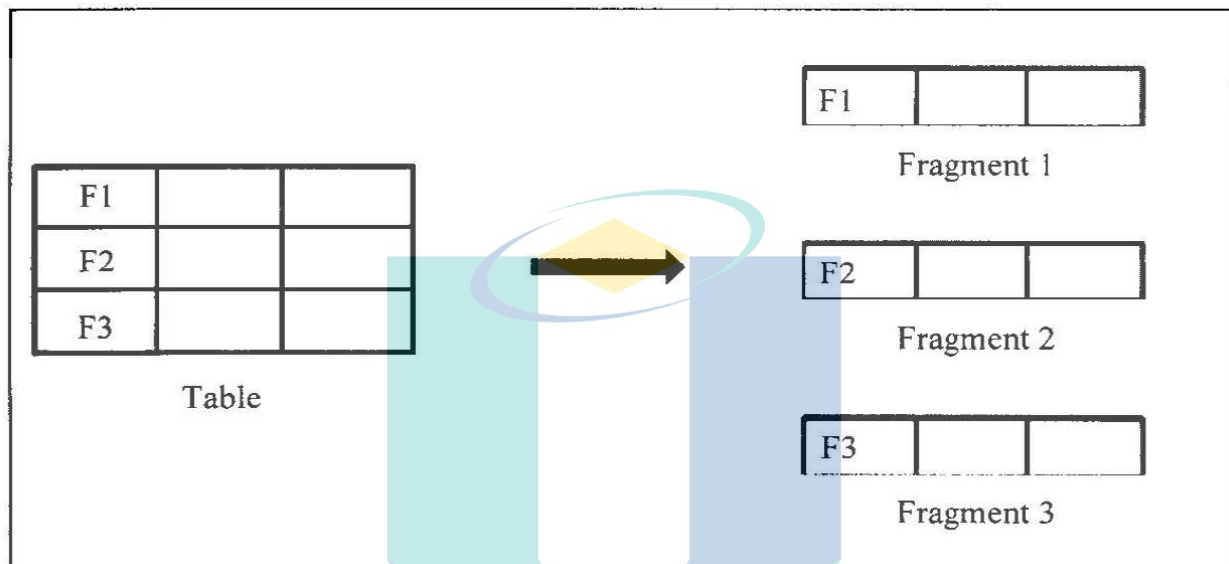


Figure 2.1 Horizontal fragmentation

Figure 2.1 illustrates horizontal fragmentation. Horizontal fragmentation groups together the tuples in a relation that are used by the important transactions (Tamhankar & Ram, 1998; Connolly, 2015). 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 the 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.

Horizontal fragmentation involves forming a subset of the global relation by selecting tuples based on the value of one or more attributes which is 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 (Tamhankar & Ram, 1998; Connolly, 2015). During horizontal fragmentation, rules must be followed to

ensure completeness, reconstruction and disjointness (Bhar & Barker, 1995; Navathe et al., 1984; Sacca et al., 1985; Connolly, 2015). Horizontal fragmentation is most effective if the locality of reference can be established by using site-based attributes for fragmentation (Tamhankar & Ram, 1998; Connolly, 2015). Horizontal fragmentation improves response time, availability and concurrency of transactions.

2.2.5.2 Vertical Fragmentation

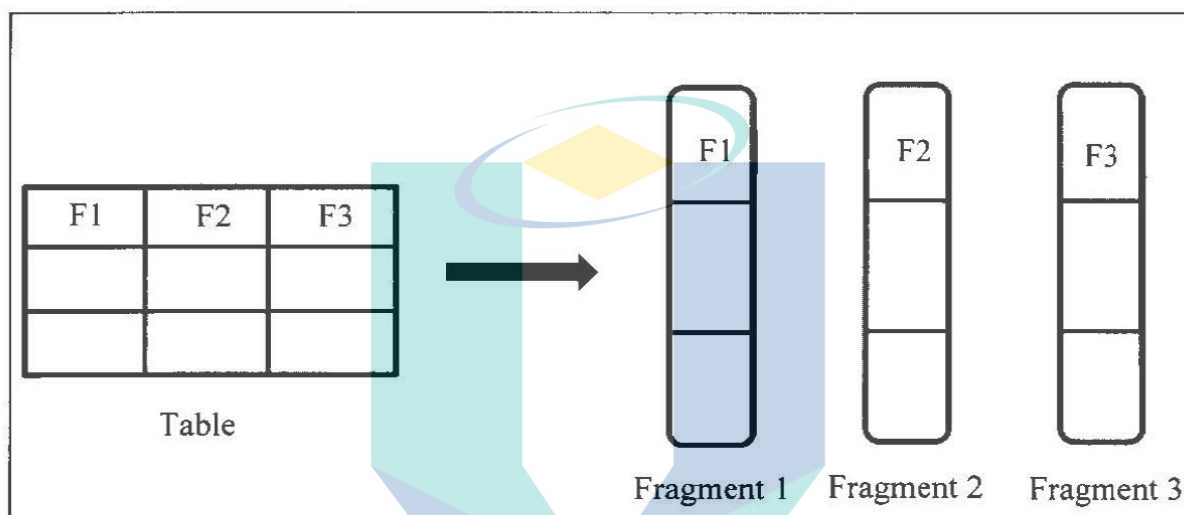


Figure 2.2 Vertical fragmentation

Figure 2.2 shows the illustration of vertical fragmentation. Vertical fragmentation groups together the attributes in a relation that are used jointly by the important transactions (Connolly, 2010). A vertical fragment is defined using the *Projection* operation of the relational algebra. Given a relation R , a vertical fragmentation is defined as $\Pi_{a_1, \dots, a_n}(R)$, where a_1, \dots, a_n are attributes of the relation R , where n is equal to the number of the attributes.

Vertical fragmentation involves dividing the attributes of a relation into groups and then projecting the relation over to each group. The attributes should not be overlapping across the groups with the exception of primary key attributes or tuple ID (Tamhankar & Ram, 1998). 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 fragmentation approaches to result in hybrid fragmentation (Tamhankar & Ram, 1998; Connolly, 2015). All the updates are executed on the master copy while other copies called snapshots are refreshed occasionally. Queries are routed to a local copy, if available. This practice 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 most suitable for networks with high delays and low reliability (Tamhankar & Ram, 1998; Connolly, 2015).

2.2.5.3 Hybrid Fragmentation

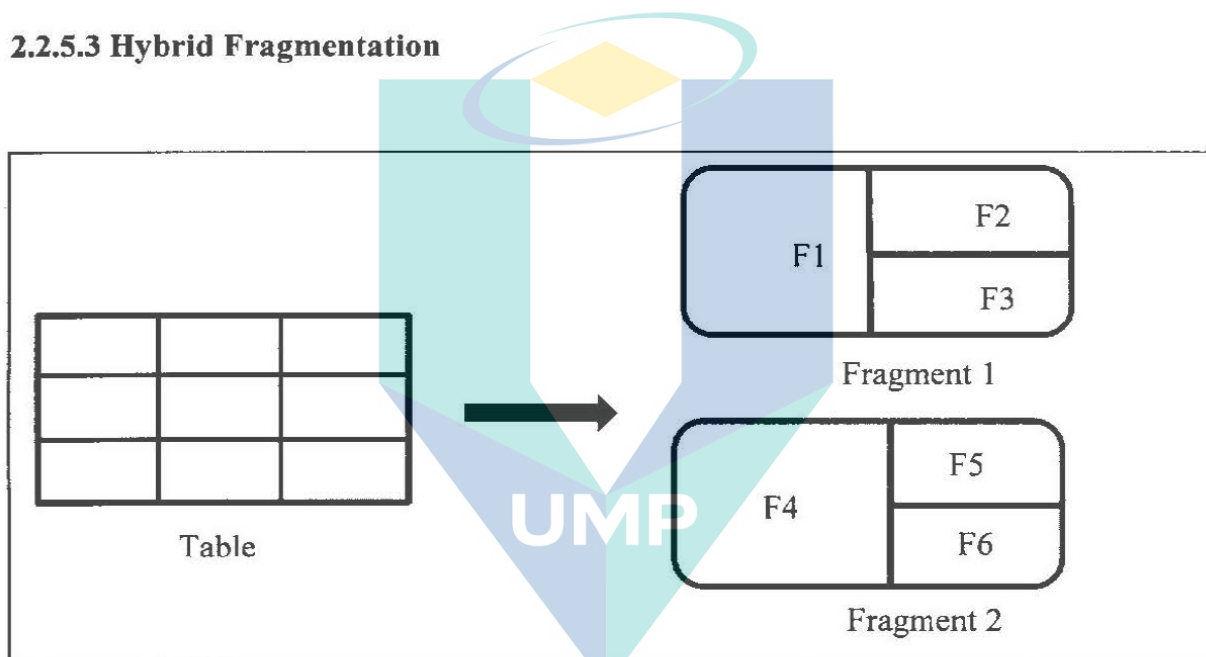


Figure 2.3 Hybrid fragmentation

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. A mixed fragment is defined using the Selection and Projection operations of the relational algebra. Given a relation R , a mixed fragment is defined as $\sigma_p(\Pi a_1, \dots, a_n(R))$ or $\Pi a_1, \dots, a_n(\sigma_p(R))$ where p is a predicate based on one or more attributes of R and a_1, \dots, a_n are attributes of R .

2.3 Literature Review

In this section, literature reviews on general data replication in grid as well as data replication with and without data mining techniques are presented.

2.3.1 General Data Replication in Grid

The general idea of replication is to store several copies of the same data in different sites across the grid. This clearly scales up the performance by reducing remote access delay and mitigating single point of failure (Boru et al., 2015). In addition, data replication helps overcoming long wide-area data transfer latencies by keeping data close to locations where queries are originated (Allcock et al., 2002). Indeed, through replication, data grid can achieve high data availability, improved bandwidth consumption, and better fault tolerance (Hamrouni et al., 2015).

There are three fundamental questions that must be answered in managing replica placement strategy in data grid (Ranganathan & Foster (2001). The three questions are

- i. When should the replicas be created?
- ii. Which files should be replicated?
- iii. Where should the replicas be placed?

Different replication strategies have been developed to overcome the problems that have been mentioned above (Charrada et al., 2010; Ounelli et al., 2010; Mistarihi et al., 2009; Zhou et al., 2008; Zhou et al., 2010; Ranganathan & Foster, 2001; Al-Mistarihi & Yong, 2008).

Depending on the answers, intensive research has been conducted on developing data replication strategies. Determining the site from which a particular data set can be retrieved most efficiently becomes a critical issue especially as data sets of interest tend to be large (Vazhkudai & Schopf, 2003). To address this issue, many replica selection strategies have been proposed. Several surveys have been conducted on data replication and replica selection strategies (Amjad et al., 2012; Grace & Manimegalai, 2014b; Ma et al., 2013; Mokadem & Hameurlain, 2015).

In replicated systems, synchronization is a one of the issues that needs to be concerned with. Synchronization schemes can be classified to be either synchronous or

asynchronous (Gray et al., 1996; Daudjee & Salem, 2004). Asynchronous replication usually updates replication using separate transaction which can cause inconsistencies. Asynchronous replication also causes the receiver to have problems in receiving the data from the sender. It works reasonably well for managing replicated data for single object updates. However, asynchronous replication fails when involving multiple objects with single update (Daudjee & Salem, 2004).

Therefore, synchronous replication is the answer for constraints that the asynchronous brought. Synchronous replication will guarantee data consistency since this replication works based on quorum to execute the operations. For any copy that has been updated, the updates are applied immediately to all the copies within the same transaction (Noraziah et al., 2009). This will ensure that all the copies in any site are the same and consistent. A consistent copy in all sites gives advantages to the organization as it provides an updated data that is accessible anytime at any place in the distributed systems environment. However, synchronous replication requires vast amounts of storage capacity as multiple copies of replicated data stored in many sites and expensive synchronization mechanism are needed to maintain the consistency of data when changes are made. As a result, a proper strategy is needed to manage the replicated data in Distributed Database System.

2.3.2 Data Replication with Data Mining Techniques

In this section, existing data replication techniques with data mining are reviewed.

2.3.2.1 Dynamic Replication Based on the Correlation of the File Strategy in Multi-Tier Data Grid Algorithm (BCSA)

The hierarchical topology provides an efficient and cost-effective technique for sharing data, computing and network resources in Dynamic Replication Based on the Correlation of the File Strategy in Multi-Tier Data Grid Algorithm (BCSA) (Cui et al., 2013). In this multi-tier model, tier-0 is called root node. Root node produces and keeps all of the data. Tier-1 is called National Data Center; tier-2 is referred as Regional Data Center and tier- n are Research Centers or universities. The nodes between tier-0 and

tier-($n-1$) are the complete computing system from the external users. It can be a single machine or a cluster. The process starts when a user in tier- n requests data. If the data is found in the middle tier, either tier-1, tier-2 or tier-($n-1$), then the middle tier will return the requested data from the layer. Otherwise, the data is returned by tier-0. The data will be copied to the middle layer in order to reduce job execution time and minimize the network load. Since the data replication can only be processed from up to down, the data storage space in each layers gradually decreases.

The root node consists of three basic parts, namely, information collecting, data mining and replication. The information collecting part collects primary data information from tier-1, as well as the data access sequence from each node, the data access number and etc. The data mining part is primarily used to analyze the data and also to discover the correlation between the data files. Replication part, as the name suggests, is in charge of the data replication. The middle layer nodes have two components called the statistic component and the replication component. Statistic component gathers access number of data and sum from the lower layer while replication component sends replicating request to the upper layer or duplicate replicas to the lower layer.

The algorithm considers spatial file locality and temporal time locality. It is based on the concept of support and confidence. It identifies the correlation between the data files through data access number and data access serial. System prefetches the frequently accessed files and their associated files, and then sends the files to the location near the access site, thereby reducing the job execution time.

In multi-tier data grid, each node belongs to a special tier. Tier(m) is the number of tier where node m is placed. If a node m and n are directly connected, and tier(m)=tier(n)-1, m is therefore the parent of n , and n is the child node. At each fixed time interval, the middle layer and the root node gather the statistical information at the next layer nodes, and total access numbers of the data file.

Root node collects information from tier-1 layer by using the information collection part, identifies the file with access numbers surpassing the threshold and its associated file by using data mining part, and then copies the files to the corresponding

node in the intermediate layer. These files are the files with access numbers surpassing the threshold and related files that meet the corresponding support and confidence.

2.3.2.2 A Prefetching-Based Replication Algorithm (PRA) in Data Grid

A transaction can request to access different data at different sites. For example, a file which has not been requested locally is likely to have been requested at another remote site earlier. Neighbouring files can be accessed at the remote site because of the spatial locality. Certainly, the remote site should have that file and its neighbouring files. That is to say, local site not only has to obtain data from the needed replica, but it also has to prefetch data from several replicas of its neighbouring files to improve the response time of file requests in future (Tian et al., 2008).

In Prefetching-Based Replication Algorithm (PRA), when a local site receives a file request but the file is not stored locally, it will search a remote site to transfer the required file replica through the Replica Directory Server. The remote site accepts a file transfer request from the local site, and transfers the file to that local site. Simultaneously, the remote site will attempt to search the adjacent files of the requested file from its file access sequence database. A message containing the list of adjacent files will be sent to the local site too. Finally, the local site will select some adjacent files to start the replication process. This process will greatly reduce the response time of file access.

However, the sequence databases need some storage space mainly because as time passes by, the size of the databases becomes larger. In order to deal with this issue, the old sequences of database can be deleted periodically. Moreover, sequences can also be compressed to reduce storage space they occupy without losing any information.

2.3.2.3 Prefetching-Based Dynamic Data Replication Algorithm (PDDRA)

Prefetching-Based Dynamic Data Replication Algorithm (PDDRA) is proposed based on pre-fetching technique. In order to increase system performance as well as decrease response time and bandwidth consumption, it is better for requester grid site to prefetch some replicas. These replicas have high probability to be requested in the near

future. After pre-fetch, when the grid site requests the replicas, it can access them locally. Hence, this decreases the access latency and response time (Saadat et al., 2012). Figure 2.4 shows a PDDRA architecture.

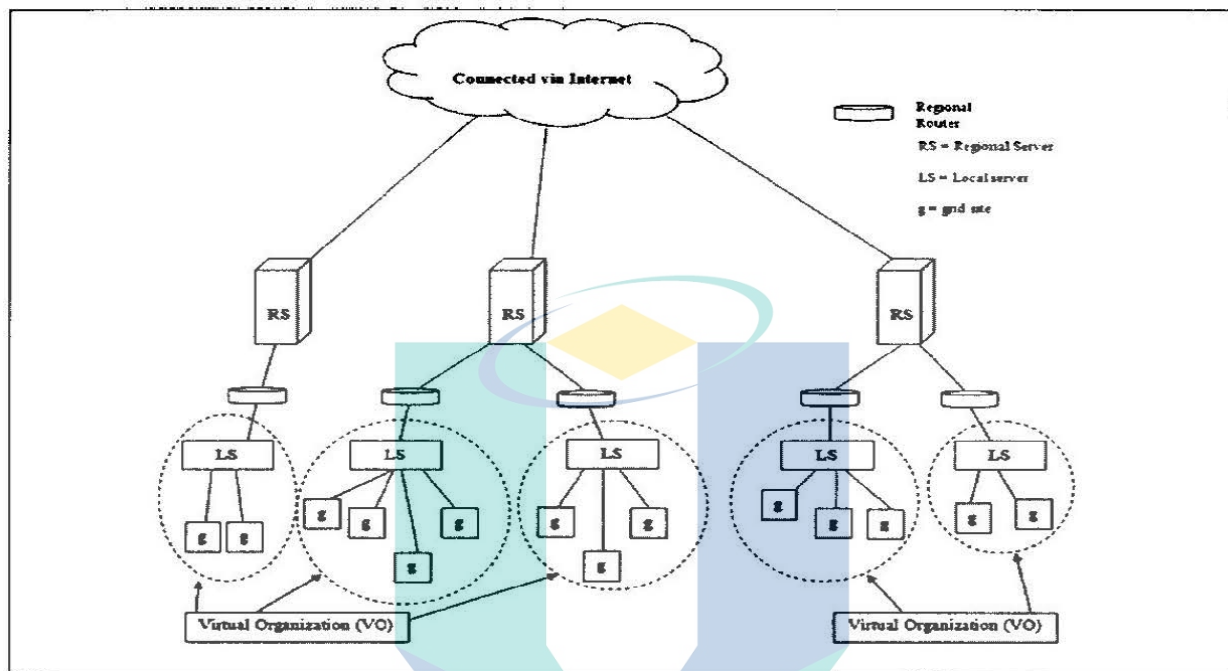


Figure 2.4 PDDRA architecture

Source: Saadat et al. (2012)

The grid sites are located at the lowest level of the proposed architecture as illustrated in Figure 2.4. These grid sites contain Computing and/or Storage Element. Virtual Organization (VO) is constituted from multiple grid sites and there is a Local Server (LS) for every Virtual Organization (VO). There is a Replica Catalogue (RC) that is located at Local Server. The bandwidth available between the sites inside a VO is higher than the bandwidth between Virtual Organizations. For that reason, accessing a file that is located within the current VO is faster than accessing the file that is placed in the other VO. In the upper layer, there is a Regional Server (RS) and each RS consists of one or more VOs. Transferring files between Regional Servers takes a long time because they are connected via internet. There is also a Replica Catalogue placed at each RS which contains a directory of all the files stored in that area. Thus, every time a file that is not stored in the current VO is requested, the RC in an RS is enquired to determine which VOs hold the required file.

Assuming that grid site 'G' requests a file that is not stored local, the RC is therefore asked to dictate which sites have the requested file. It is better to pre-fetch replicas that are probable to be requested by the requester grid site in the near future to make sure reducing access latency, response time and bandwidth consumption. A new algorithm and topology for this purpose would thus be proposed in the present research. If the requested file is not in the current VO and is stored in some other VOs, a request is sent to the RS. Then RS searches on its RC table and determines the locations of the requested file in other VOs. In this situation, only the requested file will be replicated. Pre-fetching will not be necessary and will not be done in order to prevent high propagation delay time and consequently high replication cost because of low bandwidth between VOs. Moreover, the present research also assumed that the members in a VO would have similar interests of files. Hence, the file access patterns of different VOs should not be prefetched for the requester grid site in other VOs, because their requirements and access patterns are dissimilar. The proposed algorithm would be built on the foundation of an assumption that members in a VO would have the same interest in files. In order to predict the upcoming accesses, past sequence of accesses have to be stored. By using data mining, files that will be accessed in the near future can be predicted from the past file access patterns. There are three phases in PDDRA. The first phase refers to storing file access patterns where file access sequences and data access patterns are stored in a database. The second one is the phase during which a file is requested, and replication and prefetching are performed. During this phase, a grid site requests for a file and replication process is executed for it, if necessary. Adjacent files of the required file are also prefetched for the requester grid site during this phase. Finally, the third phase is called the replacement phase. If there are enough spaces in storage element for storing a new replica, the files will be stored, or else an existing file will be selected for replacement.

2.3.2.4 Replication Strategy based on Maximal Frequent Correlated Pattern (RSCP)

Replication Strategy Based on Maximal Frequent Correlated Pattern (RSCP) scheme gathers files according to a relationship of concurrent accesses between files by jobs and stores correlated files at the same site. In order to discover these correlations, a maximal frequent correlated pattern mining algorithm of the data mining field is

introduced (Slimani et al., 2014). The all-confidence is selected as a correlation measure. The proposed scheme to be performed would be as follows:

- i. The file accesses history extraction. In this phase, each site keeps track of its access histories for all local files and remote files accessed by the jobs executed on it.
- ii. The file accesses history conversion. These file is converted into an extraction context (logical file access history). Extraction context is a table consisting of Boolean values. The accessed files are considered as items while each of the requested file for each job is regarded as one transaction.
- iii. Mining the maximal frequent correlated patterns. A new maximal frequent is applied to the correlated pattern mining algorithm in order to discover the hidden correlations between files or in other terms the groups of closely related files.
- iv. The inputs of the replication algorithm are the maximal frequent patterns that have been identified in the previous stage. For each group of correlated files to be replicated, the replication of these correlated files will take place only if there is enough storage space to hold all the files in the group. Otherwise, replacement process will be executed. In this process, the candidate files for deletion are selected according to their weight files. If the weight of the replicated group is greater than the total weights of candidate files, then they are replaced by the replicated group. Otherwise, no replication process would occur.

2.3.2.5 Replication Strategy based on Clustering Analysis (RSCA)

Replication Strategy based on Clustering Analysis (RSCA) is a strategy that identifies the correlation of client nodes' access files through clustering analysis. Afterward, RSCA creates the data files replica based on those sets, which accomplishes the objective of prefetching and buffering data. Confirming the correlation among the data files according to the access history of users is the goal RSCA. This strategy does not only assure the movement of files based on the number of access requests but also based on the correlation of the files accessed (Liu et al., 2009). For instance, when a user is requesting a data from the department files, that user will also access some other

files correlated to the department files. Similarly, other users often do the same things. However, there are many other files which also belong to the department files. Hence, the number of requests made by some users to retrieve a single file is not always above the threshold assumed beforehand. In actual fact, the user is interested in not a single department file but all the department files and the user expects to access these files quickly. RSCA will extract the characteristic of accessed data based on the access history of users, and then evaluate the correlation of the data. Then, the correlative file class sets of users by using the clustering analysis is produced. Finally, the correlation files to the data nodes is replicated. Clearly, the correlative files are numerous and the file names of them are not the same.

2.3.3 Data Replication without Data Mining Techniques

This section reviews several replication techniques which include Read-One-Write-All Monitoring Synchronization Transactions Systems (ROWA-MSTS), Hierarchical Replication Scheme (HRS), Branch Replication Scheme (BRS) and Binary Vote Assignment Grid.

2.3.3.1 Read-One-Write-All Monitoring Synchronization Transaction System (ROWA-MSTS)

Read-One-Write-All Monitoring Synchronization Transactions Systems (ROWA-MSTS) have been developed based on the ROWA technique. The ROWA-MSTS techniques handle each site regardless of whether or not the sites are operational and also to communicate with each other. The research used VSFTPD (GPL licensed FTP server for UNIX systems) as an agent communication between replicated servers (Noraziah et al., 2010). In ROWA-MSTS techniques, replicas consistencies are guaranteed by the consistency of execution on one replica; however, the client replicas are only updated and cannot provide accurate responses to queries. Synchronous replication techniques guarantee that all replicas are maintained consistently at all times by executing each transaction locally only after all replicas have agreed on the execution order. Hence, a very strict level of consistency is

maintained. Figure 2.5 shows the framework of ROWA-MSTS in distributed environment.

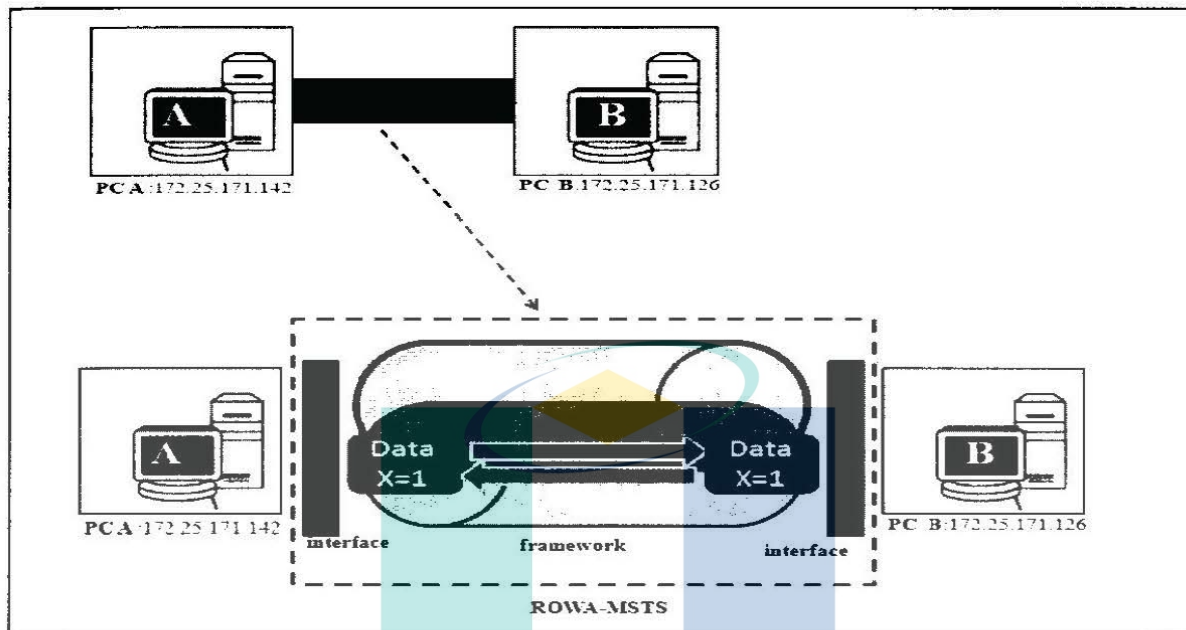


Figure 2.5 The framework of ROWA-MSTS

Source: Noraziah et al. (2010)

However, this technique practices all-data-to-all-sites replication protocol (i.e., all servers will have the same data). There will be huge amounts of data redundancy and waste of space. In addition, the execution time for a transaction will be high because the primary server has to wait for all other neighbouring servers to proceed with the transaction.

2.3.3.2 Hierarchical Replication Scheme (HRS) Protocol

Hierarchical Replication Scheme (HRS) consists of a root database server and one or more database servers organized into a hierarchy topology (Pérez et al., 2010).

Figure 2.6 shows all replicas in HRS update data.

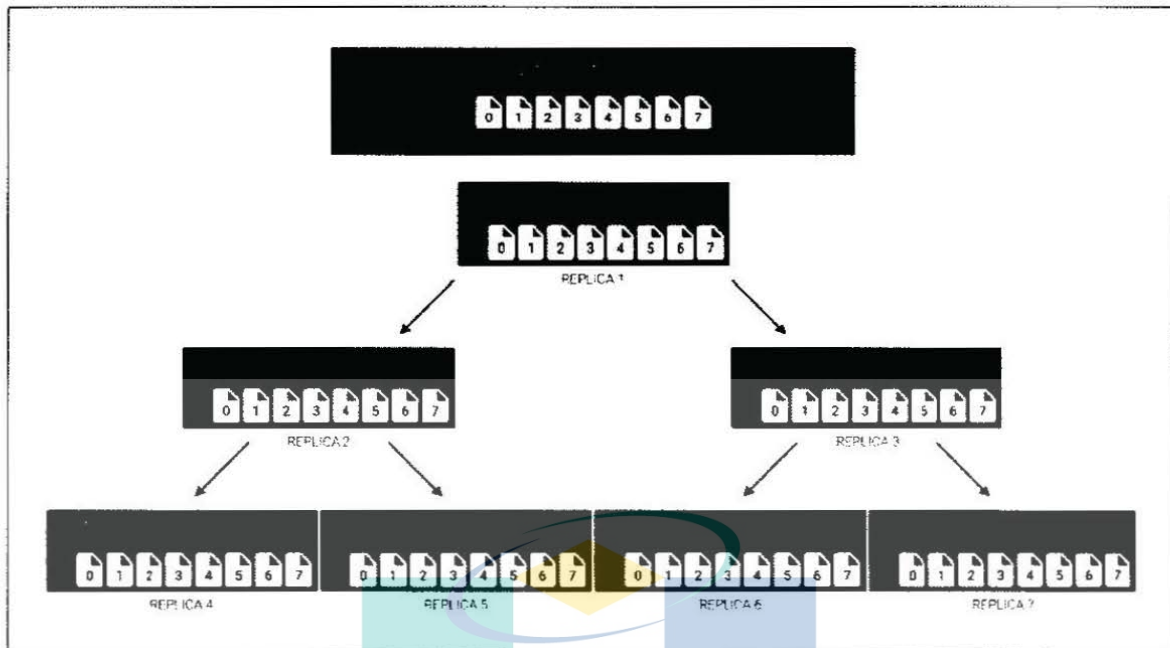


Figure 2.6 All replicas in HRS update data

As shown by the architecture of HRS in Figure 2.6, replication process starts when a transaction initiates at any block at site 1. The root replica grows in a branching way, where the master replica is replicated into several other replicas. By using this protocol, all update operations are conducted on a master replica, and then the modifications are propagated to all replicas. Once the changes have been made, all the data will be replicated into the entire replicas. Finally, all sites will have all the same data. In order to maintain consistency among the updates by clients, all blocks are propagated and locked during the transaction process. This means that only one client can modify the data at a time.

2.3.3.3 Branch Replication Scheme (BRS) Protocol

Branch Replication Scheme (BRS) goals are to increase the scalability, performance, and fault tolerance. In BRS, each replica is composed of a different set of subreplicas organized using a hierarchical topology. BRS uses parallel I/O techniques (Cortes et al., 2002; Carballeira et al., 2007) to increase the scalability and performance of the system for read and write operations. The main features of BRS are root replica, parallel replication, fine grain replication, partial replication of popular file fragments and parallel data access better resource usage. This technique needs low space per storage to support replica. Figure 2.7 shows data replication in BRS.

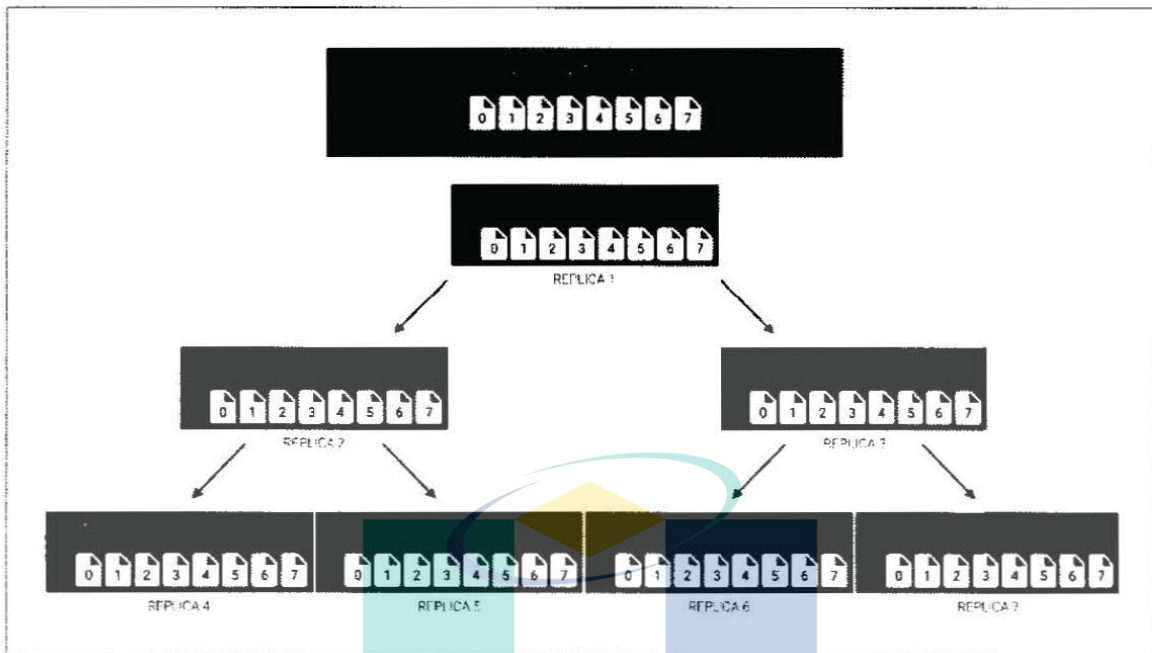


Figure 2.7 Data Replication in BRS

In order to support BRS, the following metadata information for each replica logical name must be linked:

- FR: Parent replica or subreplica (upper level). Root replica parent is itself.
- CR: Set of children subreplicas. It includes the location of the files that support the subreplicas and the portion of data replicated in each of them. Terminal subreplicas' children are themselves.
- BR: Set of subreplicas, usually at the same level, with a common upper level.

In BRS, replicas are created as close as possible to the clients that request the data files. The root replica grows toward the clients in a branching way, where the replicas will be broken into several subreplicas. By using this approach, the growing of replica tree is driven by client needs. This means that a replica is expanded or attracted towards the clients (Pérez et al., 2010).

In addition, replication does not have to be for the entire replica. Subreplicas can be also replicated following the previous conditions. Assuming that accesses to a file are not uniformly distributed, the subreplica R_i storage node would then become

overloaded. BRS can replicate only this subreplica to discharge this node. Consequently, the growth of the replication tree might not be symmetric and different branches could have different depths (Pérez et al., 2010).

In order to maintain consistency among the updates by clients, a mechanism is proposed where clients can only modify the data located in the terminal replica, or referred as the leaf nodes of the replication tree. Thus, the location of the replica is reduced to the location of the deepest subreplicas that support the range of data requested by the application. Data update is performed bottom-up, from the children replicas to the parent until it reaches the root replica. Only updated blocks are propagated. As shown by the example in Figure 2.10, assuming that block 3 of replica 2 (located in SITE 5) is written, the consistency algorithm sends block 3 to the replica's parent (SITE 2), which sends block 3 again to its parent (SITE 1). As the replica in SITE 1 is the root, the algorithm stops. Thus, replica updating can be executed by minimizing the number of steps (3) and the amount of information sent (only 1 block in this example). The amount of data transferred would be a minimum of 8 blocks.

A problem may occur when a client tries to write in a subreplica which is not terminal. This is because such subreplica has been split into other subreplica. In this case, the error message “*write not allowed*” is sent to the client. This may only happen because the client opens the file in the read-only mode. Thus, the client has to open the file for writing or updating and search for the replica that contains the data range needed by the client. Besides that, like HRS, the drawback in BRS is also because it requires many servers.

2.3.3.4 Binary Vote Assignment Grid (BVAG)

Binary Vote Assignment Grid (BVAG) is treading a new path in replication that helps to maximize the write availability with low communication cost due to the minimum number of quorum size required. In addition, the replication is combined with transaction technique.

In BVAG, all sites are logically organized in the form of two-dimensional grid structure. For example, if a BVAG consists of nine sites, it will logically organize in the

form of 3 x 3 grid structure. Each site has a premier data file. A site is either operational or fails to operate and the state of each site is statistically independent from one another. When a site is operational, the copy at the site will be available; otherwise, the copy at the site will not be available.

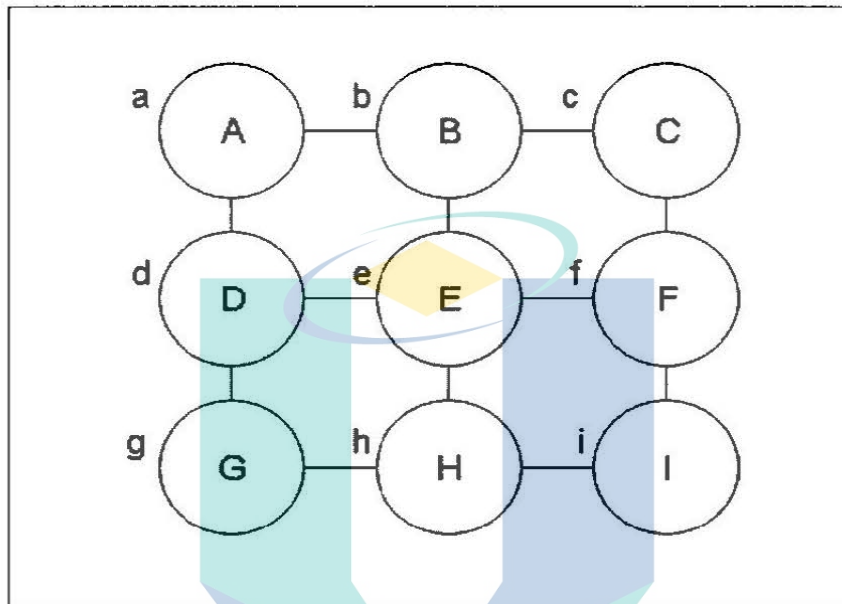


Figure 2.8 Data Replication in BVAG

Data will be replicated to the neighbouring sites from the primary site. As shown in Figure 2.8, site A is a neighbour to site B, if A is logically-located adjacent to B. Hence, the total number of data replication, $d \leq 5$. For simplicity, the primary site of any data file and its neighbours are assigned with vote one and vote zero otherwise. This vote assignment is called binary vote assignment on grid. A neighbour binary vote grid assignment on grid, B, is a function such that $B(i) \in \{0, 1\}$, $1 \leq i \leq n$ where $B(i)$ is the vote assign to site i . This assignment is treated as an allocation of replicated copies and a vote assigned to the site results in a copy allocated at the neighbour. That is, 1 vote \equiv 1 copy.

Time-efficiency of transaction processing is measured by how fast the final decision of the transaction (i.e., either to commit or to abort) has been reached. The faster the decision is reached, the higher the time efficiency will be (Deris et al., 2003). Timeliness in synchronization has not only been impressive in maximizing the usage of the system but has also contributed to consistent and reliable computing at the same time. BVAG resolves this challenge by alleviating lock with small quorum size before

capturing update and committing transaction synchronously to the database that requires the same update data. The transaction semantic shows that the system preserves the data consistency through the synchronization approach. Furthermore, it guarantees the consistency because the transaction execution is equivalent to one-copy-serializability.

2.4 Discussion

In this section, the replication strategies with and without data mining techniques discussed in the previous sections will be thoroughly examined. Table 2.1 presents the summary of comparative analysis of these techniques.

The PRA strategies predict the future access behaviour of the data. Prediction is done according to the past file access sequences of the grid sites. The drawback of this technique is that the data mining technique only focuses on sites requests without considering their contexts such as the file usage specifically (i.e., which file is requested by which job). In addition, the PRA strategy does not distinguish the different requests arriving from the different grid sites and consider them as successive file accesses. It also uses unlimited data storage and copies all data to all sites which will cause high data redundancy and communication cost.

اونيورسيتي مليسيا قهغ

UNIVERSITI MALAYSIA PAHANG

Table 2.1 Comparative table of replication strategies with data mining

Replication Techniques	Data Mining Techniques	C/D Replication	Replication method	Data storage	Data copied	Drawbacks
PRA	Frequent Sequence Mining	Decentralized	Synchronous – tested using simulator	Unlimited	All data to all sites	<ul style="list-style-type: none"> • Predict the future access behaviour of data. • Only focuses on sites requests • Does not separate the different requests arriving from different grid sites.
PDDRA	Tree Mining	Decentralized	Synchronous	Limited	Some data to some sites	<ul style="list-style-type: none"> • RS are connected via internet - transferring files between them takes long time • Do not address the issues of communication cost in the research. • The last phase is the replacement phase – effect the data availability and reliability
BSCA	Association Rules	Decentralized	Asynchronous	Limited	Some data to some sites	<ul style="list-style-type: none"> • Replication only be done from up to down. • Replication only be executed during the collecting components process – asynchronous replication. • Not suitable for systems with critical data.
RSCA	Clustering	Centralized	Asynchronous	Limited	Some data to some sites	<ul style="list-style-type: none"> • The file groups determined by the clustering method are disjoint - a file can only belong to a single equivalence class.
RSCP	Maximal frequent correlated pattern mining	Decentralized	Asynchronous	Limited	Some data to some sites	<ul style="list-style-type: none"> • When there is not enough storage space, data replacement process will take place – effect the data availability and reliability.

In PDDRA, when a transaction requests a file that is not stored locally, a request will be sent to its Regional Server (RS). Then, RS will search in its Research Catalogue (RC). After that, the replication process can proceed. Each time a transaction requests to update an unavailable data, it has to go through this process. The drawback of this technique is, as stated by the researcher, RS is connected via internet, transferring files between them therefore takes a long time. PDDRA also does not address the issue of communication cost in the research. In addition, the final phase in this technique is replacement phase in which the old files will be replaced with new ones if there is not enough storage space available for data replication. This will affect the data availability and system reliability because some data have been deleted.

The BSCA strategy applies association rules in its replication strategies. Association rules are used to find the correlations between the data. This technique will improve the average response time for the transactions. However, BSCA replication process only can be done top-down. The data replication will only be done during the process of collecting the components. Hence, this technique does not apply synchronous replication. Thus, the technique is not suitable for a system with critical data. This is because the value of data in the nodes is not synchronized. In other words, the value of data at different nodes will not be the same synchronously.

The RSCA strategy is based on the clustering data mining technique. A major drawback of clustering is that the file groups determined by the clustering technique are disjoint. This means that a file can only belong to a single equivalence class (i.e., a single cluster). This assumption is actually not always valid since multiple correlations between files can arise, for example, a file may be simultaneously correlated with more than one group of files.

The RSCP uses association rules to find the correlations between data. However, in this technique, if there is not enough space for data replication, a replacement process will be carried out. During replacement process, two average weights are calculated, namely, one for the group of files to replicate, and the other for the selected group for deletion. If the former is greater than the latter, candidate files are replaced by the new correlated group of files. Hence, this technique would not be compared with the proposed technique because the present research would not consider

file deletion. The present research has made an assumption that all data would be important and deleting it would affect the data availability.

Subject to the data used in the data mining process and in accordance with the adopted technique, two types of data will be involved:

- i. For prediction data mining techniques, the data used are values of the predictor variables as well as the target variable. Prediction of future behaviour and pre-fetching based replication strategies. In this case, when a grid site or user requests for a file, these strategies try to predict the subsequent file requests by applying data mining in the historical access data. Therefore, predicted file requests are replicated in advance before they are actually being requested. The main advantage gained is to minimize the significant delay caused by techniques of replication on request (i.e., replication done after a request is arrived). However, in spite of improvements in accuracy, predictive pre-fetching has problems, specifically in terms of incorrect predictions (pre-fetched files that are not needed) which can have a direct negative impact on the overall system performance.
- ii. For the association rule mining and the sequential pattern mining, there is a bi-dimensional extraction context which is composed either by sites in lines and accessed files in columns or jobs in lines and accessed files in columns. Actually, jobs executed in the sites may access the files. Therefore, considering a history of file access jobs is reasonable choice to infer semantic relationships between files.

UNIVERSITI MALAYSIA PAHANG

In all the techniques that have been thoroughly discussed, the data replication part has only been tested using simulation; it has not been done in real time environment. Much of the affirmative work has so far resorted to stimulation. Although the results have been useful in giving insights into the practical ways of achieving data replication with data mining, simulation approach may not give accurate representation of the actual case. Such tests would require an external validity hypothesis, which may or may not be true, depending on the circumstances.

For instance, if the Battle of Waterloo is simulated using tiny soldier and horse miniatures, then the hypothetical assumption would be the speed of the horse miniatures which stands approximately in the same relation with the speed of the soldier miniatures on the map as the speed of infantry units mean should match that of the speed of cavalry units in the year 1815. Simulation model can be investigated only under this hypothetical condition. However, what would have happened if Napoleon had chosen a different strategy? Even though the results could arguably still be considered as valid, the accuracy of such results may be questionable. There must be some differences with real time events because in real time events, other numerous constraints and issues would arise. Hence, the effectiveness of these techniques cannot be confirmed because of various difficulties and problems which do not exist in simulation may occur during the experiments in real time environment.

A real time experiment can provide more evidence to validate a theoretical model, if the theory makes some contestable assumption about some component of the target system, and if the experiment includes the real component. Thus, both real time experiments and simulations are knowledge-producing research. However, the knowledge needed to run a good simulation is not quite the same as the one needed to run a good experiment. When reproducing a real world system in the laboratory, the relationships describing the behaviour of both systems may not be known in advance. In real time experiments, researchers have to make sure that the cases are executed in devices that resemble as closely as possible those of the target system. In addition, researchers must also ensure that the components of the 'mimetic' device are assembled just like those of the target system, and that there is no interference. Therefore, a more thorough study of data replication techniques in real time has been done and its summary is presented in Table 2.2.

Two replication techniques, namely, Read-One-Write-All (ROWA) and Hierarchical Replication Scheme (HRS), copy all data to all sites. This means that all servers will have the same data. Data reliability and data availability are confirmed; however, there will be arising issues such as high data redundancy, waste of storage space and high processing time for a transaction because of the commitment of the transaction at all servers.

In Branch Replication Scheme (BRS), not all data are replicated to all sites. The data will be fragmented and stored at other replication servers. BRS is a better technique compared to ROWA and HRS in terms of communication cost and job processing time. Binary Vote Assignment Grid (BVAG) also copies some data to some sites. This technique requires the lowest number of replication servers which is more than or equal to 5. Hence, it has the lowest communication cost compared to BRS, HRS and ROWA. However, in BVAG, there is no fragmentation involved despite the fact that the degree of concurrency or parallelism in the system can be increased if fragmentation is involved.

Table 2.2 Comparative table of replication strategies in real time environment

Replication Techniques	C/D Replication	Replication method	Data storage	Data copied	Drawbacks
ROWA	Decentralized	Synchronous	Unlimited	All data to all sites	<ul style="list-style-type: none"> • High job execution time. • High communication cost.
HRS	Decentralized	Synchronous	Unlimited	All data to all sites	<ul style="list-style-type: none"> • High job execution time. • High communication cost.
BRS	Decentralized	Synchronous	Limited	Some data to some sites	<ul style="list-style-type: none"> • High job execution time. • High communication cost.
BVAG	Decentralized	Synchronous	Limited	Some data to some sites	<ul style="list-style-type: none"> • Low job execution time. • Low communication cost.

However, all these four techniques do not take data correlations into consideration. They allocate all data randomly at the servers. This will result in high job processing time. It is evident from the summary of the analysis presented that the existing replication techniques still needs improvements in order to deliver an

efficiency system. Moreover, existing replication techniques namely ROWA and HRS, synchronously replicate all data to all sites. Even though data availability in synchronous replication system is guaranteed, there are still some issues that need to be addressed such as job processing time and communication cost. In the present research, Binary Vote Assignment on Grid Quorum with Association Rule (BVAGQ-AR) has been proposed to overcome these issues. BVAGQ-AR is a replication technique that combines with data mining technique which has been tested in real time environment.

2.5 Summary

This chapter reviews the database replication, database fragmentation as well as data grid. A thorough analysis of data mining, data mining tasks and mining frequent itemsets in association rules has also been made. In addition, this chapter also discusses data replication techniques such as BVAG, ROWA, BRS and HRS. Besides replication, a review on data mining in data replication techniques such as PRA, PDDRA, RSCA, RSCP and BSCA has also been presented in this chapter. Finally, a summary of comparative analysis of all the techniques is provided. It has been evident from the review of related literatures that the current replication techniques would need some enhancements to ensure the system's efficiency. Hence, the present research has proposed a replication technique called BVAGQ-AR. In this techniques, database replication technique is integrated with fragmentation data mining to manage a transaction. The proposed technique will be described in the next chapter.

اونيورسيتي ملايسيا قهغ

UNIVERSITI MALAYSIA PAHANG

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter describes the methodology of the present research. In particular, this chapter presents detailed descriptions of the proposed replication technique employed in the present research, namely, Binary Vote Assignment on Grid Quorum with Association Rule (BVAGQ-AR). In addition, it also presents a step-by-step guidance for replica selection strategies based on data mining techniques. Last but not least, this chapter provides the example cases which would be expected to happen during the transactions in BVAGQ-AR.

3.2 BVAGQ-AR Overview

Binary Vote Assignment on Grid Quorum with Association Rule (BVAGQ-AR) is a replication technique based on data mining techniques. The main idea of this technique is to analyze operational and historical data that can be obtained from grid system in order to efficiently discover new meaningful knowledge which is the correlation between the data. The knowledge discovered would enable improvement of the management of the grid system in terms of data transfer. Hamrouni et al. (2015) have proposed the replica selection strategies based on data mining techniques. By following these strategies, BVAGQ-AR has been proposed. These selection strategies have the following four main steps: centralization, number of provider, target objectives and data mining technique. The taxonomy of replica selection strategies based on data mining techniques is illustrated in Figure 3.1.

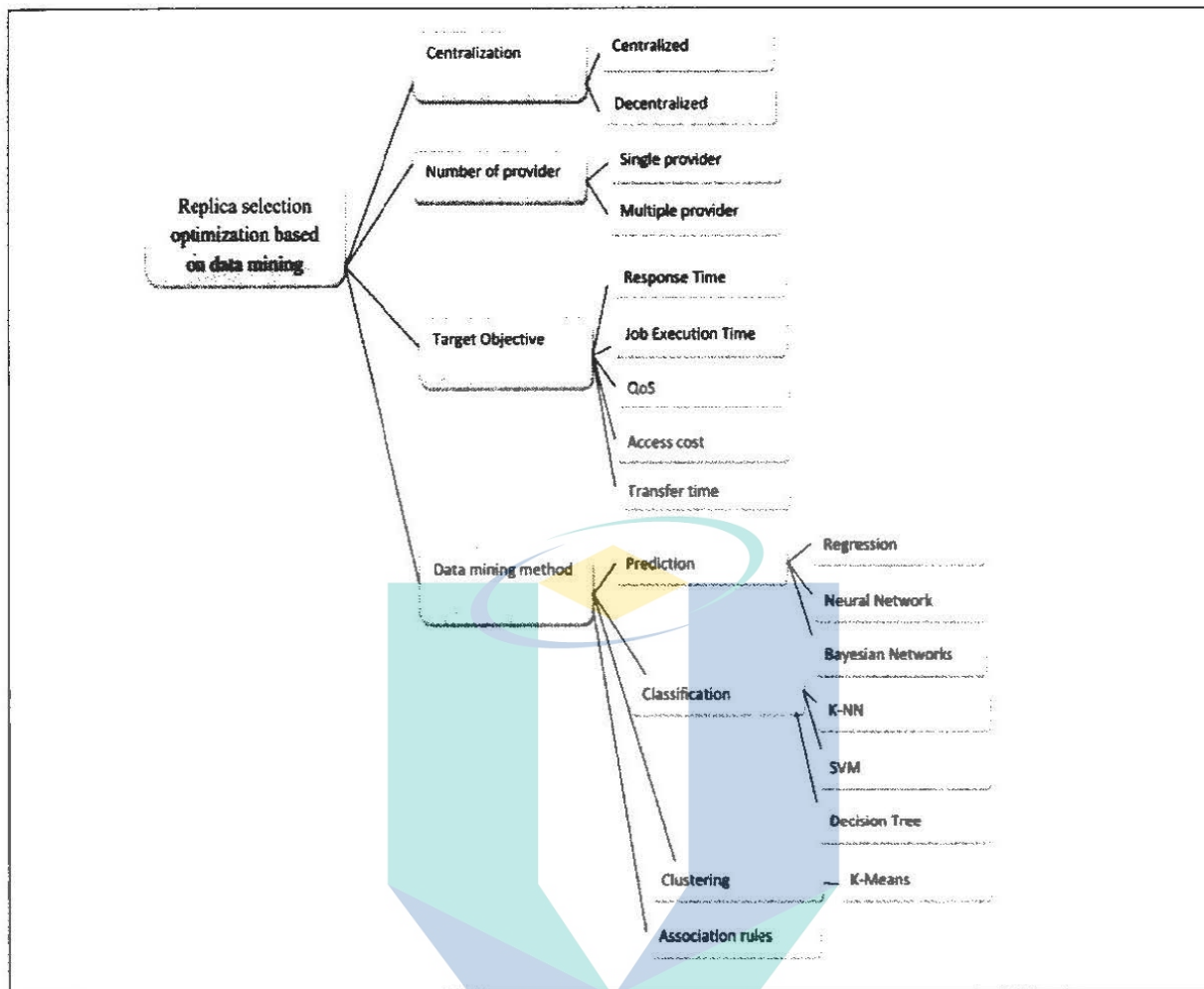


Figure 3.1 Replica selection strategies based on data mining techniques

Source: Hamrouni et al. (2015)

As shown in Figure 3.1, the first step would be to choose the centralization. In the present research, both centralized and decentralized techniques were used. Data mining process was executed in only one server. Details of the process involved in data mining will be explained in the data mining phase. After analysing the results, the server would fragment the database and distribute them into the multiple replication servers. For data replication process, the decentralized technique was used because the present research has been based on BVAG replication technique. BVAG was used because a copy would be allocated only to the neighbouring servers. This means that this technique would use the smallest size of quorums compared to other techniques reviewed in Chapter 2 of this thesis. Hence, BVAG would need low computational time to send and receive messages from its neighbouring servers. However, there would still be some drawbacks in using the BVAG technique. For instance, this technique would copy all files in its replication servers. Hence, data redundancy would still be high.

Besides that, when a server attempts to commit an update for a transaction, it has to send the whole file to its replication servers. This would cause high job processing time. The present research aimed to solve this issue by using fragmentation.

The second step would involve choosing the number of providers. Replica selection relying on a certain number of providers could be selected simultaneously. There are two types of replica selection, namely, the single replica provider and the multiple replica providers. In BVAGQ-AR, the single replica provider was selected for the data mining part in order to transfer the total requested data while the multiple providers was selected for the data replication part. In case of the multiple providers, several sites would concurrently work to send the requested data or their parts to minimize the total data transfer time which would provide high-performance access to data (Almuttairi et al., 2013).

The third step would be the target objectives. In the present research, two targets have been considered during the experiments: (1) job execution time and access, and (2) the communication cost. These two targets have been chosen in order to achieve the first research objective, namely, to design and develop BVAGQ-AR algorithms that would emphasize on data mining and data replication in distributed database environment with low communication cost and processing time for a transaction.

Finally, the fourth step would involve choosing the suitable data mining technique. Data mining technique that has been deployed in this experiment is called association rules. The technique was used to discover the correlation between data. Apriori algorithm from association rules was used for frequent itemset mining. Learning association rules basically means finding the items that appear together more frequently than the others.

In BVAGQ-AR, all sites would be logically organized in the form of a two-dimensional grid structure. For example, if BVAGQ-AR consists of 25 sites, the sites would be logically organized in the form of 5 x 5 grid. In this section, BVAGQ-AR is proposed by considering the distributed database fragmentation. The notations defined are as follows:

- i. S is a relation in database;
- ii. S' is relation after mining;
- iii. s is the instance in S or S' ;
- iv. J_1 is the frequent itemsets;
- v. J_2 is not the frequent itemsets;
- vi. $S(B)^1$ is the four sites in the corners;
- vii. $S(B)^2$ is the other sites on the boundaries;
- viii. $S(B)^3$ is the middle sites;
- ix. V is a transaction;
- x. T is a tuple in J_1 ;
- xi. x is an instant in T which will be modified by element of V ;
- xii. y is an instant in T which will not be modified by element of V ;
- xiii. S_1 is a vertical fragmented relation with instant x derived from J_1 ;
- xiv. S_2 is a vertical fragmented relation without instant x derived from J_1 ;
- xv. Pk is a primary key;
- xvi. Pk,x is a primary key with data x ;
- xvii. Pk,y is a primary with data y , where $y \neq x$;
- xviii. $S_{1(Pk,x)}$ and $S_{1(Pk,y)}$ are a horizontal fragmentation relation derived from J_1 ;
- xix. η and ψ are groups for the transaction V ;
- xx. $\lambda = \eta$ or ψ where it represents different transaction V (before and until get quorum);
- xxi. V_η is a set of transactions that comes before V_ψ , while V_ψ is a set of transactions that comes after V_η ;
- xxii. D is a union of all data objects managed by all transactions V of BVAGQ-AR;
- xxiii. Target set = $\{1, 0\}$ is a result of transaction V (see Table 3.3);
- xxiv. BVAGQ-AR transaction element V_λ is an element either in different set of transactions V_η or V_ψ ;
- xxv. wV_λ is write counter for the transaction;
- xxvi. \hat{V}_{λ_x} is a transaction that is transformed from V_{λ_x} ;
- xxvii. V_{μ_x} represents the transaction feedback from a neighbour site. V_{μ_x} exists if either V_{λ_x} or \hat{V}_{λ_x} exists;
- xxviii. Successful transaction at primary site $V_{\lambda_x} = 0$ where $V_{\lambda_x} \in D$ (i.e., the

transaction locked an instant x at primary). Meanwhile, successful transaction at neighbour site $V(\mu_x) = 0$, where $\mu_x \in D$ (i.e., the transaction locked a data x at neighbour); and,

xxix. $\lfloor \frac{n}{2} \rfloor$ is the greatest integer function (i.e., $n=9$, $\lfloor \frac{9}{2} \rfloor = 5$).

3.3 BVAGQ-AR Transaction Manager (BTM)

In the proposed BVAGQ-AR, there would be four important phases which are illustrated in Figure 3.2.

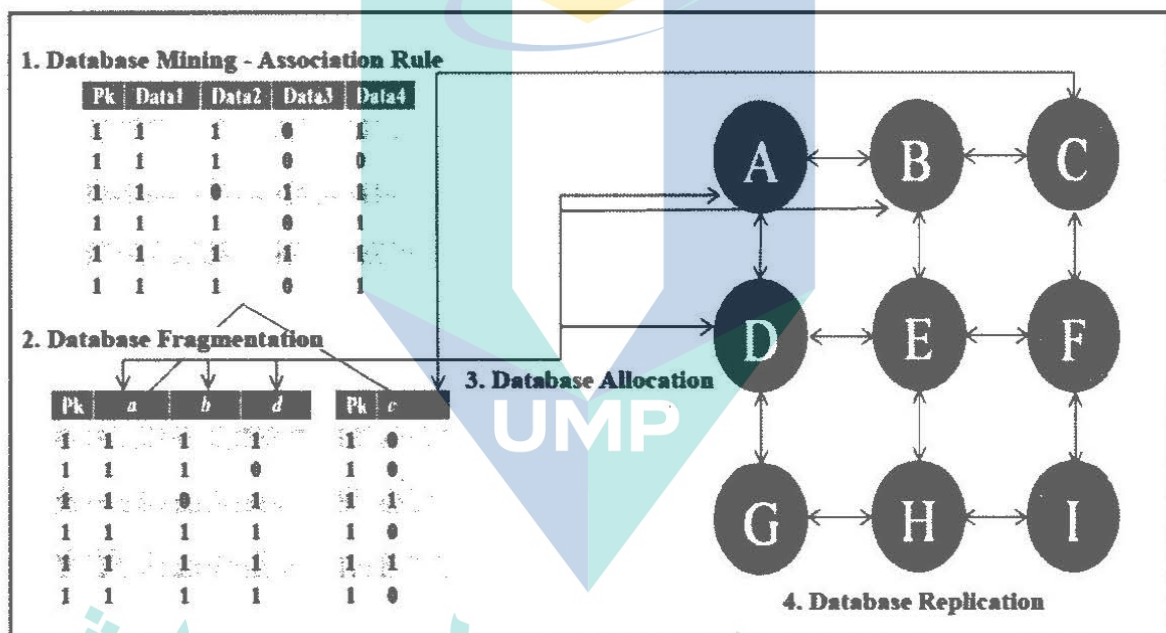


Figure 3.2 BVAGQ-AR technique illustrations

All the processes involved in these phases would be controlled by BVAGQ-AR Transaction Manager (BTM). Every transaction would go through the BTM before undergoing processing. Each primary or neighbour replica $i \in S(B)$ would have its own BTM. The BTM is divided into two parts, namely, data mining part and data replication part. BTM data mining functions would include: (1) Receiving a preliminary database, S from a client; (2) Executing Apriori Algorithm on S' in order to identify J_1 and J_2 ; (3) Fragmenting S' into J_1 and J_2 ; and, (4) Allocating J_1 at replication servers.

Meanwhile, BTM data replication functions would include:

- i. Accepting transactions from clients either V_η or V_ψ . When V_λ , $\lambda = \eta$ or ψ coming concurrently, based on the small arrival rate;
- ii. Receiving all types of transaction V_λ , $\lambda = \eta$ or ψ from clients, \hat{V}_{λ_x} from primary and neighbour replica. Each transaction would go through the BTM for the purpose of the determination of type of replica (either to be as primary or neighbour replica processing);
- iii. Performing synchronous commit \hat{V}_{λ_x} after the user has finished update the data, neutralizing all flags and unlocking instant x ; and,
- iv. If a replica was required to release a lock from another primary replica, BTM would abort V_λ at its replica and rollbacks all the transactions.

The algorithm *bvagqar_transaction_management()* of Listing 2 would implement the BVAGQ-AR Transaction Manager (BTM) for data replication part. In the algorithms, several important variables used would include:

- i. A *transStat* referring to the status of the transaction.
- ii. The unique process id of V_λ indicated as *pid* $_V_\lambda$ would be stored for the purpose of transaction processing such as queueing the transaction would recognize transaction that would get locked, handle incoming and outgoing transaction, and identify transaction that would need to abort or commit.
- iii. The login time of V_λ represented as *log* $_tV_\lambda$ would be very important in determining which transaction supposed to proceed in case timestamping method would be required.
- iv. A *\$LCount* representing the counter lock used to check either the write counter or unknown status counter already obtained by a particular replica $i \in S(B)$. It would be set to 0 to indicate that V_λ still has not obtained either the write or unknown status counter from its neighbour or otherwise it would be set to 1.
- v. The *xServ_Vote* would be used to show the locking phases of V_λ , \hat{V}_{λ_x} or V_{μ_x} for a particular replica $i \in S(B)$ where NeighbourID[i]= $\$Serv$ (server). *xServ_Vote* would be set to 0 if data were unlocked and in neutral condition. Meanwhile, *xServ_Vote* would be set to 1 when user has already finished updating data x but V_λ has not finished committing. In addition, *xServ_Vote* would be set to 101

during the phases between obtaining locks until user has finished updating the data.

3.3.1 Data Mining in BVAGQ-AR

Listing 1 describes the first part of BVAGQ-AR (i.e., the data mining part). This part would involve database mining, database fragmentation and database allocation.

Listing 1: The BTM Algorithm: Data Mining

```

1  primary_replica_processing ( )
2  {
3  /* receive S */
4      receive (BTM of S);
5      convert (BTM of S  $\rightarrow$  1,0); /* convert S into binary format */
6      produce Apriori algorithm (S, minsupp) /* minsupp is minimum support
*/
7      /* find frequent itemsets */
8       $J_k = \{ \text{frequent items} \};$ 
9      for ( $k=2; J_{k-1} \neq \emptyset; k++$ ) {
10          $S' = \text{candidates generated from } J_{k-1}$ 
11         /* cartesian product  $J_{k-1} \times J_{k-1}$  */
12         /* eliminate  $J_2$ 
13         for each transaction  $V$  in  $S$ 
14             do {
15             /*increment the count of all candidates in  $S'$  that are
contained in  $S$  */
16                  $J_k = \text{candidates in } S' \text{ with minSupport}$ 
17             }
18     }
19     /* BTM fragment  $S'$  */
20     fragment (BTM of  $S' \rightarrow J_1, J_2$ );
21     while  $\pi s_1, s_2, \dots, s_n \in J_1$ 
22         do
23         if  $J_1 \geq 3,$ 
24             then allocate at  $S(B)^1$ 
25         Else if  $J_1 = 4$ 
26             then allocate at  $S(B)^2$ 
27         Else if  $J_1 \geq 5$ 
28             then allocate at  $S(B)^3$ 
29         End if
30     while  $\pi s_1, s_2, \dots, s_n \in J_2$ 
31         do
32             allocate at any  $S(B)$ 
33 }

```

Data mining technique that has been deployed in this experiment is called association rules because the technique was tasked to search for frequent patterns, correlations or associations between the data. Apriori algorithm in association rules has been used to find the frequent itemsets. Frequent itemset mining would specify the discovery of associations and correlations between items in large transactional or relational data sets. The finding of interesting correlation between massive amounts of the data transaction records would facilitate the decision-making processes, specifically the user behavior analysis.

Apriori algorithm was proposed by R. Agrawal & R. Srikant in 1994 for mining frequent itemsets for Boolean association rules. The name of the algorithm has been established on the fact that the algorithm uses prior knowledge of frequent itemset properties, which will be explained later.

Apriori is an iterative method known as a level-wise search where k -itemsets are used to explore $(k + 1)$ -itemsets. The Apriori algorithm can be explained in several stages or steps. First, the set of frequent 1-itemsets is discovered by scanning the database to determine the count for each item, and assembling the items that satisfy the minimum support. The resulting set is represented as L_1 . After that, L_1 is used to identify the set of frequent 2-itemsets, L_2 , which is later used to identify L_3 . This will continue further until no more frequent k -itemsets can be discovered. The process of discovering each of the L_k involves one full scan of the database.

An important property called the Apriori property is used to reduce the search space in order to improve the efficiency of the level-wise generation of frequent itemsets:

Apriori property: *All nonempty subsets of a frequent itemset must also be frequent.*

The Apriori property is based on the following observation: by definition, if an itemset, I , does not satisfy the minimum support threshold, $min\ sup$, then I is not frequent, that is, $P(I) < min\ sup$. If an item A is added to the itemset I , then the resulting itemset (i.e., $I \cup A$) cannot occur more frequently than I . Therefore, $I \cup A$ is not frequent either, that is, $P(I \cup A) < min\ sup$.

This Apriori property belongs to a special category of properties called antimonotonicity in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well. It is called antimonotonicity because the property is monotonic in the context of failing a test.

For instance, assuming that there is a transactional data from *MyGrants* transaction database, S , as shown in Table 3.1, transactional data are the data describing an event as a result of a transaction. Such data are used to illustrate the Apriori algorithm for finding frequent itemsets in S .

Table 3.1 Transactional data from *MyGrants*

Transactional ID (TID)	List of item IDs
T1	I1, I3, I4
T2	I2, I3, I5
T3	I2, I3
T4	I1, I2, I4
T5	I3, I4
T6	I2, I4
T7	I3, I4
T8	I2, I3, I4, I5
T9	I2, I3

1. In the initial step of the algorithm, every item is a member in the set of candidate 1-itemsets, C_1 . The algorithm will scan all of the transactions to total the number of occurrences for each item.
2. Assuming that the minimum support count required is 2, this means that the $min\ sup = 2$. At this time, support count is used. Thus, the support is $2/9 = 22\%$. The set of frequent 1-itemsets, L_1 , can then be determined. It consists of the candidate 1-itemsets that satisfy the minimum support. In this example, all of the candidates in C_1 satisfy minimum support.

Scan S for count of each C_1 →	C_1		Compare C_1 support count with minimum support →	L_1	
	Itemset	Supp. count		Itemset	Supp. count
	I1	2		I1	2
	I2	6		I2	6
	I3	7		I3	7
	I4	6		I4	6
	I5	2		I5	2

Figure 3.3 Set of frequent 1-itemsets

- To find the set for frequent 2-itemsets, L_2 , the algorithm uses the join, $L_1 \bowtie L_1$ to generate a candidate set of 2-itemsets, C_2 . Figure 3.3 shows that each subset of the candidates is frequent. Thus, no candidates are removed from C_2 during the prune step.
- Next, the transactions in S are scanned and the support count of each candidate itemset in C_2 is accumulated, as shown in the Figure 3.4.

Generates C_2 from L_1 candidate →	C_2		Scan S for count of each C_2 →	C_2		Compare C_2 support count with minimum support →	L_2	
	Itemset	Supp. count		Itemset	Sup. count		Itemset	Sup. count
	{I1, I2}			{I1, I2}	1		{I1, I4}	2
	{I1, I3}			{I1, I3}	1		{I2, I3}	4
	{I1, I4}			{I1, I4}	2		{I2, I4}	3
	{I1, I5}			{I1, I5}	0		{I2, I5}	2
	{I2, I3}			{I2, I3}	4		{I3, I4}	4
	{I2, I4}			{I2, I4}	3		{I3, I5}	2
	{I2, I5}			{I2, I5}	2			
	{I3, I4}			{I3, I4}	4			
	{I3, I5}			{I3, I5}	2			
	{I4, I5}			{I4, I5}	1			

Figure 3.4 Set of frequent 2-itemsets

- The set of frequent 2-itemsets, L_2 , is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.
- The detailed generation candidate 3-itemsets set, C_3 , is illustrated in Figure 3.5.

Generates C_3 from L_2 →	C_3	Scan S to count the minimum support of C_3 →	C_3	Sup. count	Compare support count for the C_3 with minimum support	L_3
	Itemset		Itemset			Itemset
	{I2, I3, I4}		{I2, I3, I4}	1		{I2, I3, I5}
	{I2, I3, I5}		{I2, I3, I5}	2		2

Figure 3.5 Set of frequent 3-itemsets

7. From the join step, $C_3 = L_2 \bowtie L_2 = \{\{I1, I2, I3\}, \{I1, I2, I4\}, \{I1, I3, I5\}, \{I1, I3, I4\}, \{I2, I3, I4\}, \{I2, I3, I5\}\}$. According to the Apriori property, all subsets of a frequent itemset must also be frequent. Hence, it is certainly determined that the four early candidates cannot be frequent. Therefore, the candidates are removed from C_3 . This process saves the effort of finding the candidates counts during the subsequent scan of S to determine L_3 .

- i. Join: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\} \bowtie \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\} = \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I4, I5\}\}$.
- ii. Prune using Apriori property: all nonempty subsets of a frequent itemsets must also be frequent. Does any of the candidates have a subset that is not frequent?

- The 2-item subsets of $\{I1, I2, I3\}$ are $\{I1, I2\}$, $\{I1, I3\}$ and $\{I2, I3\}$. All 2-item subsets of $\{I1, I2, I3\}$ are members of L_2 . Therefore, $\{I1, I2, I3\}$ are kept in C_3 .
- The 2-item subsets of $\{I1, I2, I5\}$ are $\{I1, I2\}$, $\{I1, I5\}$ and $\{I2, I5\}$. All 2-item subsets of $\{I1, I2, I5\}$ are members of L_2 . Therefore, $\{I1, I2, I5\}$ are kept in C_3 .
- The 2-item subsets of $\{I1, I3, I5\}$ are $\{I1, I3\}$, $\{I1, I5\}$ and $\{I3, I5\}$. $\{I3, I5\}$ is not a member of L_2 , and thus, it is not frequent. Therefore, $\{I1, I3, I5\}$ are removed from C_3 .

- The 2-item subsets of $\{I2, I3, I4\}$ are $\{I2, I3\}$, $\{I2, I4\}$ and $\{I3, I4\}$. $\{I3, I4\}$ is not a member of L_2 , and thus, it is not frequent. Therefore, $\{I2, I3, I4\}$ are removed from C_3 .
- The 2-item subsets of $\{I2, I3, I5\}$ are $\{I2, I3\}$, $\{I2, I5\}$ and $\{I3, I5\}$. $\{I3, I5\}$ is not a member of L_2 , and thus, it is not frequent. Therefore, $\{I2, I3, I5\}$ are removed from C_3 .
- The 2-item subsets of $\{I2, I4, I5\}$ are $\{I2, I4\}$, $\{I2, I5\}$ and $\{I4, I5\}$. $\{I4, I5\}$ is not a member of L_2 , and thus, it is not frequent. Therefore, $\{I2, I4, I5\}$ are removed from C_3 .

iii. Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

These results have shown that items I1, I2 and I3 always appeared together in a transaction, as well as items I1, I2 and I5. Hence, these items will be allocated and replicated in neighbour servers to make sure transactions can access them together faster and more easily. This process was applied to all the data involved in the experiments.

With reference to the algorithm in Listing 1, line 4 initializes the process by receiving a preliminary database, S , into one site. S will be converted into binary format. Each row corresponds to a transaction and each column corresponds to an item. An item can be treated as a binary variable. In other words, if the item is present in a transaction, the value should then be 1; if otherwise, the value should be 0.

Table 3.2: Database with binary variable

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>	<i>i</i>	<i>j</i>	<i>k</i>	<i>l</i>	<i>m</i>	<i>n</i>	<i>o</i>	<i>p</i>	<i>q</i>	<i>r</i>	<i>s</i>	<i>t</i>	<i>u</i>
1	1	1	1	0	0	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1
1	0	0	0	1	0	1	0	1	0	1	1	0	0	0	1	0	1	1	1	0
1	1	1	1	0	0	1	0	1	0	1	1	0	1	0	1	0	0	0	1	0
1	1	0	1	1	0	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1
1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	0	0	1	0
1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	0	0	1	1
1	1	0	1	1	0	1	0	1	0	1	0	0	1	0	1	0	1	1	1	1
1	1	1	1	1	0	1	0	1	0	0	1	0	1	0	1	0	1	1	1	1
1	1	1	1	1	0	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1
1	1	1	1	1	0	1	0	1	0	1	1	0	1	0	1	0	0	0	1	0
1	0	1	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0
1	0	0	0	1	0	1	0	1	0	1	1	1	1	0	1	0	0	0	1	1
1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1
1	1	0	1	1	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1
1	0	0	0	1	0	1	0	1	0	1	0	0	1	0	1	0	1	1	1	1
1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0
1	1	1	1	1	0	1	0	1	0	1	0	0	0	1	0	0	0	1	1	1
1	1	1	1	1	0	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1

For example, a database with binary variable is shown in Table 3.2. W and Z represent the instances in the database and n is the total number of transactions.

Support, s , is the fraction of transactions that contain both W and Z where

$$s = \frac{\sigma(a, b, c, d)}{n} = \frac{7}{20} = 0.35 @ 35\% \quad 3.1$$

Confidence, c , measures how often items in Z appear in transactions that contain W .

$$c = \frac{\sigma(a, b, c, d)}{\sigma(a, b)} = \frac{7}{10} = 0.7 @ 70\% \quad 3.2$$

For simplicity, data from row 1 to 5 and column 1 to 6 in Table 3.2 are used as an example case. Figure 3.6 shows an illustration of the frequent itemset generation in

the Apriori algorithm for the transactions. Line 6 produces the Apriori algorithm support threshold. Assuming that the support threshold is 60%, this is equivalent to a minimum support count equal to 3 because in this example, the items have to appear more than half of the transactions to be taken as a frequent itemset. In large databases, if the threshold is 40% or below, all the data will most likely appear together.

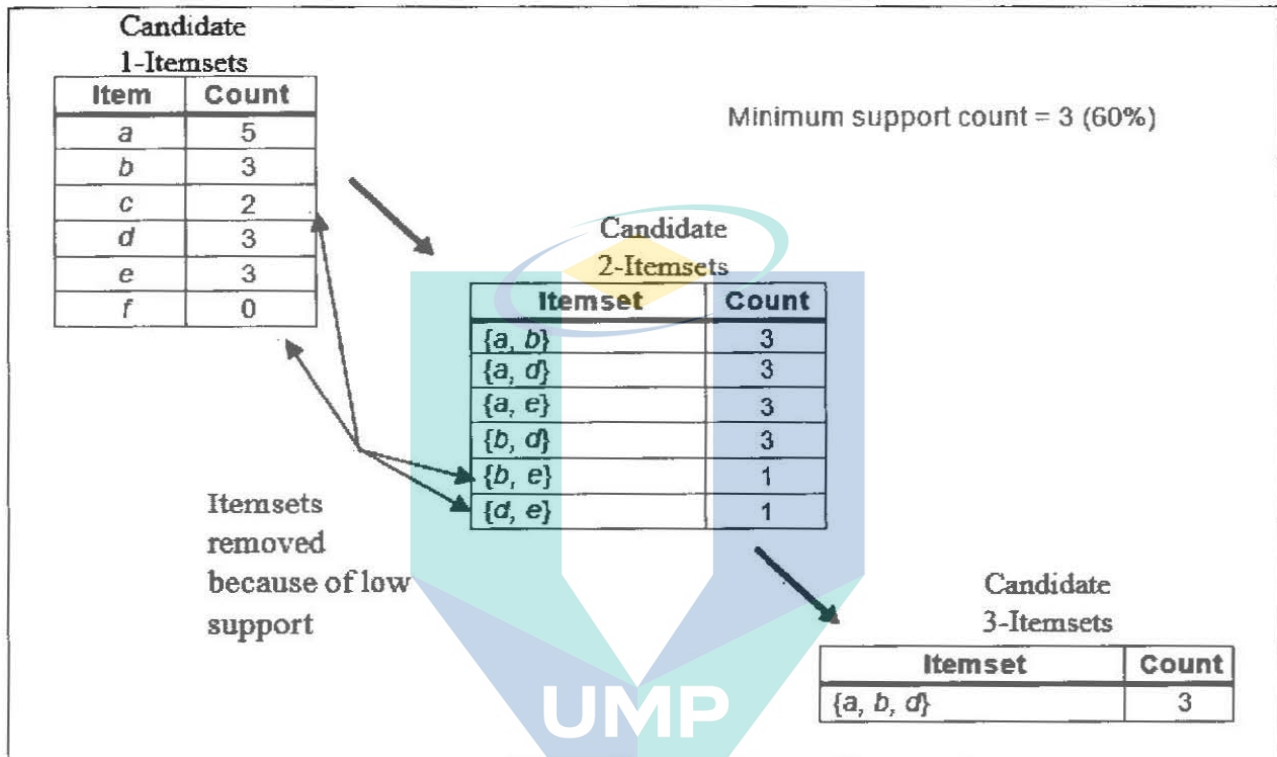


Figure 3.6 Generating frequent itemsets using the Apriori algorithm

The loop starts at line 8. Initially, every item is considered as a candidate 1-itemset. After counting their supports, the candidate itemsets {c} and {f} are discarded because they appear in fewer than three transactions. Lines 9-16 indicate the next iteration where candidate 2-itemsets are generated using only the frequent 1-itemsets because the Apriori algorithm ensures that all supersets of the infrequent 1-itemsets must be infrequent. Because there are only four frequent 1-itemsets, the number of candidate 2-itemsets generated by the algorithm is $\binom{4}{2} = 6$.

Two of these six candidates, {b, e} and {d, e}, are subsequently found to be infrequent after computing their support values. The remaining four candidates are frequent, and thus, they will be used to generate candidate 3-itemsets. Without support-based pruning, there are $\binom{6}{3} = 20$ candidate 3-itemsets that can be formed using the six items given in this example. With the Apriori algorithm, only candidate

3-itemsets whose subsets are frequent will be kept. The only candidate that has this property is $\{a,b,d\}$. The loop will iterate until all the frequent itemsets are determined.

This technique also has been proposed to make sure data replication can be effectively done while minimizing storage. In general, applications work with some relations rather than the entire relations. Therefore, working with subsets of a relation is better than working as one unit of distribution in data distribution. Thus, not all data will be replicated to all sites.

In line 20, the relation that results from identifying the frequent itemsets, S' is fragmented into relation with frequent itemsets, J_1 , and relation without frequent itemsets, J_2 , using vertical fragmentation. When S' is fragmented, it is divided into a number of fragments S'_1, S'_2, \dots, S'_n .

$$S' = S'_1 \cup S'_2 \cup \dots \cup S'_n \quad 3.3$$

The fragmentation should be done in such a way that relation S can be reconstructed from the fragments:

$$S' = S'_1 \bowtie S'_2 \bowtie S'_3 \bowtie \dots \bowtie S'_n \quad 3.4$$

It is necessary to include the primary key or some candidate key attributes in every vertical fragment so that the full relation can be reconstructed from the fragments.

After the fragmentation process has finished, fragmented database allocation process starts at line 21. Line 23 checks if J_1 is less than or equivalent to three, then line 24 allocates the data at $S(B)^1$ because the $S(B)^1$ has three replication servers. If the J_1 is equivalent to four, then line 26 allocates the data at $S(B)^2$ because the $S(B)^2$ has four replication servers. If J_1 is more than or equivalent to five, then line 28 allocates the data at $S(B)^3$ because the $S(B)^3$ has five replication servers. Lines 30-32 allocate the non-frequent itemsets at any the $S(B)$ s. After all data are replicated to their specific servers, the replication process can be executed.

3.3.2 Data Replication in BVAGQ-AR

Listing 2 describes part of the process for the second phase of BVAGQ-AR which is the data replication part. This algorithm shows how BTM works when a transaction is initiated at a primary and neighbour server.

Listing 2: The BTM Algorithm: Data Replication

```

1  manage_bvaqgar_transaction ( )
2  {
3  while (Incomplete)
4  do
5      while (transStat  $\neq$  "Abort")
6      do
7          /* receive  $V_{\lambda_x} \parallel \hat{V}_{\lambda_x} \parallel V_{\mu_x}$  where  $\lambda = \eta, \psi$  either from client or any BTM of
           replica  $i \in S(B)$  */
8          receive (client @ BTM of  $i \in S(B) : V_{\lambda_x} \parallel \hat{V}_{\lambda_x}$ );
9          pid_ $V_{\lambda_x}$  = process id of  $V_{\lambda_x}$ ;
10         log_t $V_{\lambda_x}$  = login time of  $V_{\lambda_x}$ ;
11         /*recognize replica task either to be as primary or neighbour processing
           for  $V_{\lambda_x}, \lambda = \eta, \psi$  */
12         switch {
13             case (receive(client @ BTM of neighbour:  $V_{\lambda_x} \parallel \hat{V}_{\lambda_x} \parallel V_{\mu_x}$ ):
14                 primary_replica_processing ( );
15                 break;
16             case (receive(BTM of primary:  $V_{\lambda_x} \parallel \hat{V}_{\lambda_x}$ )):
17                 neighbour_replica_processing ( );
18                 break;
19             }
20         receive (BTM:  $\hat{V}_{\lambda_x}, x\$serv\_Vote \parallel \hat{V}_{\lambda_x}, x\$serv\_Vote,$ 
21              $\_LCount, uV_{\lambda_x}$ );
22         if (x$serv_Vote = 1) then
23             Commit  $\hat{V}_{\lambda_x}$ ;
24         endif
25         if (x$serv_Vote = 0);
26         Endif
27         /*On receiving transStat = "Abort" from other replica and release its lock*/
28         if (receive (BTM:  $\hat{V}_{\lambda_x}, \in V_{\eta}$  transStat, PrimaryID)) then
29              $V_{\lambda_x} \in V_{\psi} = V_{\lambda_x}$ ; /*current  $V_{\lambda_x}$  become  $V_{\lambda_x} \in V_{\psi}$  */
30              $\hat{V}_{\lambda_x} = \hat{V}_{\lambda_x} \in V_{\eta}$ ; /* $\hat{V}_{\lambda_x} \in V_{\eta}$  that BTM received will survive*/
31         Abort  $V_{\lambda_x}$ ;
32         Rollback;
33          $V_{\lambda_x} = 1$ ; /*Target Set is equal to 1, means primary already gets lock*/
34     endif

```

Each site now has a primary data file. A site is either in operational or failed state, and the state (i.e., operational or failed) of each site is statistically independent of the others. When a site is operational, the copy at the site is available; otherwise, it is unavailable.

The primary replica for a particular instant x is a replica that accepts the client's request. In BVAGQ-AR model, each replica of $S(B)$ can be a primary or a neighbour replica at the same time. Any replica $i \in S(B)$ can be chosen as the primary replica, while other replicas $j \in S(B)$ where $i \neq j$ are neighbours. Line 7 receives a transaction V_{λ_x} requests to update instant x from any replica of $S(B)$. Lines 8-18 determine that replica will be the primary, while others will be the neighbour replica for processing V_{λ_x} .

If $S(B)$ is the set of replicas with replicated copies being stored corresponding to the assignment B for particular instant x , then $S(B_x) = \{m(i,j), m(i-1,j), m(i,j-1), m(i,j+1), m(i+1,j)\}$. Two sets of transactions are V_{η} , which requests instant x from $m(i,j)$ replica, and V_{ψ} , which requests instant x from $m(i-1,j)$, respectively. The $m(i,j)$ replica functions as the primary replica for processing V_{η} , where $m(i-1,j)$, $m(i,j-1)$, $m(i,j+1)$, $m(i+1,j)$ are neighbour replicas for processing $V_{\gamma_x} \in V_{\eta}$. Simultaneously, $m(i-1,j)$ replica functions as the primary replica for processing V_{ψ} , while $m(i,j-1)$, $m(i,j+1)$, $m(i+1,j)$ and $m(i,j)$ are neighbour replicas for processing $V_{\gamma_x} \in V_{\psi}$. Both $m(i,j)$ and $m(i-1,j)$ replicas execute two different processing tasks concurrently. The $m(i,j)$ replica is the primary replica processing V_{η} and neighbour replica processing for V_{ψ} , whereas the $m(i-1,j)$ replica is the primary replica for processing V_{ψ} and neighbour replica for processing V_{η} . BVAGQ-AR model considers different sets of transactions V_{η} and V_{ψ} . V_{η} is a set of transactions that comes before V_{ψ} . The effect of BVAGQ-AR transaction is defined as the processing of one instance of the transaction. Hence, the following *type* is mapping as $V : D \rightarrow$ Target Set as shown in Table 3.3

Table 3.3 Target set

Target set	Meaning
0	<p>This means that no failure occurred during BVAGQ-AR transaction's execution. By $V(\lambda_x) = 0$, where $\lambda_x \in D$ is the instant processed by the transaction at primary site and $y = \eta, \psi$, it means that the transaction was successful. (i.e., the transaction locked the instant x at primary). Write counter will track this value.</p> <p>For neighbour site, $V(\mu_x) = 0$, where $\mu_x \in D$</p>
1	<p>This means accessing failure. By $V(\lambda_x) = 1$, it means that the destination server could not perform the job. This is because the instant x managed by the primary site is not free and already locked. Hence, BVAGQ-AR transaction has not executed.</p> <p>For neighbour site, $V(\mu_x) = 1$ where $\mu_x \in D$</p>

All elements of V_η and V_ψ may request data object x simultaneously at any site of $S(B)$ either at the same or different site. When BTM receives a transaction, line 20 checks the write counter for that particular replica. Line 22 sets the lock status to 1 when a user has already finished updating data x but V_λ has not finished committing. Then, line 23 will commit. If the lock status is set to 0 by line 25, the data are unlocked and in neutral condition. Lines 28-34 abort a transaction when the target set of the replica is equal to 1 because the replica is locked by other transaction.

3.4 The Coordinating Algorithm for Primary Replica

When a primary replica receives a set of transactions from clients, the BTM will pass the transactions to *primary_replica_processing* algorithm to maintain the data availability and consistency of all replicas.

3.4.1 Initial Lock

When any user invokes a transaction to update data x , the primary replica requests a lock for itself by calling *initialize_lock* function of Listing 3.

Listing 3: The initialize lock function

```
1 initialize_lock ()
2 {
3 receive  $V_{\lambda x}$ ;
4  $wV_{\lambda x}++$ ;
5  $V_{\lambda x}=1$ ;
6  $x\$serv\_Vote=101; V_{\lambda x, q1}$ 
7 Abort  $V_{\lambda+1x}, \dots, V_{\lambda+qx}$ ;
8 }
```

Line 3 receives a transaction from $V_{\lambda x}$. When a transaction gets a lock at its primary replica, line 4 sets the Target Set to 1. Line 5 increases the write counter for the transaction when it successfully locks its neighbours' replicas. Line 6 is the intermediate phases between gets the lock until the user has finished updating data x . Line 7 aborts all transactions that come after $V_{\lambda x}$.

3.4.2 Propagate Lock

The *request_lock_synchronous* function of Listing 4 is used by the primary for propagating a lock to its entire neighbours.

Listing 4: The Request Lock Synchronous Function

```
1 request_lock_synchronous ( $V_{\lambda x}, T\$serv, \$\_LCount, wV_{\lambda x}, uV_{\lambda x}$ )
2 {
3 /*  $V_{\lambda x}$  still has not obtained the quorum since write counter less than majority and
   a particular
4 server  $i \in S(B)$  still has not obtained the write counter from its neighbour */
```

Listing 4: Continued

```
5   if ( $wV_{\lambda_x} < \left\lceil \frac{d}{2} \right\rceil$  && , $_LCount = 0) then
6       send (BTM:  $wV_{\lambda_x}$ , PrimaryID);

7       tell the BTM to request lock from neighbour;
8       receive (BTM:  $V_{\mu_x}$ );
9       switch {
10          case ( $uV_{\lambda_x} = 0$ ): /*Target Set is equal to 0 at neighbour, means data x has
              free lock*/
11               $wV_{\lambda_x} ++$ ;
12              $_LCount = 1; /*Initiates  $V_{\lambda_x}$  already gets write counter*/
13          break;
14      }
15  endif
16 }
```

V_{λ_x} propagates its lock to the neighbours' replicas until it gets the majority quorum. Recall the Binary Vote Assignment on Grid (BVAG) technique (Deris et al., 2003). However, BVAG only covers the voting and a part of the replication process.

Definition 3.1: A site X is a neighbour to site Y , if X is logically located adjacent to Y .

A data will replicate to the neighbouring sites from its primary site. The number of data replication, d , can be calculated using Property 3.1, as described below.

Property 3.1: The number of data replication from each site, $d \leq 5$.

Proof: Let n be a set of all sites that are logically organized in a two-dimensional grid structure form. Then, n sites are labelled $m(i, j)$, $1 \leq i < \sqrt{n}$, $1 \leq j < \sqrt{n}$. Two-way links will connect site $m(i, j)$ with its four neighbours, sites $m(i \pm 1, j)$ and $m(i, j \pm 1)$, as long as there are sites in the grid. It should be noted that four sites on the corners of the grid have only two adjacent sites, and other sites on the boundaries have only three neighbours. Thus, the number of neighbours of each site is less than or equal to 4. Because the data will be replicated to neighbours, the possible number of data replication from each site, d , should then be:

$d \leq$ the number of neighbours + a data from site itself

$$\leq 4 + 1 = 5.$$

For example, as shown in Figure 3.7, data from site A are replicated to site B and site D which are their neighbours. Site E has four neighbours, namely, sites B, D, F and H. As such, site E has five replicas. Meanwhile, data from site F are replicated to site C, E and I.

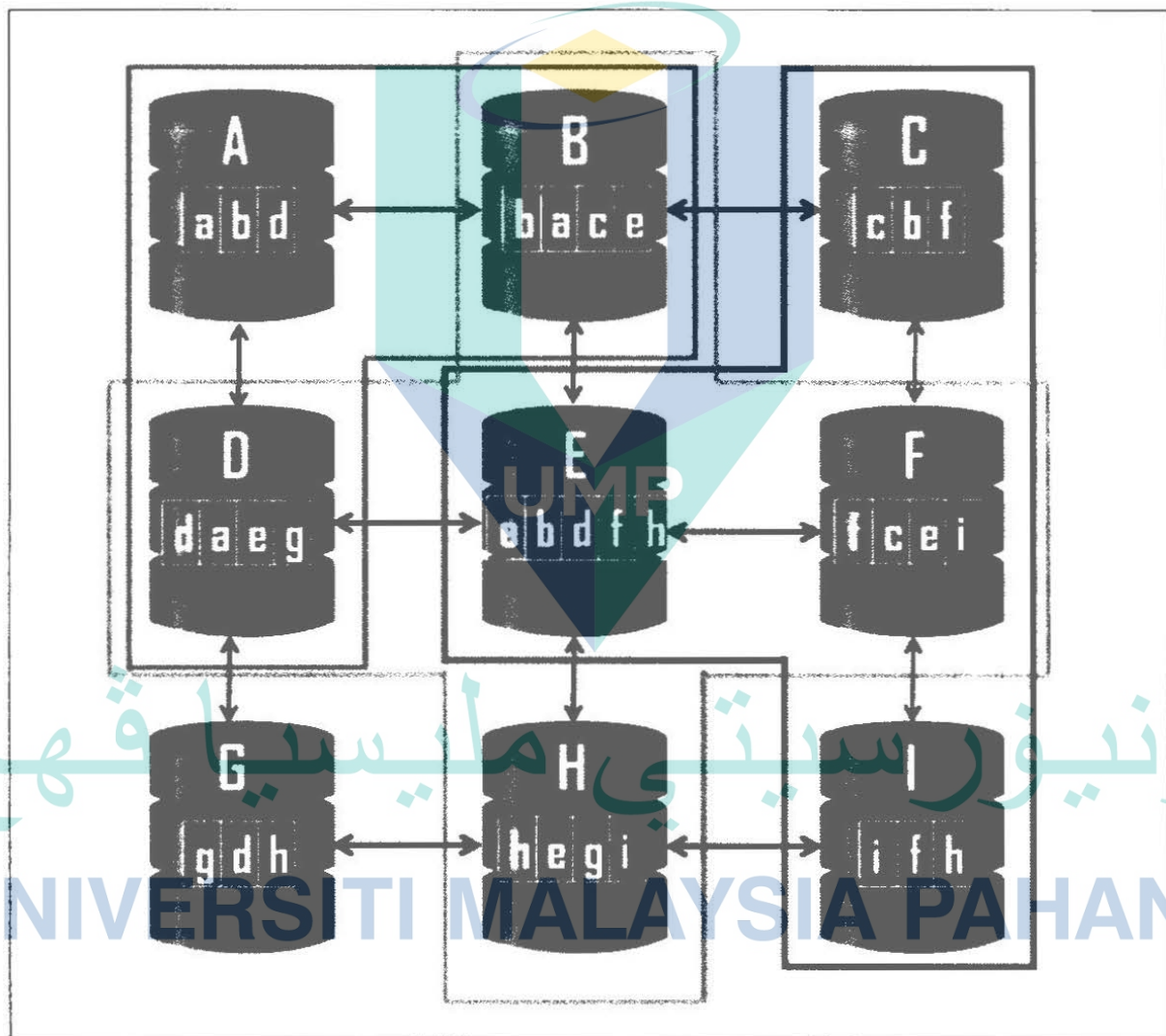


Figure 3.7 Examples of data replication in BVAGQ-AR.

The primary site of any data file and its neighbours are assigned with either vote 1 or otherwise, vote 0. This vote assignment is called neighbour binary vote assignment on grid. A neighbour binary vote assignment on grid, B , can be expressed as follows:

$$B(i) \in \{0,1\}, 1 \leq i \leq n \quad 3.5$$

where $B(i)$ is the vote assigned to site i . This assignment is treated as an allocation of replicated copies and a vote assigned to the site results in a copy allocated at the neighbour, that is:

$$1 \text{ vote} \equiv 1 \text{ copy} \quad 3.6$$

Let,

$$L_B = \sum_{i=1}^n B(i) \quad 3.7$$

where, L_B is the total number of votes assigned to the primary site and its neighbours and it is also equal to the number of copies of a file allocated in the system. Thus, $L_B =$

If r and w denote the read quorum and write quorum, respectively, to ensure that the read operation always gets up-to-date value, $r + w$ must be greater than the total number of copies (votes) assigned to all sites. The following conditions are used to ensure the consistency:

- 1) $1 \leq r \leq L_B, 1 \leq w \leq L_B$
- 2) $r + w = L_B + 1$

Conditions (1) and (2) ensure that there is a nonempty intersection of copies between every pair of read and write operation. Thus, the conditions ensure that a read operation can access to most recently updated copy of the replicated data. Condition (2) relates to nonempty quorum intersection property because the total number of read quorum and write quorum is more than L_B when there is an increase of one quorum. A data item is not read and written by two transactions concurrently. BVAGQ-AR obeys one-copy serializability under these conditions. Therefore, the replicated data item always remains consistent at multiple sites.

If $S(B)$ is the set of sites at which replicated copies are stored corresponding to the assignment B , then:

$$S(B) = \{i \mid B(i) = 1, 1 \leq i \leq n\} \quad 3.8$$

Definition 3.2: For a quorum q , a *quorum group* is any subset of $S(B)$ which has a size that is greater than or equal to q . The collection of quorum group is defined as the *quorum set*.

If $Q(B, q)$ is the quorum set with respect to the assignment B and quorum q , then:

$$Q(B, q) = \{G \mid G \subseteq S(B) \text{ and } |G| \geq q\} \quad 3.9$$

For example, as shown in Figure 3.8, site E is the primary site of the primary data item k . Its neighbours are sites B, D, F and H. Black circles represent other sites that do not have the replicated data item k . If an assignment B for the data item k is considered as $B_k(E) = B_k(B) = B_k(D) = B_k(F) = B_k(H) = 1$ and $L_{B_k} = B_k(E) + B_k(B) + B_k(D) + B_k(F) + B_k(H) = 5$, therefore, $S(B_k) = \{E, B, D, F, H\}$.

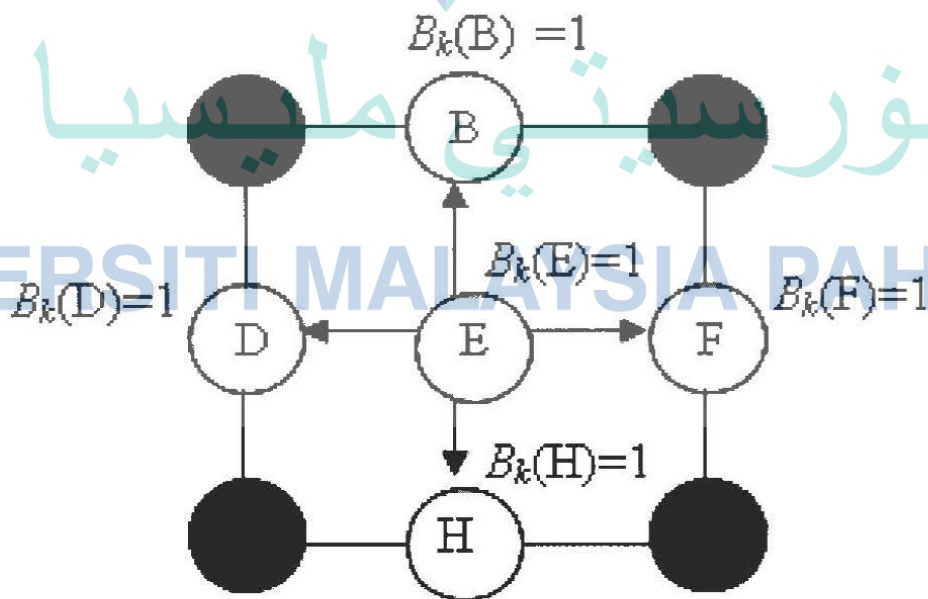


Figure 3.8 An assignment B for data file k where $S(B_k) = \{E, B, D, F, H\}$

If a read quorum for data file k , $r = 2$ and a write quorum $w = L_{B_k} - r + 1 = 4$, then the quorum sets for read and write operations are $Q(B_k,2)$ and $Q(B_k,4)$ respectively, where

$$Q(B_k,2) = \{\{E,B\}, \{E,D\}, \{E,F\}, \{E,H\}, \{B,D\}, \{B,F\}, \{B,H\}, \{D,F\}, \{D,H\}, \{F,H\}, \{E,B,D\}, \{E,B,F\}, \{E,B,H\}, \{E,D,F\}, \{E,D,H\}, \{E,F,H\}, \{B,D,F\}, \{B,D,H\}, \{B,F,H\}, \{D,F,H\}, \{E,B,D,F\}, \{E,B,D,H\}, \{E,B,F,H\}, \{E,D,F,H\}, \{B,D,F,H\}, \{E,B,D,F,H\}\}$$

and

$$Q(B_k,4) = \{\{E,B,D,F\}, \{E,B,D,H\}, \{E,B,F,H\}, \{E,D,F,H\}, \{B,D,F,H\}, \{E,B,D,F,H\}\}$$

The loop starts at line 5, if the counter lock for V_{λ_x} is lower than $\lfloor d/2 \rfloor$ or equal to 0, then line 6 will tell the BTM to request lock from the neighbour. Line 8 receives feedback from the neighbour. If the Target Set is equal to 0 at the neighbor replica, then data x has free lock. Lines 11-12 will set the write counter lock to 1, indicating that V_{λ_x} already gets a write counter. This loop will iterate until V_{λ_x} gets the majority quorum.

3.4.3 Primary Replica Processing

Listing 5 describes the primary replica processing algorithm.

Listing 5: The Primary Replica Processing Function

```

1 primary_replica_processing ()
2 {
3   while (TRUE)
4   do
5     receive (BTM:  $V_\eta$ );
6     if  $\exists V_\psi$  then
7       receive (BTM:  $V_\psi$ );
8     endif
9     switch {

```

Listing 5: Continued

```
15             break;
16         case ( $\lambda = \eta, \psi \ V_\eta \neq V_\psi$ ):
17             initialize_lock ();
18             Abort  $V_\psi$ ;
19             break;
20     }
21     break;
22     case ( $V_{\lambda_x} = 1$ ):
23         Abort  $V_\psi$ ;
24         InComplete=FALSE;
25     break;
26 }
27  $uV_{\lambda_x} = 0$ ; /*neutralize unknown status counter*/
28 while ( $(wV_{\lambda_x} < \lfloor \frac{d}{2} \rfloor) \ \&\& \ (uV_{\lambda_x} < \lfloor \frac{d}{2} \rfloor) \ \&\& \ (\text{transStat} \neq \text{"Abort"})$ );
29 do
30     for  $\forall$  neighbour  $i \in S(B)$ 
31         do
32             COBEGIN
33                 Serv= NeighbourID[i]; /*particular server  $i \in S(B)$  */
34                  $\_L\text{Count} = 0$ ; /* still does not get the write counter for particular
server  $i \in S(B)$  */
35             COEND
36         if ( $uV_{\lambda_x} \geq \lfloor \frac{d}{2} \rfloor \ \&\& \ (d = 5)$ ) then
37             /*replicas failure more than majority quorum-timestamping method*/
38             for  $\forall$  neighbour  $i \in S(B)$ 
39                 do
40                     if ( $\_L\text{Count} = 0$ ) /* $V_\eta$  fail get lock since  $V_\psi$  already hold x lock */
```

Listing 5: Continued

```
41     then
42         Execute on server;
43         send (BTM:  $V_{\lambda_x}$  of neighbour, PrimaryID);
44     else
45          $\hat{V}_{\lambda_x} = V_{\lambda_x} \in V_{\eta}$ ;
46         break;
47     endif
48     if ( $\log\_t V_{\lambda_x}$  of primary <  $\log\_t V_{\lambda_x}$  of neighbour) then
49          $\hat{V}_{\lambda_x} = V_{\lambda_x} \in V_{\eta}$ ; /*the earliest transaction survives*/
50          $V_{\psi} = V_{\lambda_x}$  of neighbours;
51          $V_{\psi}$ 's transStat = "Abort";
52         send (BTM:  $\hat{V}_{\lambda_x} \in V_{\eta}$ ,  $V_{\psi}$ 's transStat, PrimaryID), to neighbour
53      $i \in S(B)$  &&
54         $ _LCount =0;
55     else /*  $V_{\lambda_x}$  of neighbours will perform transaction*/
56          $\hat{V}_{\lambda_x} = V_{\lambda_x} \in V_{\psi}$ ;
57     endif
58 endif
59 change the access permission mode of data  $x$ ; /* user can update data  $x$ */
60 send (client:  $\hat{V}_{\lambda_x}$ )
61     tell client to start modify data object  $x$ ;
62     wait until user finishes update;
63     fragment  $S' \rightarrow S_I$  and  $S_2$ 
64     fragment  $S_I \rightarrow S_{1(Pk,x)}$  and  $S_{1(Pk,y)}$ .
65     xServ_Vote =1;
66     COBEGIN
67         send (BTM:  $\hat{V}_{\lambda_x}$ , xServ_Vote), to  $\forall$  replica  $i \in S(B)$ ;
68     COEND
69     tell  $\forall$  BTM of replica  $i \in S(B)$  to commit  $\hat{V}_{\lambda_x}$ ;
70 }
```

The process starts with line 5 receiving V_η in. If V_ψ gets the majority quorum, then line 7 receives V_ψ as the primary replica. In line 10, $V_{\lambda_x} = 0$ means data x at primary has free lock. If there are two transactions V_η and V_ψ requesting to update data x at different replicas, both of the transactions will then initiate their lock in lines 12-15. If there are two transactions V_η and V_ψ requesting to update data x at the same replicas, line 18 will then initiate the lock for V_η while line 19 will abort V_ψ . In line 23 where $V_{\lambda_x} = 1$, data x at primary is already locked by other transaction. Hence, line 24 aborts V_ψ .

When the transaction is not aborted, BTM synchronously propagates $V_{\lambda_x} \in V_\eta$ lock for the purpose to obtain quorum. If V_η gets the majority quorum but still does not get the write counter for particular replica because V_ψ already holds x lock, then line 44 will tell BTM to request V_{λ_x} login time of neighbour. Otherwise, V_η prepares to perform the transaction so that the user can update as shown in line 46.

V_η is the earliest transaction and transaction at neighbour, V_ψ came after the transaction at primary. Hence, in line 53, V_ψ is aborted. Lines 54-55 tell BTM of the neighbour replica to release the lock so that primary can obtain the quorum. Meanwhile, lines 56-57 perform the transaction for V_ψ whereas lines 61-63 tell user to modify the data object x and wait until the user finishes the update. Then, lines 78-79 fragment the database using vertical and horizontal fragmentations. In line 66, the user has already finished updating data x but V_{λ_x} still has not finished committing. Hence, lines 68-70 tell the BTM to commit the transaction.

UNIVERSITI MALAYSIA PAHANG

3.5 The Cooperative Algorithm for Neighbour Replica

When a neighbour replica receives a request from a primary replica, it checks whether or not the request can proceed and act accordingly. The *neighbour_replica_processing* algorithm is given in Listing 6.

Listing 6: The Neighbour Replica Processing Function

```
1  neighbour_replica_processing ()
2  {
3  while (TRUE)
4  do
5      x$Serv_Vote = 0;
6      receive (BTM:Vλx, PrimaryID);
7  switch {
8      case (Vμx = 0):
9          send (BTM: Vμx );
10         Vμx = 1
11         break;
12     case (Vμx = 1):
13         send (BTM: Vμx);
14         break;
15     }
16 }
```

Data x is unlocked and in neutral condition for the particular neighbour site if the xServ_Vote = 0$. For case $V_{\mu_x} = 0$, the Target Set is equal to 0, which means data x at neighbour has free lock. Thus, line 10 tells the BTM of the primary replica that V_{μ_x} return 1, which means that primary gets the lock. For case $V_{\mu_x} = 1$, the Target Set is equal to 1. Therefore, line 14 tells the BTM of the primary replica that V_{μ_x} is already locked by another transaction.

3.6 Example Cases

BVAGQ-AR is designed to manage database replication in distributed database environment. By using replication, the data reliability and performance are enhanced; however, the key issue is that the consistency of the replicated data needs to be maintained because once a copy of the data is modified, other copies of the same data become inconsistent.

In this section, examples of cases that occur during real time transactions are presented. All elements of V_η and V_ψ may request instant x simultaneously at any site of $S(B)$ either at the same site or a different site. As shown in Figures 3.9–3.12, A, B, C, D and E represent the replicated servers that hold primary data a , b , c , d and e , respectively.

3.6.1 Case 1: A set of transaction V_η Request to Update Instant b at One Site

$i \in S(B)$

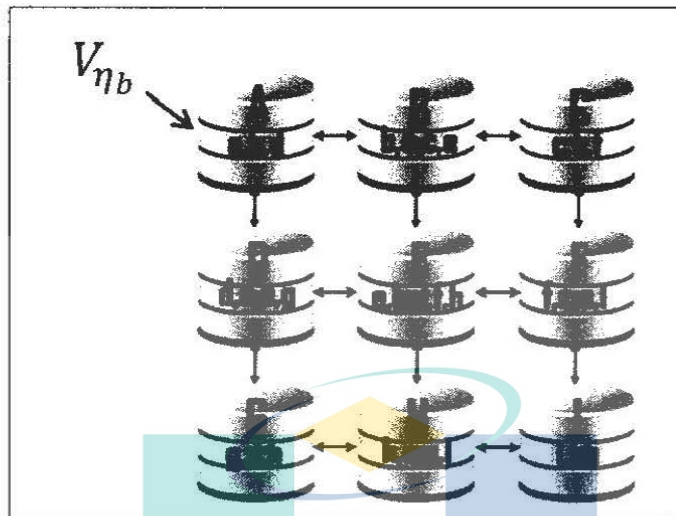


Figure 3.9 One set of transaction at one site

Figure 3.9 shows one transaction request to update data at one site. In this case, the transaction requests to update instant b . For sites $S(B = \{i \mid B(i)=1, 1 \leq i \leq n\}, V_{\lambda_b} \in V_\eta$ will get locked so that V_{λ_b} will be executed while $V_{\lambda+1_b}, \dots, V_{\lambda+q_b}$ wait to be executed after V_{λ_b} has finished. wV_{λ_b} is the write counter for V_{λ_b} that increases when V_{λ_b} gets a lock.

3.6.2 Case 2 - Different Transactions where V_η and V_ψ Request Instant b at Different Site $i \in S(B)$



Figure 3.10 Two sets of transactions at two sites

Figure 3.10 shows two transactions requests to update data at two sites. V_η and V_ψ can either come synchronously or asynchronously. Both transactions, V_{λ_b} where $\lambda = \eta, \psi$ get locked, respectively. To make it clearer, assuming that all elements V_{η_b} and V_{ψ_b} come to modify instant b at site A and C , respectively. V_{η_b} locked instant b at site A whereas V_{ψ_b} locked instant b at site C .

3.6.3 Case 3 - Different Set of Transactions V_η and V_ψ Request to Update Instant b at Same Site $i \in S(B)$.

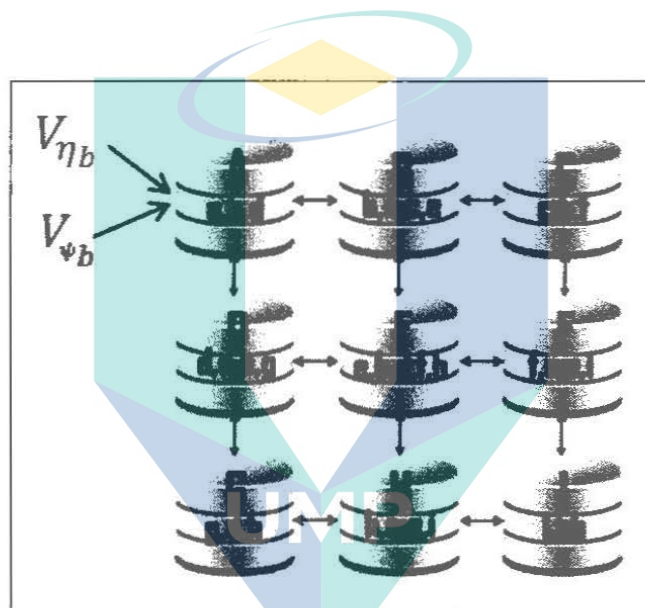


Figure 3.11 Two sets of transactions at one site

Figure 3.11 shows two sets transaction requests to update data at one site. Different sets of transactions (i.e., $V_{\lambda_b} \in V_\eta$ and $V_{\lambda_b} \in V_\psi$) may request the same instant b at site A . Therefore, $V_{\lambda_b} \in V_\psi$ is accessing a failure and is queued for the next transaction while $V_{\lambda_b} \in V_\eta$ will be executed. The existence of neighbour transaction V_{μ_b} depends on V_{λ_b} or \widehat{V}_{λ_b} . When V_{λ_b} propagates a lock to its neighbour sites $i \in S(B)$, V_{μ_b} will monitor its own status of Target Set either 0 or 1 for instant b and send a feedback to V_{λ_b} . If V_{λ_b} successfully locks the site, the write counter will be increased and it will check whether or not V_{λ_b} already gets the majority quorum. Consequently, V_{λ_b} propagates its lock to other neighbours until it gets all quorums.

3.6.4 Case 4 - A Set of Transaction V_η Request to Update an Unavailable Instant c , at Site $i \in S(B)$

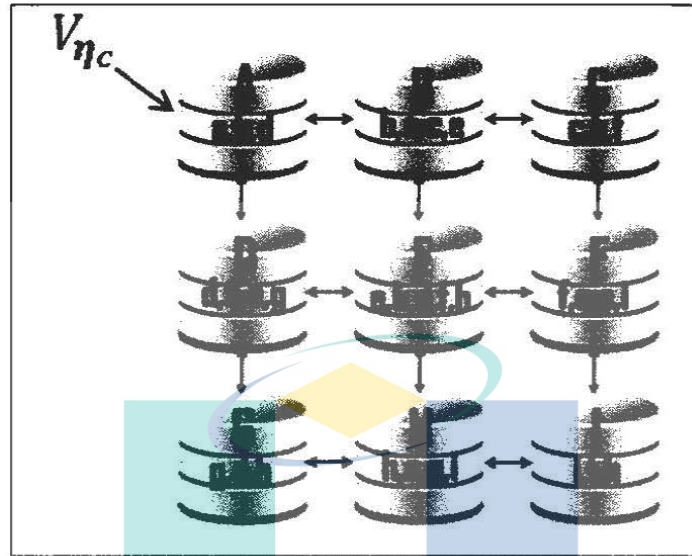


Figure 3.12 Two sets of transactions at two sites

Figure 3.12 shows one transaction request to update unavailable data at a site. In this case, the requested data is c . For sites $S(B) = \{i \mid B(i) = 1, 1 \leq i \leq n\}$, $V_{\lambda_c} \in V_\eta$ will get locked. Data c is not available at site A; hence, V_{λ_c} will check the mapping table to search for data c . Then, V_{λ_c} will get locked. After that, V_{λ_c} will be executed while $V_{\lambda+1_c}, \dots, V_{\lambda+q_c}$ wait to be executed after V_{λ_c} has finished. wV_{λ_c} is the write counter for V_{λ_c} that increases when V_{λ_c} gets a lock.

In BVAGQ-AR transaction model, when several sets of transaction requests to update the same instant x , only one transaction is executed at one time. In the case where only $V_{\lambda_b} \in V_\eta$ is requesting to modify instant b , the transaction will propagate its lock until it gets all locks from replica $i \in S(B)$. Transaction \hat{V}_{λ_b} is a transformed transaction when V_{η_b} gets all the quorums. In this case, after getting all locks from the replicas, \hat{V}_{λ_b} will change the access permission mode of the instant b and then acknowledge the client for an updating process. After the client has updated instant b , \hat{V}_{λ_b} is fragmented using vertical fragmentation into S_1 and S_2 . Then, S_1 is fragmented again using horizontal fragmentation into $S_{1(p_k,x)}$ and $S_{1(p_k,y)}$. After both fragmentations have been completed, \hat{V}_{λ_b} will commit the transaction, $S_{1(p_k,x)}$.

In cases where more than one transaction request to update data b at different sites, the first V_{λ_b} that obtains the quorum will become $V_{\lambda_b} \in V_{\eta}$ and it will be transformed into \hat{V}_{λ_b} while other $\forall V_{\lambda_b}$ will become $V_{\lambda_b} \in V_{\psi}$ and then aborted. \hat{V}_{λ_b} acknowledges the client to update the instant b , fragments using vertical and horizontal fragmentation and commits the transaction.

When V_{λ_x} gets a lock for a particular data object x , it will be propagated synchronously to neighbour sites $i \in S(B)$ until it obtains the quorum. A lock is propagated based on BVAGQ-AR Primary-Neighbours Grid Coordination (BPNGC) as shown in Table 3.4 where N stands for neighbour.

Table 3.4 BVAGQ-AR Primary-Neighbours Grid Coordination.

PRIMARY REPLICA	NEIGHBOUR REPLICAS			
	N1	N2	N3	N4
$m(i,j)$	$m(i-1,j)$	$m(i,j-1)$	$m(i,j+1)$	$m(i+1,j)$
$m(i-1,j)$	$m(i,j-1)$	$m(i,j+1)$	$m(i+1,j)$	$m(i,j)$
$m(i,j-1)$	$m(i,j+1)$	$m(i+1,j)$	$m(i,j)$	$m(i-1,j)$
$m(i,j+1)$	$m(i+1,j)$	$m(i,j)$	$m(i-1,j)$	$m(i,j-1)$
$m(i+1,j)$	$m(i,j)$	$m(i-1,j)$	$m(i,j-1)$	$m(i,j+1)$

3.7 Illustrative Examples

In this section, some illustrative examples are presented for the purpose of clarifying the algorithms. These examples show how the algorithms works in managing database mining and replication transactions.

BVAGQ-AR of logical design of 5 x 5 grid structure with the neighbours binary vote assignment: $B_x(E) = B_x(B) = B_x(D) = B_x(F) = B_x(H) = 1$ is imposed in all of the illustrative examples. Table 3.5 shows the BVAGQ-AR Primary-Neighbours Grid Coordination (BPNGC) for $S(B_x) = \{E, B, D, F, H\}$. In particular, this table is also used for all cases in this section.

Table 3.5 The BPNGC for $S(B_x) = \{E, B, D, F, H\}$.

PRIMARY REPLICA	NEIGHBOUR REPLICAS			
	N1	N2	N3	N4
E	B	D	F	H
B	D	F	H	E
D	F	H	E	B
F	H	E	B	D
H	E	B	D	F

3.7.1 Case 1 - Database Mining Management

In Case 1, a transaction, V_η , is assumed to have successfully locked replica E and its neighbour replicas. This example will describe how database mining and replication transaction are managed using BVAGQ-AR technique.

A transaction requests to access instant e from replica server E. BTM receives V_{η_e} from client at replica E. Therefore, replica E is recognized as the primary replica. BTM uses *initialize_lock* function of Listing 3 to get the lock. The first element $V_{\lambda_e} \in V_\eta$ gets the lock while aborting others. In order to lock its neighbour replicas, BTM will use database replication. After successfully locking the neighbour replicas B, D, F and H, respectively, BTM calls the algorithm in Listing 1.

Table 3.6 An example of how BVAGQ-AR handles database mining

REPLICA	E	B	D	F	H
TIME					
t1	unlock(e)	unlock(e)	unlock(e)	unlock(e)	unlock(e)
t2	Begin_transaction				
t3	write lock(e)				
t4	convert $S' \rightarrow 1,0$				
t5	discover $J_{k-1}, k++$				

Table 3.6 Continued

REPLICA	E	B	D	F	H
TIME					
t6	compare with minSupp				
t7	fragment $S' \rightarrow J_1, J_2$				
t8	eliminate J_2	lock(e) from E	lock(e) from E	lock(e) from E	lock(e) from E
t9	allocate J_1 at $S(B)^x$ ($x = 1, 2$ or 3)				
t10	allocate J_2 at any $S(B)^x$				
proceed with database replication process					

3.7.2 Case 2: Database Replication Transaction Management

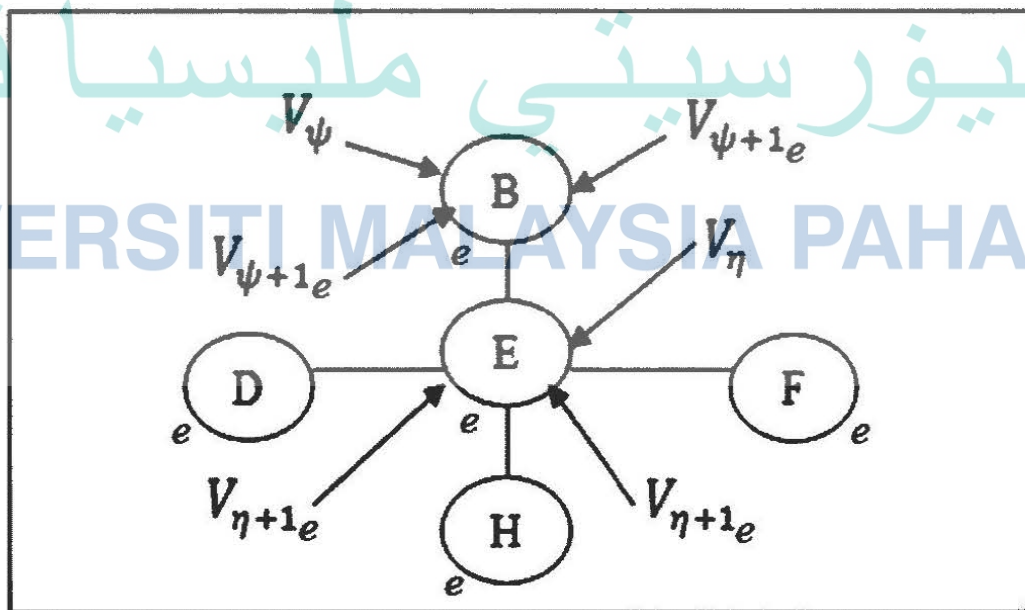


Figure 3.13: An example of BVAGQ-AR transaction processing

In Case 2, it is assumed that the different sets of transactions V_η and V_ψ request to access particular data file e , at replicas E and B, respectively. This example will describe how transaction and replication are managed using BVAGQ-AR technique. Figure 3.13 indicates an example of BVAGQ-AR transaction processing where V_η and V_ψ request to access particular data file e , at replicas E and B, respectively.

On receiving $V_{\eta_e}, V_{\eta+1_e}, \dots, V_{n+q_e}$ from clients, BTM uses Algorithm 2. Therefore, replica E is recognized as the primary replica E. Then, BTM calls Algorithm 2 and uses *initialize_lock* function of Listing 3 to determine which element among the set of transactions that gets the lock. The first element $V_{\lambda_e} \in V_\eta$ gets the lock while aborting others. $V_{\lambda_e} \in V_\eta$ propagates lock synchronously to other replicas, as shown in Table 3.4, by using the *request_lock_synchronous* function of Listing 4. Replicas B, D, F and H received $V_{\lambda_e} \in V_\eta$ through Algorithm 2. Next, the BTM calls Algorithm 3 after recognizing $V_{\lambda_e} \in V_\eta$ to be processed by the neighbour replica. Neighbour replicas B, D, F and H use Algorithm 3 for the feasibility check whether or not to perform and send a feedback to primary replica E. On receiving V_{μ_e} from neighbour replicas, once again replica E uses Algorithm 2. The BTM of replica E proceeds to Algorithm 3 to check whether or not $V_{\lambda_e} \in V_\eta$ gets the quorum. If V_{λ_e} still has not obtained the quorum, it then repeats sending request to neighbour replicas by using Algorithm 3 while neighbour replicas $i \in S(B)$ use Algorithm 2 and 3 on receiving V_{λ_e} from primary replica E.

The same procedure and algorithms as described above are used. The case 2 above is related with additional new invoking set of transactions where $V_{\psi_e}, V_{\psi+1_e}, \dots, V_{\psi+q_e}$ request to update data e , at replica B. The first primary replica that obtains majority of the quorum is denoted as \hat{V}_{λ_e} and proceeds with the transaction. If $V_{\lambda_e} \in V_\eta$ from primary replica E first arrives at replica D, thus $V_{\lambda_e} \in V_\eta$ obtains majority of the quorum. Therefore, $V_{\lambda_e} \in V_\eta$ of replica E becomes \hat{V}_{λ_e} .

Primary replica E proceeds with Algorithm 3 and aborts $V_{\lambda_e} \in V_\psi$ of primary replica B. $V_{\lambda_e} \in V_\eta$ of primary replica B imposes Algorithm 2 to release the lock, then rollbacks and sets its lock. Next, primary replica E proceeds with \hat{V}_{λ_e} through Algorithm 3, acknowledges the client and waits for the user to finish updating. After that, \hat{V}_{λ_e}

fragments the relation to get $S_{1(p_k,e)}$ using vertical fragmentation which is then followed by horizontal fragmentation through Algorithm 2. Finally, primary replica E and all its neighbour replicas synchronously commit by \hat{V}_{λ_e} using Algorithm 2. Table 3.7 shows an example of how BVAGQ-AR handles the concurrent transactions $V_{\lambda_e} \in V_\eta$ and $V_{\lambda_e} \in V_\psi$ at primary replica E and B.

Table 3.7 An example of how BVAGQ-AR handle concurrent transactions.

REPLICA/ TIME	E	B	D	F	H
t1	unlock(e)	unlock(e)	unlock(e)	unlock(e)	unlock(e)
t2	begin transaction				
t3	write lock(e) counter_w(e)=1	begin transaction			
t4	wait	write lock(e) counter_w(e)=1			
t5	propagate lock:B				
t6	propagate lock:D	propagate lock:D			
t7			lock(e) from E		
t8	get lock:7 counter_w(e)=2	propagate lock:F			
t9	propagate lock:F			lock(e) from B	
t10	propagate lock:H	get lock:F counter_w(e)=2			
t11		propagate lock:H			lock(e) from E
t12	get lock:H counter_w(e)=3	propagate lock:E			
t13	obtain quorum release lock:3	propagate lock:H			
t14		abort $V_{\lambda_e} \in V_\psi$ & rollback, lock(e) from 8		rollback, lock(e) from E	
t15	update e				

Table 3.7 Continued

REPLICA/ TIME	E	B	D	F	H
t16	fragment S' into S'_1 and S'_2 S'_1 is fragmented				
t17	into $S'_{1(p_k,x)}$ and $S'_{1(p_k,y)}$				
t18	commit $\hat{V}_{\lambda_e} \in V_\eta$	commit $\hat{V}_{\lambda_e} \in V_\eta$	commit $\hat{V}_{\lambda_e} \in V_\eta$	commit $\hat{V}_{\lambda_e} \in V_\eta$	commit $\hat{V}_{\lambda_e} \in V_\eta$
t19	unlock(e)	unlock(e)	unlock(e)	unlock(e)	unlock(e)

3.8 Correctness

Assertion: If the transaction gets all locks from replicas $i \in S(B_x)$, then the transaction will be executed successfully.

Proof: The only way that a transaction gets a lock in initiate lock is when $V_{\lambda_x} = 1$ with $V_{\lambda_x} \in V_\eta$. After $V_{\lambda_x} \in V_\eta$ is successful in initiating lock at a server, then, $V_{\lambda+1_x}, \dots, V_{\lambda+q_x}$ which are the elements that exist in V_η will be queued. To get majority of the quorum, $wV_{\lambda_x} \geq \left\lceil \frac{n}{2} \right\rceil$ is required. This means that the primary server needs to get majority of the locks from its neighbour servers by calling request lock from the neighbour servers. Each of the neighbours $i \in S(B)$ will send a feedback to the primary server to notify it whether or not each neighbour server is in free lock. If the primary gets majority of the locks of instant x , this means that $\hat{V}_{\lambda_x} = V_{\mu_x} = V_\eta$ where \hat{V}_{λ_x} gets a quorum. Next, the primary will send error notification to other neighbours $i \in S(B)$ in the quorum. Consequently, when $V_{\lambda_x} \in V_\psi$ releases its lock, $V_{\lambda_x} \in V_\eta$ gets the lock from every neighbour $i \in S(B_x)$. After V_{λ_x} gets majority quorum, relation S is fragmented into S_1 and S_2 using vertical fragmentation. S_1 is then fragmented again into $S_{1(p_k,x)}$ and $S_{1(p_k,y)}$ but this time, using horizontal fragmentation. When the user finishes updating the instant, \hat{V}_{λ_x} commits (sends the fragmented data) to \forall neighbour $i \in$

$S(B_x)$. Therefore, all replicas of $S(B_x)$ will perform and execute the transaction successfully.

3.9 Summary

In this chapter, a technique called BVAGQ-AR that combines data mining and data replication in managing database replication in DDS has been described. In this technique, a preliminary database is mined before fragmented. After that, the fragmented database is replicated to its neighbour sites so that replication can proceed. Users will be able to update an instant only after their transactions succeed in obtaining majority of the quorum. After a user updates an instant, the relation with that instant is fragmented using vertical and horizontal fragmentation. After the fragmentation, the values that are left are only the primary key and the instant that needs to be updated. Finally, the transaction is committed to all sites. A series of experiments have been run in order to provide proof in support of this theory and to show that this technique is the best compared to existing techniques which have been reviewed in Chapter 2 of this thesis. Details of the experimental results are discussed in Chapter 4.

The logo of Universiti Malaysia Pahang (UMP) is a large, downward-pointing triangle. It is divided into four quadrants by a vertical and a horizontal line. The top-left quadrant is light blue, the top-right is light green, the bottom-left is light purple, and the bottom-right is light yellow. The letters 'UMP' are written in white, bold, sans-serif font across the center of the triangle.

UMP

اونيورسيتي مليسيا قهغ

UNIVERSITI MALAYSIA PAHANG

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter details the implementation of the synchronous database replication which describes the algorithms and example of cases as described in Chapter 3 of this thesis. In addition, this chapter also provides evidence that BVAGQ-AR has been able to mine useful knowledge from huge database and also support database fragmentation in order to allocate the database at servers. It also presents evidence that supports synchronous replication process for the database. In the experiments conducted, a transaction updating a tuple in a relation has been considered, in which the replication manager locking only the relation containing such tuple. Finally, the chapter compares and contrasted the results of the present study which has employed the BVAGQ-AR technique with other replication techniques.

4.2 Hardware and Software Specifications

From the perspective of users, the functionality offered by BVAGQ-AR application is for database replication. Nevertheless, for this implementation, BVAGQ-AR application is tested in different sites under the local area network (LAN). The implementation of BVAGQ-AR requires minimum hardware and software specifications. Each replication server component has its hardware specification as shown in Table 4.1.

Table 4.1 Server component specifications

Hardware	Specifications
Processor	Intel (R) Core ((TM) 2 Quad CPU Q9650 @3.00 GHz 2.99 GHz
Memory	4.00 Gigabytes
Cache	3624 Megabyte
Hard Disk	300 Gigabytes
Chip Set	ATI Radeon HD 3450- Dell Optiplex
Network Card	Intel (R) 82567 LM-3 Gigabit Network Connection

The implementation of the BVAGQ-AR Transaction Manager (BTM) was carried out by using PHP scripting language and deployed in Windows 8.1 Professional. Table 4.2 shows the system development tools specifications for this implementation.

Table 4.2 System development tools specifications

System Development Software	Specifications
PHP	Version 5.6
SQLyog	Community Edition 12.4.3
MySQL Server	Version 5.6
.Net Framework	Version 2.0 Redistributable Package (x64)
Sql connector	Version 5.1.42
Windows 8.1	Professional
Wamp Server	Version 3.0.6 64 bits

The experiments were executed in three and five replication servers. The minimum and maximum numbers of replication servers used in the present research were three and five, respectively. This is because they were the minimum and maximum numbers of replication servers in BVAGQ-AR. In order to evaluate the performance of the proposed architecture, BVAGQ-AR Replication Manager was

developed. This tool was used for calculation of the job execution time and comparison of results with existing techniques.

4.3 BVAGQ-AR Experimental Results

In this section, the experiments for managing transaction and replication are described. The experiments involve the phases that have been described in Section 3.4, Chapter 3 of this thesis. Different cases were considered in the experiments.

To demonstrate BVAGQ-AR transaction, nine servers logically organized in 3×3 were considered based on BVAGQ-AR two-dimensional logical design. A total of nine servers were used because the number of replicated data, d , can be 3, 4 or 5. Hence, a total of nine servers were chosen in order to get maximum replicated data, $d = 5$ in the experiments. Five replication servers were deployed as shown in Figure 4.1. Each server or node was connected to one another through a fast Ethernet switch hub. There were two parts of the experiments. The first part of the experiment involved data mining, data fragmentation and data allocation process while the second part involved data replication. Five cases of experiments for the replication process were conducted. These cases will be explained later in Sub-sections 4.3.1–4.3.6 of this chapter.

Theoretically, each of the neighbour replication servers and the primary replication server should be connected to each other logically as shown in Figure 4.1. Each server was assigned with vote 0 or 1. Vote 0 means the server would be free and therefore unlocked; hence, the server would be able to proceed with a new transaction. By contrast, vote 1 means the server would be busy; in other words, the server would already be locked. Hence, new transaction cannot be initiated on that server.

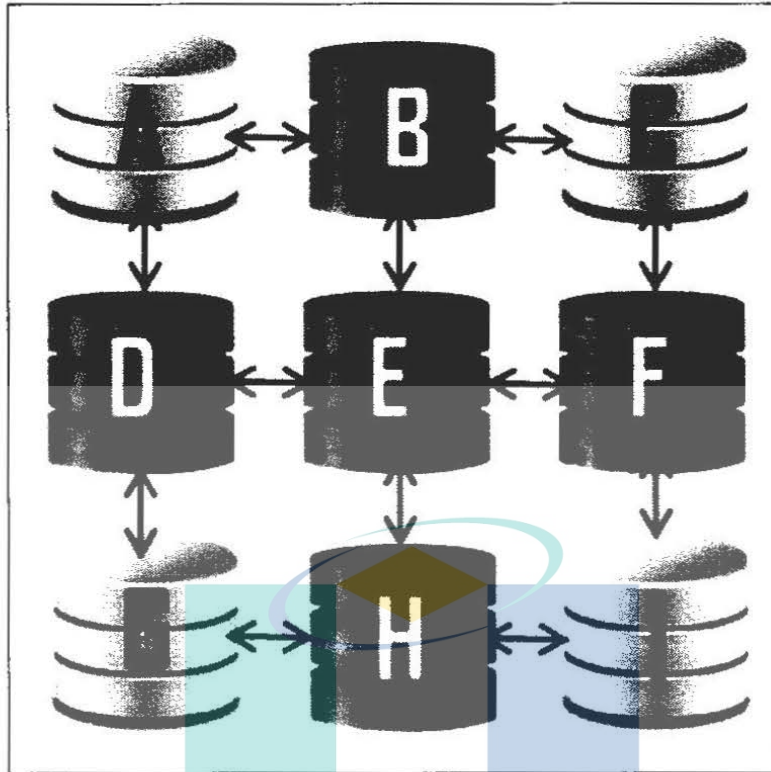


Figure 4.1 Five replication servers connected to each other.

The Binary Vote Grid Coordination is presented in Table 4.3. Replica B with IP 172.21.202.163, replica D with IP 172.21.202.162, replica E with IP 172.21.202.169, replica F with IP 172.21.202.168 and replica H with IP 172.21.202.2167.

Table 4.3 BVAGQ-AR Grid Coordination

Primary		Neighbours		
B:	D:	E:	F:	H:
172.21.202.163	172.21.202.162	172.21.202.169	172.21.202.168	172.21.202.167
D:	E:	F:	H:	B:
172.21.202.162	172.21.202.169	172.21.202.168	172.21.202.167	172.21.202.163
E:	F:	H:	B:	D:
172.21.202.169	172.21.202.168	172.21.202.167	172.21.202.163	172.21.202.162
F:	H:	B:	D:	E:
172.21.202.168	172.21.202.167	172.21.202.163	172.21.202.162	172.21.202.169
H:	B:	D:	E:	F:
172.21.202.167	172.21.202.163	172.21.202.162	172.21.202.169	172.21.202.168

4.3.1 Experiment 1: Mining S to Identify J_1 and the $S(B)^e$ for S'

In Experiment 1, the first part of the process was tested. The processes involved in this experiment were data mining, data fragmentation and data allocation. The aim of the first part of this experiment was to mine the preliminary data in order to obtain the frequent itemsets. The frequent itemsets were then fragmented and allocated to the neighbour servers. When meaningful data were allocated together, this would help the transaction to access the data easily. Hence, the time needed to complete a transaction would be shorter. This experiment recorded the BVAGQ-AR Replication Manager execution time to identify J_1 , fragmented the S according to S and allocated the data to $S(B)^e$ where $e = 1, 2$ or 3 . The result captured from BVAGQ-AR tool is shown in Figure 4.2.

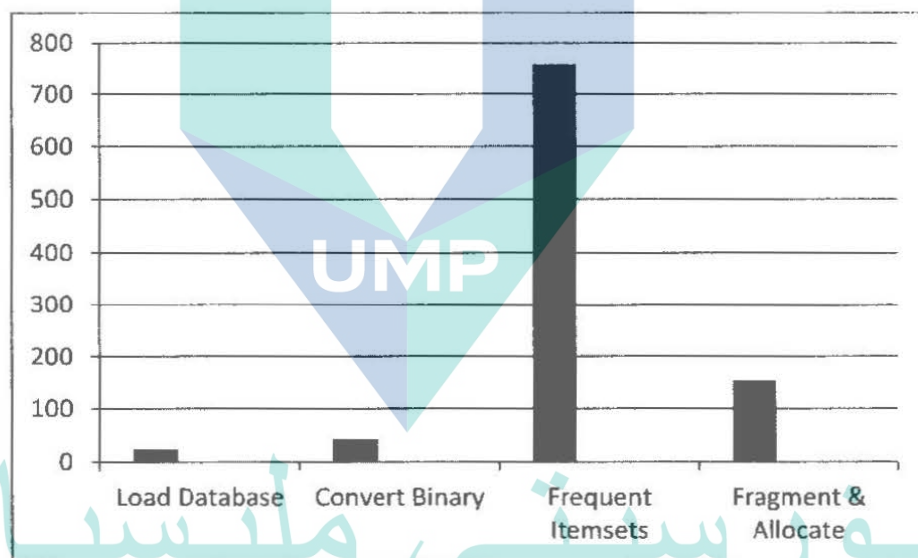


Figure 4.2 Execution time for Experiment 1.

Figure 4.2 shows the execution time for Experiment 1. The execution time to load S was 23.543 milliseconds (ms). Meanwhile, it took 41.544 ms to convert the S into binary format and 758.132 milliseconds to execute Apriori algorithm in association rule to identify S' , respectively. Finally, the execution time to fragment S' into J^1 and J^2 as well as the allocation of J^1 in $S(B)^x$ was 153.422 milliseconds. The total job execution time for the first part of Experiment 1 was 976.641 ms.

Table 4.4 Mining S to Identify J_1 and the $S(B)^e$ for S'

REPLICA	E	B	D	F	H
TIME					
t1	unlock(e)	unlock(e)	unlock(e)	unlock(e)	unlock(e)
t2	begin transaction				
t3	write lock(e)				
t4	convert $S \rightarrow 1,0$				
t5	discover $J_{k-1}, k++$				
t6	compare with minSupp				
t7	fragment $S' \rightarrow J_1, J_2$				
t8	eliminate J_2				
t9		lock(e) from E	lock(e) from E	lock(e) from E	lock(e) from E
t10	Obtain all lock				
t11	allocate J_1 at $S(B)^e$ ($e = 1, 2$ or 3)				
t12	allocate J_2 at any $S(B)^e$				
proceed with database replication process					

From Table 4.4, at time equal to 1 ($t1$), instant e at all servers were unlocked. At $t2$, the transaction began at server E. At $t3$, server E was locked by a transaction. Hence, write counter for server E was equivalent to 1. At $t4$, a preliminary database, S was loaded in server E and converted into binary format. Apriori algorithm was executed to discover $J_{k-1}, k++$ at $t5$. After the discovery of the frequent itemsets, S' was fragmented into J_1 and J_2 . At $t9$, server E propagated its lock at all of its neighbour servers. After obtaining all locks from its neighbour servers at $t10$, J_1 was allocated at $S(B)^e$ where $e = 1, 2$ or 3 . At $t12$, J_2 was allocated at any available $S(B)^e$. A time diagram about the whole process for these five replication servers is shown in Figure 4.3.

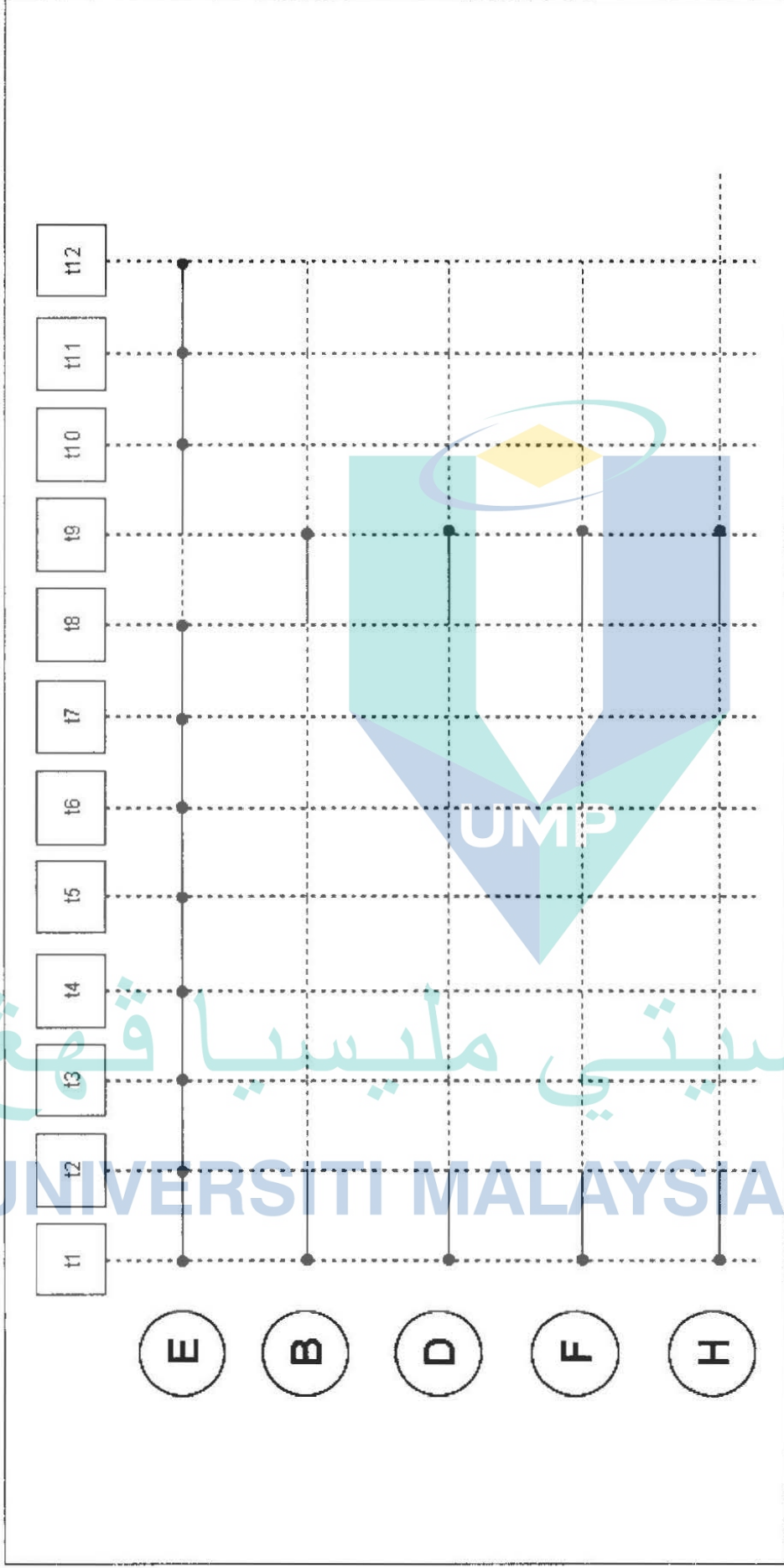


Figure 4.3 Time diagram for Mining S to Identify J1 and the S(B)e for S'

4.3.2 Experiment 2: A Transaction V_{η} Request to Update Instant e at Site E

In Experiment 2, the replication part of the technique was tested. The goal of this experiment was to solve Case 1 as described in Sub-section 3.6.1, Chapter 3 of this thesis. For this experiment, a transaction, V_{η} , requested to update instant e at site E . The aim of this experiment was to check if BVAGQ-AR would be able to manage a synchronous database replication through the transaction for this case and to record the job execution time for the replication process of this case. The result captured from BVAGQ-AR tool is shown in Figure 4.4.

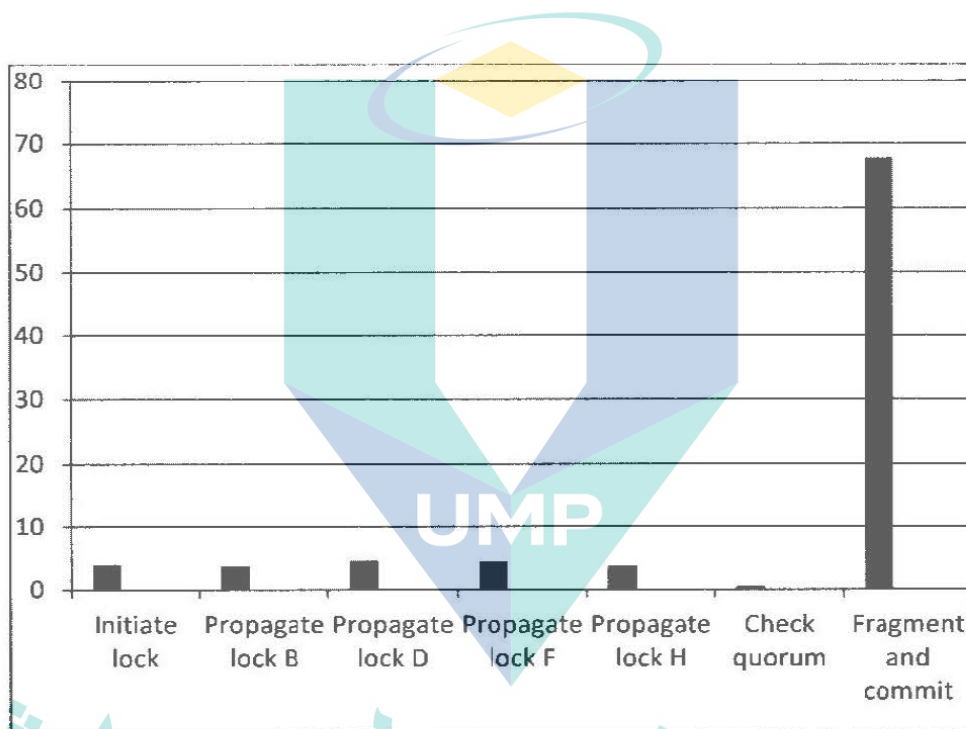


Figure 4.4 Execution time for Experiment 2

Figure 4.4 shows the execution time for Experiment 2. The time taken for V_{η} to initiate lock at E was 3.905 milliseconds (ms). Meanwhile, it took 3.691 ms, 4.472 ms, 4.371 ms, and 3.675 ms to propagate lock to server B, server D, server F and server H, respectively. Execution time to check quorum was 0.448 ms. Finally, the time taken to fragment the database and commit the transaction was 67.712 ms. Hence, the total job execution time for Experiment 2 was 88.274 ms.

Table 4.5 Experimental results for one transaction at one site

REP-LICA	E	B	D	F	H
TIME					
t1	unlock(<i>e</i>)	unlock(<i>e</i>)	unlock(<i>e</i>)	unlock(<i>e</i>)	unlock(<i>e</i>)
t2	begin_transac tion	begin_tra nsaction	begin_transac tion	begin_transac tion	begin_transac tion
t3	V_{η_e} write lock(<i>e</i>), counter_w(<i>e</i>) =1				
t4	V_{η_e} pro- pagate lock:B				
t5		V_{η_e} lock(<i>e</i>) from E			
t6	V_{η_e} get lock:B, counter_w(<i>e</i>) =2				
t7	V_{η_e} propagate lock:D				
t8			V_{η_e} lock(<i>e</i>) from E		
t9	V_{η_e} get lock:D, V_{η_e} counter_w(<i>e</i>)=3				
t10	V_{η_e} propagate lock:F				
t11				V_{η_e} lock(<i>e</i>) from E	
t12	V_{η_e} get lock:F, counter_w(<i>e</i>) =4				
t13	V_{η_e} propagate lock:H				
t14					V_{η_e} lock(<i>e</i>) from E
t15	V_{η_e} get lock:H, counter_w(<i>e</i>) =5				
t16	V_{η_e} obtain quorum				
t17	V_{η_e} update <i>e</i>				

Table 4.5 Continued

REPLI CA/ TIME	E	B	D	F	H
t18	S is fragmented into S_1 and S_2				
t19	S_1 is fragmented into $S_{1(Pk,x)}$ and $S_{1(Pk,y)}$				
t20	commit $\hat{V}_{\lambda_e} \in V_\eta$	commit $\hat{V}_{\lambda_e} \in V_\eta$	commit $\hat{V}_{\lambda_e} \in V_\eta$	commit $\hat{V}_{\lambda_e} \in V_\eta$	commit $\hat{V}_{\lambda_e} \in V_\eta$
t21	unlock(e)	unlock(e)	unlock(e)	unlock(e)	unlock(e)

As shown in Table 4.5, at time equivalent to 1 ($t1$), instant e at all servers were unlocked. At $t2$, the transaction began.

At $t3$, there was a transaction, V_{η_e} that requested to update instant e at server E . The transaction initiated lock. Hence, write counter for server E was now equal to 1.

At $t4$, V_{η_e} propagated lock at its neighbour server B , V_{η_e} lock(e) from server E . Thus at $t6$, the transaction achieved in getting locked from server B which then resulted in write quorum to be equal to 2.

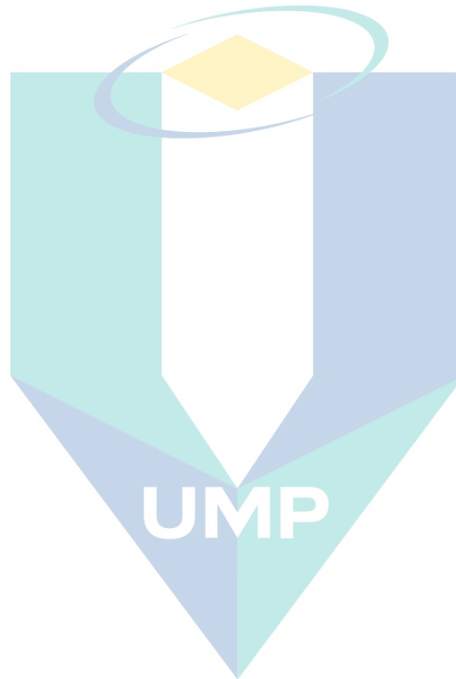
Next, V_{η_e} propagated lock at server D at $t7$ and at $t8$, V_{η_e} locked (e) from E . Thus, at $t9$, the transaction achieved in getting locked from server D which then resulted in write quorum to be equivalent to 3.

After that, V_{η_e} propagated lock at server F at $t10$ and at $t11$, V_{η_e} locked (e) from server F . Thus, at $t12$, the transaction achieved in getting locked from the server F which then resulted in write quorum to be equivalent to 4.

Then, V_{η_e} propagated lock at server H at $t13$ and at $t14$, V_{η_e} locked (e) from H . Thus at $t15$, the transaction achieved in getting locked from server H then resulted in write quorum to be equal to 5.

At $t16$, V_{η_e} obtained all quorums and then instant e was updated at $t17$. At $t18$, the relation S was fragmented into S_1 and S_2 using vertical fragmentation. At $t19$, the relation S_1 was fragmented again using horizontal fragmentation into $S_{1(p_k,x)}$ and $S_{1(p_k,y)}$.

Finally, at $t20$, $\hat{V}_{\lambda_e} \in V_{\eta}$ was committed and at $t21$, instant e at all replica servers would be unlocked and were ready for the next transaction to take place. A time diagram about the whole process for these five replication servers is illustrated in Figure 4.5.



اونيورسيٲي ملايسيا قهغ

UNIVERSITI MALAYSIA PAHANG

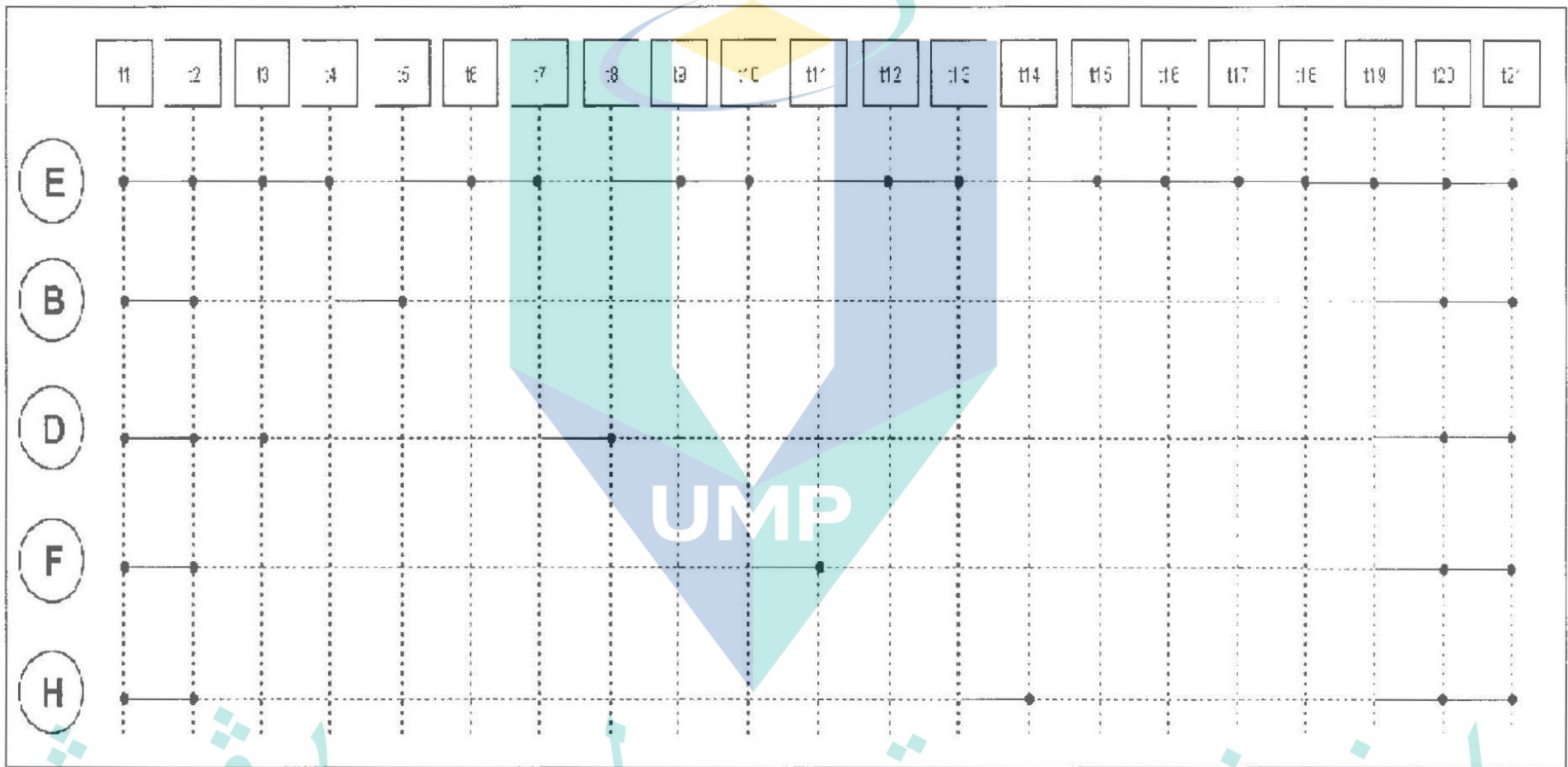


Figure 4.5 Time diagram when a set of transaction V_η request to update instant e at site E

4.3.3 Experiment 3: Different Sets of Transactions, V_η and V_ψ Request Instant e at Different Site

The goal of Experiment 3 was to solve Case 2 as described in Sub-section 3.6.2, Chapter 3 of this thesis. For this experiment, V_η and V_ψ requested to update instant e at two different servers, E and F , at the same time. The aim of this experiment was to check if BVAGQ-AR would be able to manage a synchronous database replication through the transaction for this case and to record the job execution time for the replication process of this case. The result captured from BVAGQ-AR tool is shown in Figure 4.6.

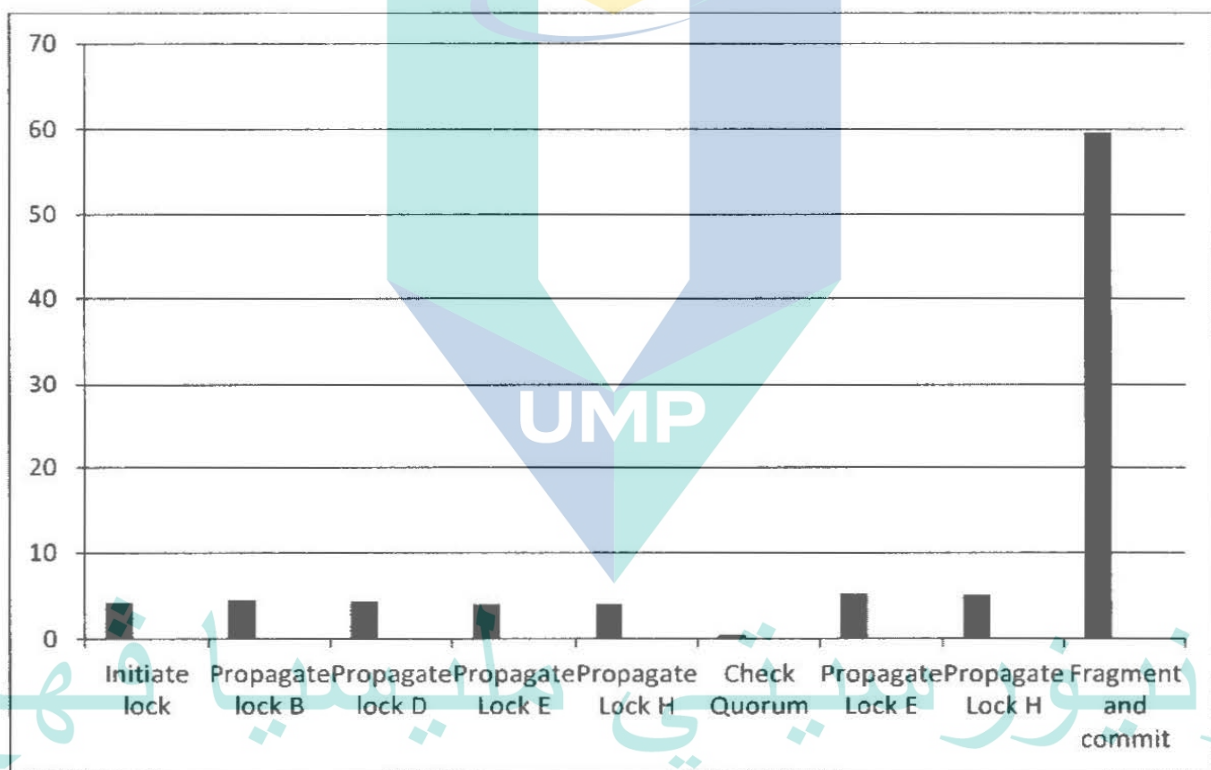


Figure 4.6 Execution time for the Experiment 3

Figure 4.6 shows the execution time for the primary server E . V_η took 4.168 ms to initiate lock. The time taken for V_η to propagate its lock at server B and server D were 4.454 ms and 4.311 ms, respectively. The system failed to propagate lock at server E and server H because V_ψ already locked the servers. The time taken to check quorum was 0.394 ms. Because V_η had obtained majority of the quorum, V_ψ released its lock.

After that, server V_{η} successfully propagated its lock at server E and server H, which took 5.145 and 5.103 ms, respectively. Finally, the time taken to fragment the database and commit the transaction was 63.551 ms. Therefore, the total job execution time for this experiment was 86.126 ms.

Table 4.6 Experimental results for two transactions updating the same data at two sites

REP-LICA TIME	E	B	D	F	H
t1	unlock(e)	unlock(e)	unlock(e)	unlock(e)	unlock(e)
t2	begin_transaction	begin_transaction	begin_transaction	begin_transaction	begin_transaction
t3	V_{η_e} write lock(e), counter_w(e)=1			V_{ψ_e} write lock(e), counter_w(e)=1	
t4	V_{η_e} propagate lock:H			V_{ψ_e} propagate lock:B	
t5		V_{ψ_e} lock(e) from F			V_{η_e} lock(e) from E
t6	V_{η_e} get lock:H, counter_w(e)=2			$V_{\psi_{e,q_1}}$ get lock:H, counter_w(e)=2	
t7	V_{η_e} propagate lock:F			V_{ψ_e} propagate lock:D	
t8	V_{η_e} fail to get lock:F, counter_w(e)=2		V_{ψ_e} lock(e) from F		
t9	V_{η_e} propagate lock:B			V_{ψ_e} get lock:D, counter_w(e)=3, obtain majority quorum	
t10	V_{η_e} fail to get lock:B, counter_w(e)=2				
t11	V_{η_e} release lock				V_{η_e} release lock
t12					V_{ψ_e} lock(e) from F

Table 4.6 Continued

REPL ICA/ TIME	E	B	D	F	H
t13				V_{ψ_e} get lock:E and H, counter_w (e)=5	
t14				update e	
t15				S is fragmented into S_1 and S_2	
t16				S_1 is fragmented into $S_{1(p,k,x)}$ and $S_{1(p,k,y)}$	
t17	commit \hat{V}_{λ_e} $\in V_{\psi_e}$	commit \hat{V}_{λ_e} $\in V_{\psi_e}$	commit \hat{V}_{λ_e} $\in V_{\psi_e}$	commit \hat{V}_{λ_e} $\in V_{\psi_e}$	commit \hat{V}_{λ_e} $\in V_{\psi_e}$
t18	unlock(e)	unlock(e)	unlock(e)	unlock(e)	unlock(e)

As shown in Table 4.6, at time equivalent to 1 ($t1$), instant e at all servers were unlocked. At $t2$, the transaction began.

At $t3$, there were two transactions, V_{η_e} and V_{ψ_e} that requested to update instant e at the same time. Both of the transactions initiated lock. At this time, the target set for each server had changed to 1 which means that the server was busy. Hence, write counter for both server E and server F was now equivalent to 1.

At $t4$, V_{η_e} propagated lock at its neighbour replica H and V_{ψ_e,q_1} propagated lock at its neighbour replica B . Because target set for both server H and server B was 0, the servers were free for the transaction.

At $t5$, V_{η_e,q_1} locked (e) from E at server H , and V_{ψ_e,q_1} locked (e) from server F at server B . Thus at $t6$, both transactions obtained the lock from their neighbours. Hence, the write quorum for each transaction was equal to 2.

Next, $V_{\eta_{e,q_1}}$ propagated lock at server F at $t7$. At the same time, $V_{\psi_{e,q_1}}$ propagated lock at server D . Because $V_{\psi_{e,q_1}}$ had already locked instant e at server F , the target set for the server was now equal to 1. Hence, $V_{\eta_{e,q_1}}$ failed to obtain lock from server F . At $t9$, $V_{\psi_{e,q_1}}$ locked (e) from server D and write counter for $V_{\psi_{e,q_1}}$ was now equal to 3.

Then, at $t10$, $V_{\psi_{e,q_1}}$ obtained majority of the quorum. Thus, at $t11$ $V_{\eta_{e,q_1}}$ released its lock from server E and server H . At $t12$, $V_{\psi_{e,q_1}}$ locked (e) from server F at server E and H and write counter for $V_{\psi_{e,q_1}}$ now was equal to 5. Therefore, instant e was updated at server F at $t14$.

At $t15$, the relation S was fragmented into S_1 and S_2 using vertical fragmentation. The relation S_1 was then fragmented again at $t16$ into $S_{1(Pk,x)}$ and $S_{1(Pk,y)}$ using horizontal fragmentation.

Finally, at $t17$, $\hat{V}_{\lambda_{e,q_1}} \in V_{\psi_e}$ was committed to all sites and at $t18$, instant e at all replica servers would be unlocked and ready for the next transaction to take place. A time diagram about the whole process for these five replication servers is presented in Figure 4.7.

اونيورسيتي مليسيا قهغ

UNIVERSITI MALAYSIA PAHANG

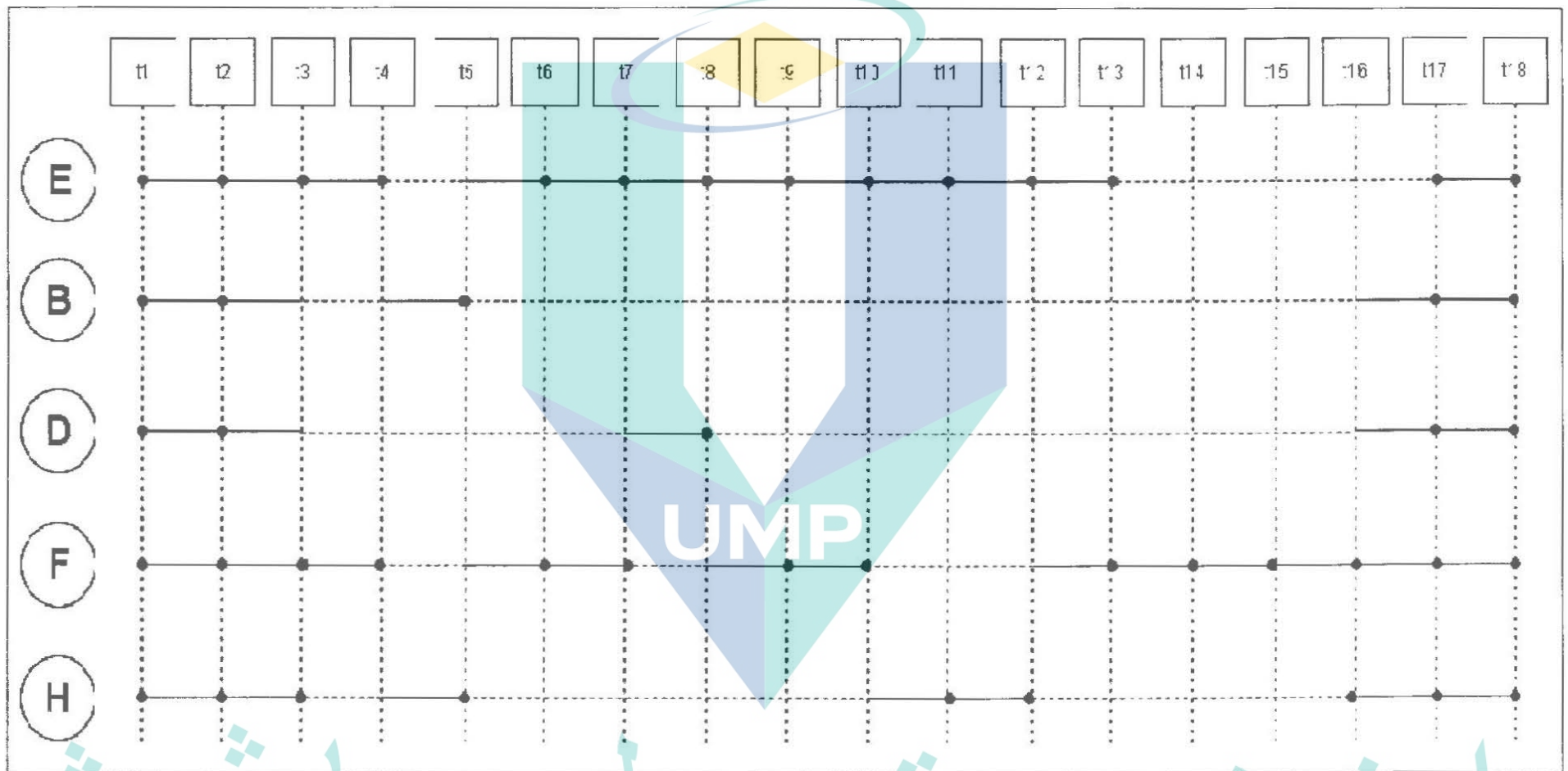


Figure 4.7 Time diagram when different sets of transactions, V_η and V_ψ request instant e at different site

4.3.4 Experiment 4: Different Sets of Transactions, V_η and V_ψ Request Instant e at the Same Site.

The goal of Experiment 4 was to solve Case 3 as described in Sub-section 3.6.3, Chapter 3 in this thesis. For this experiment, V_η and V_ψ requested to update instant e at the same server, E , at the same time. The aim of this experiment was to check if BVAGQ-AR would be able to manage a synchronous database replication through the transaction for this case and to record the job execution time for the replication process of this case. The result captured from BVAGQ-AR Replication Manager is shown in Figure 4.8.

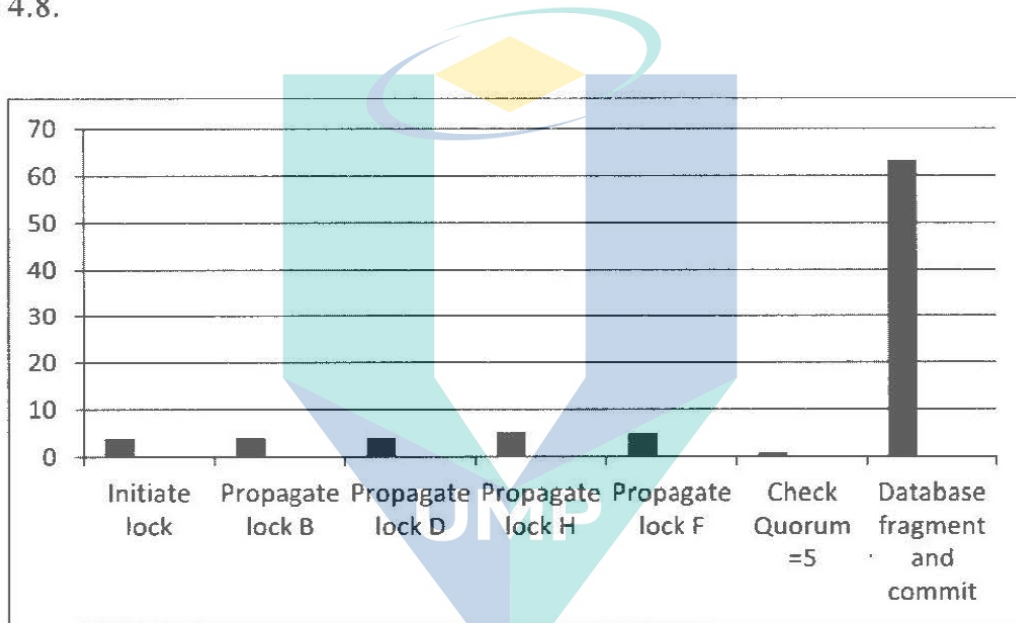


Figure 4.8 Execution time for Experiment 4

Figure 4.8 shows the execution time for the primary server E . V_η took 3.833 ms to initiate lock. It took Experiment 4 longer than the other experiments because at this time, another transaction, V_ψ was attempting to lock server E at the same time. The first transaction to arrive, V_η , would successfully lock the server. The time taken to propagate lock to server B, server D, server H, and server F were 3.983 ms, 3.972 ms, 5.082 ms, and 4.593 ms, respectively. It took 0.728 ms to check quorum. Finally, the time taken to fragment the database and commit the transaction was 63.918 ms. Therefore, the total job execution time for this experiment was 86.109 ms.

Table 4.7 Experimental result for two transactions at the same server

REPL ICA/ TIME	E	B	D	F	H
t1	unlock(e)	unlock(e)	unlock(e)	unlock(e)	unlock(e)
t2	begin_ transaction	begin_ transaction	begin_ transaction	begin_ transaction	begin_ transaction
t3	V_{η_e} write lock(e), counter_w(e) =1 V_{ψ} write lock(e), failed.				
t4	V_{η_e} propagate lock:B				
t5		V_{η_e} lock(e) from E			
t6	V_{η_e} get lock:B, counter_w(e) =2				
t7	V_{η_e} propagate lock:D				
t8			V_{η_e} lock(e) from E		
t9	V_{η_e} get lock:D, V_{η_e} counter_ w(e)=3				
t10	V_{η_e} propagate lock:H				
t11					V_{η_e} lock(e) from E
t12	V_{η_e} get lock:H, counter_w(e) =4				
t13	V_{η_e} propagate lock:F				
t14				V_{η_e} lock(e) from E	
t15	V_{η_e} get lock:F, counter_w(e) =5				

Table 4.7 Continued

REPLICA/ TIME	E	B	D	F	H
t16	V_{η_e} obtain quorum				
t17	V_{η_e} update e				
t18	S is fragmented into S_1 and S_2				
t19	S_1 is fragmented into $S_{1(Pk,x)}$ and $S_{1(Pk,y)}$				
t20	commit $\hat{V}_{\lambda_e} \in V_{\eta}$	commit $\hat{V}_{\lambda_e} \in V_{\eta}$	commit $\hat{V}_{\lambda_e} \in V_{\eta}$	commit $\hat{V}_{\lambda_e} \in V_{\eta}$	commit $\hat{V}_{\lambda_e} \in V_{\eta}$
t21	unlock(e)	unlock(e)	unlock(e)	unlock(e)	unlock(e)

As shown in Table 4.7, at time equivalent to 1 ($t1$), instant e at all servers were unlocked. At $t2$, the transaction began. At $t3$, there were two transactions, V_{η_e} and V_{ψ} , requested to update instant e at server E . Because transaction V_{η_e} successfully initiated lock, write counter for server E was now equivalent to 1. When the write counter was 1, V_{ψ} could not proceed with the transaction and would be aborted.

At $t4$, V_{η_e} propagated lock at its neighbour replica B and at $t5$, V_{η_e} locked (e) from server E . Thus, at $t6$, the transaction achieved in getting locked from server B, which then resulted in write quorum to be equal to 2.

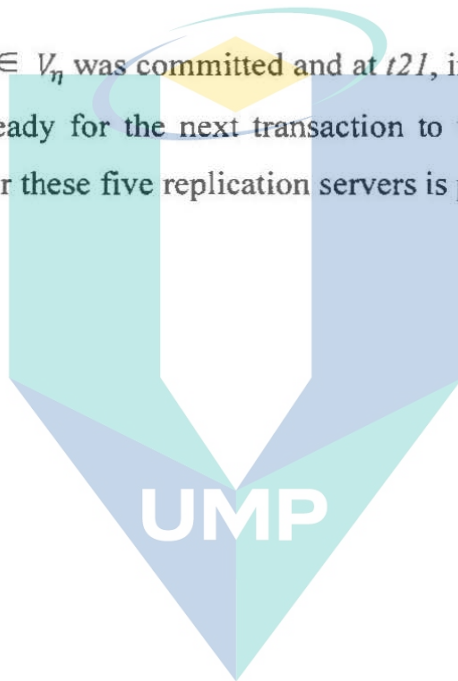
Next, V_{η_e} propagated lock at server D at $t7$ and at $t8$, V_{η_e} locked (e) from E . Thus, at $t9$, the transaction achieved in getting locked from server D , which then resulted in write quorum to be equivalent to 3.

After that, V_{η_e} propagated lock at server H at $t10$ and at $t11$, V_{η_e} locked (e) from server H . Thus, at $t12$, the transaction achieved in getting locked from server H and the write quorum was equivalent to 4.

Then, V_{η_e} propagated lock at server F at $t13$ and at $t14$, V_{η_e} locked (e) from server F . Thus, at $t15$, the transaction achieved in getting locked from server F and the write quorum was equivalent to 5.

At $t16$, V_{η_e} obtained all quorums and then instant e was updated at $t17$. At $t18$, the relation S was fragmented into S_1 and S_2 using vertical fragmentation. At $t19$, the relation S_1 was fragmented again into $S_{1(PK,x)}$ and $S_{1(PK,y)}$ using horizontal fragmentation.

Finally, at $t20$, $\hat{V}_{\lambda_e} \in V_{\eta}$ was committed and at $t21$, instant e at all replica servers would be unlocked and ready for the next transaction to take place. A time diagram about the whole process for these five replication servers is presented in Figure 4.9.



اونيورسيتي مليسيا قهغ

UNIVERSITI MALAYSIA PAHANG

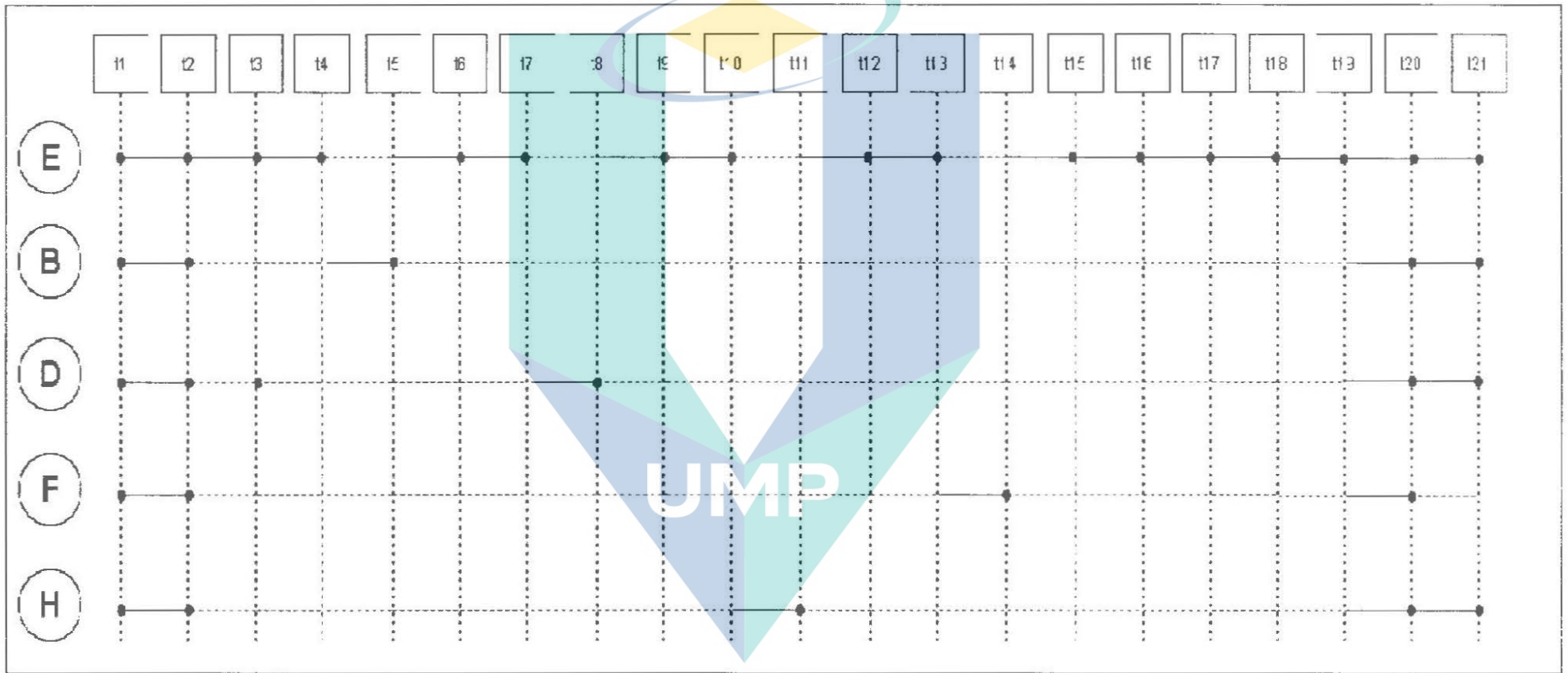


Figure 4.9 Time diagram when different sets of transactions, V_η and V_ψ request instant e at the same site

اونيور سيثي ملايسيا قهغ

UNIVERSITI MALAYSIA PAHANG

4.3.5 Experiment 5: A Transaction V_{η} Request to Update an Unavailable Instant, i at a Site

The goal of Experiment 5 was to solve Case 4 as described in Sub-section 3.6.4, Chapter 3 of this thesis. For this experiment, the case tested was when a transaction, V_{η} , requested to update instant i at server E . The aim of this experiment was to check if BVAGQ-AR would be able to manage a synchronous database replication through the transaction for this case and to record the job execution time for the replication process of this case. The result captured from BVAGQ-AR Replication Manager is shown in Figure 4.10.

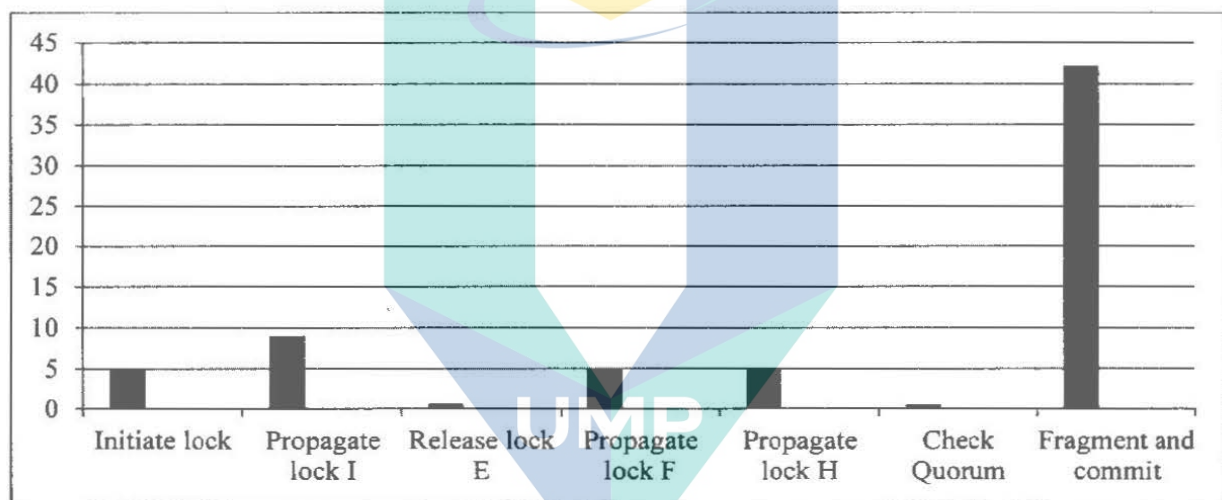


Figure 4.10 Execution time for Experiment 5

Figure 4.10 shows the execution time for Experiment 5. V_{η} took 4.941 ms to initiate lock. However, V_{η} requested to update data y which were not available in server E . Hence, server E would send the request to the server with data y which was server I . The time taken to propagate lock to server I was 8.964 ms. Once V_{η} locked server I , server E would release its lock and become available for another transaction. The execution time to propagate lock to server F and H were 4.878 ms and 4.898 ms, respectively. The time taken to check quorum was 0.473 ms. Finally, the time taken to fragment the database and commit the transaction was 42.152 ms. Thus, the total job execution time for this experiment was 66.306 ms.

Table 4.8 Experimental result for transaction request to update unavailable data

REPL ICA/ TIME	E	B	D	F	H
t1	unlock(<i>i</i>)	unlock(<i>i</i>)	unlock(<i>i</i>)	unlock(<i>i</i>)	unlock(<i>i</i>)
t2	begin_ transaction	begin_ transaction	begin_ transaction	begin_ transaction	begin_ transaction
t3	V_{η_i} write lock(<i>i</i>), counter_w(<i>e</i>)=1				
t4	V_{η_i} checks data map				
t5		V_{η_i} lock: I			
t6	V_{η_i} release lock	V_{η_i} write lock(<i>i</i>), counter_w(<i>i</i>)=1			
t7		V_{η_i} propagate lock:F			
t8				V_{η_i} lock(<i>i</i>) from I	
t9		V_{η_i} get lock:F, counter_w(<i>i</i>)=2			
t10		V_{η_i} propagate lock:H			
t11					V_{η_i} lock(<i>i</i>) from I
t12		V_{η_i} get lock:I, V_{η_i} counter_w (<i>i</i>)=3			
t13		V_{η_i} obtain quorum			
t14		V_{η_i} update <i>i</i>			
t15		<i>S</i> is fragmented into <i>S</i> ₁ and <i>S</i> ₂			
t16		<i>S</i> ₁ is fragmented into <i>S</i> _{1(<i>Pk,y</i>)} and <i>S</i> _{1(<i>Pk,x</i>)}			
t17		commit $\hat{V}_{\lambda_i} \in$ V_{η}	commit $\hat{V}_{\lambda_i} \in$ V_{η}	commit $\hat{V}_{\lambda_i} \in$ V_{η}	
t18		unlock(<i>i</i>)	unlock(<i>i</i>)	unlock(<i>i</i>)	unlock(<i>i</i>)

As shown in Table 4.8, at time equal to 1 (*t1*), instant *e* at all servers were unlocked. At *t2*, the transaction began.

At $t3$, there was a transaction, V_{η_i} requesting to update instant i at server E . Because transaction V_{η_i} successfully initiated lock, write counter for server E was now equal to 1.

However, at $t4$, because data i were not available at server E , it would check the data mapping system to find the server with the specific data. At $t5$, V_{η_i} locked server I because it was holding the data i . V_{η_i} released its lock at server E at $t6$ and now the write counter at Server I was equal to 1.

At $t7$, V_{η_i} propagated lock at its neighbour server F and at $t8$, V_{η_i} locked (i) from I . Thus, at $t9$, the transaction achieved in getting locked from server F which then resulted in write quorum to be equal to 2.

Next, V_{η_i} propagated lock at server H at $t10$ and at $t11$, V_{η_i} locked (i) from I . Thus, at $t12$, the transaction achieved in getting locked from server H which then resulted in write quorum to be equal to 3.

At $t13$, V_{η_i} obtained all quorums and then instant i was updated at $t14$. At $t15$, the relation S was fragmented into S_1 and S_2 using vertical fragmentation. At $t16$, the relation S_1 was fragmented again into $S_{1(Pk,y)}$ and $S_{1(Pk,x)}$ using horizontal fragmentation.

Finally, at $t17$, $\hat{V}_{\lambda_i} \in V_{\eta}$ was committed and at $t18$, instant i at all replica servers would be unlocked and ready for the next transaction to take place. A time diagram about the whole process for these five replication servers is presented in Figure 4.11.

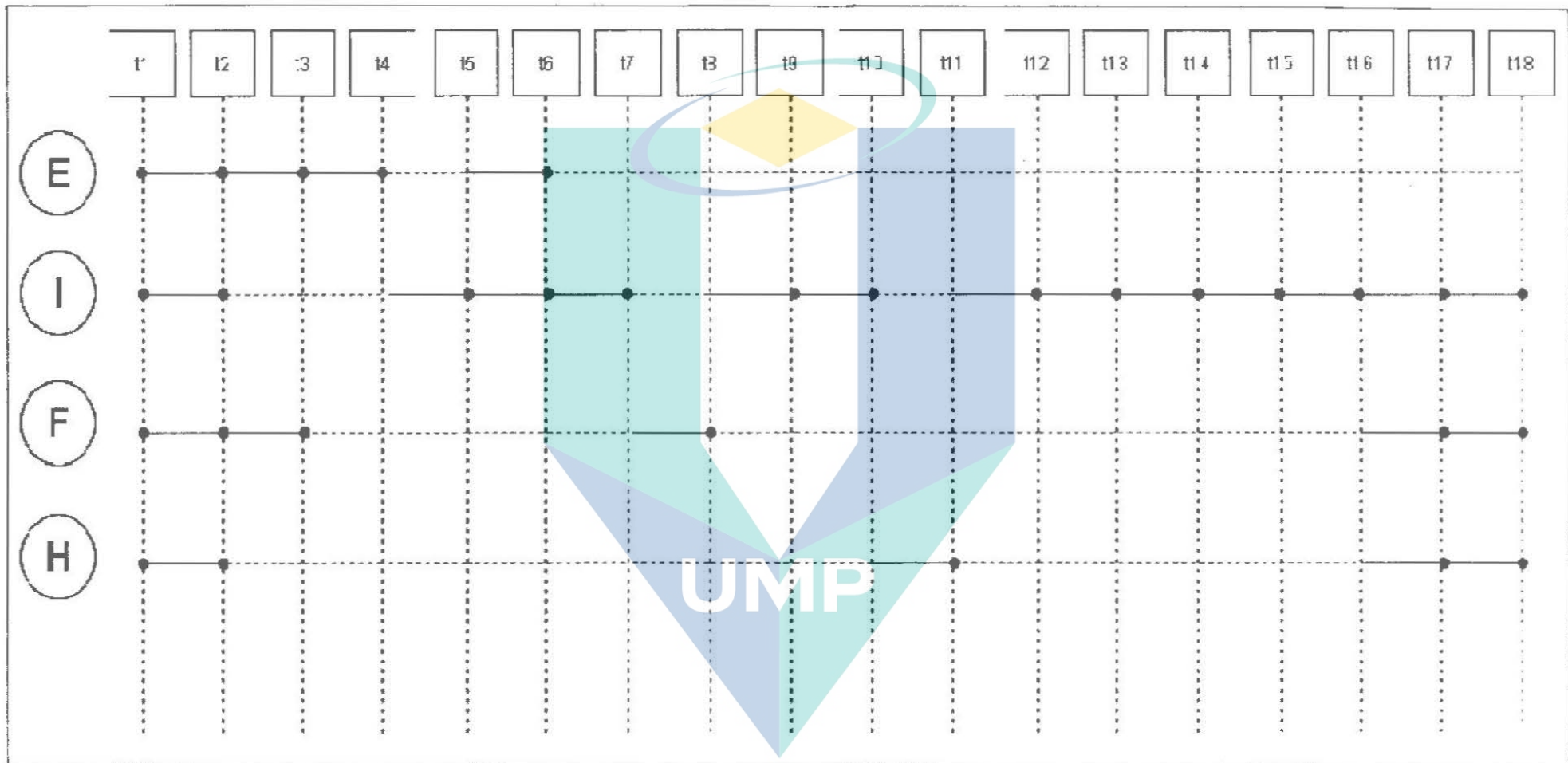


Figure 4.11 Time diagram when a set of transaction V_i request to update an unavailable instant, i at a site.

اونيورسيتي ملايسيا قهغ

UNIVERSITI MALAYSIA PAHANG

4.4 Results and Discussion

The proposed BVAGQ-AR has been compared with other replication techniques in terms of the total job execution time for a transaction and communication cost. In this section, the total job execution time to update data and communication cost between five existing techniques, namely, BSCA, PRA, HRS, BRS and ROWA, are compared with the proposed technique.

4.4.1 Validity Threats

Several validity threats can be associated with these experimental studies. A few threats have been identified and their effects on the results are elaborated.

First, the benchmark choice represents an essential threat. The experimental benchmarks from other studies in literature have been adopted. However, it cannot be guaranteed these benchmarks would represent the actual software and hardware configurations in real world. Nevertheless, the benchmarks are derived from configurations of different software programs.

Second, a comparison with other techniques is another threat. Other replication techniques with data mining such as BSCA and PRA are tested using simulation tools. The present research focused on testing the replication technique in real time DDS because simulation cannot capture the problems that arise in real time environment. Nevertheless, the comparison is valid because all the techniques compared have been tested using the same software and hardware in real time environment.

Finally the choice for the total number of replica servers can also pose as threat. In the present research, the total number of servers was equal to $\sqrt{n} \times \sqrt{n}$ to complete the grid structure. However, depending on each technique, the communication cost for the minimum and maximum numbers of replica servers has been identified for comparison with this proposed technique.

4.4.2 Comparison of Communication Costs

Table 4.9 shows the comparison of communication costs for BSCA, PRA, ROWA, HRS, BRS and BVAGQ-AR with the total number of sites, $n = 9, 25, 36, 49$ and 64 , respectively.

Table 4.9 Comparison of communication costs

Replication technique	Number of sites (n)				
	9	25	36	49	64
BSCA	$3 \leq d \leq 9$	$3 \leq d \leq 25$	$3 \leq d \leq 36$	$3 \leq d \leq 49$	$3 \leq d \leq 64$
PRA	3 to ∞	3 to ∞	3 to ∞	3 to ∞	3 to ∞
ROWA	9	25	36	49	64
HRS	9	25	36	49	64
BRS	8 to ∞	8 to ∞	8 to ∞	8 to ∞	8 to ∞
BVAGQ-AR	$3 \leq d \leq 5$	$3 \leq d \leq 5$	$3 \leq d \leq 5$	$3 \leq d \leq 5$	$3 \leq d \leq 5$

Table 4.9 shows that BSCA, PRA and BVAGQ-AR have the lowest communication cost, where d would be equal to the number of servers. The minimum number of copies of replicated data in BSCA, PRA and BVAGQ-AR would only be 3. However, the maximum number of copies of replicated data in BSCA and PRA would be high. This is because in BSCA, all data would be copied at least at one site while in PRA, the data would be copied at a server every time should a user request data which would be unavailable at that server. Hence, the replication data would be increased from time to time.

Unlike BSCA and PRA, even though the number of replicated servers in BVAGQ-AR were 9, 25, 36, 49 or 64, the replica copies of each data would only be replicated in the 5 neighbour servers. BRS has the second lowest communication cost where a minimum of 8 copies would be needed. However, there could exist as many replica copies as possible. This is because the replication server in BRS would be driven through client's request.

Table 4.10 Improvement shown by BVAGQ-AR in terms of communication cost (%)

REPLICA SERVERS	9	25	36	49	64
BSCA	44.44	80	86	89.8	92.19
PRA	0 to ∞	0 to ∞	0 to ∞	0 to ∞	0 to ∞
ROWA	44.44	80	86	89.8	92.19
HRS	44.44	80	86	89.8	92.19
BRS	33.33	12	8.33	6.12	4.69

Table 4.10 shows that, in 9 servers environment, BVAGQ-AR has improved 33.33% compared to BRS techniques while in 25, 36, 49, and 64 servers environment, BVAGQ-AR has improved 12%, 8.33%, 6.12%, and 4.69% compared to BRS techniques, respectively. PRA, ROWA and HRS would have the same communication costs because all of the techniques applied all-data-to-all-site replication scheme. In addition, in 9 servers environment, BVAGQ-AR has improved 44.44% compared to ROWA and HRS techniques while in 25, 36, 49, and 64 servers environment, BVAGQ-AR has improved 80%, 86%, 89.8%, and 92.19% compared to ROWA and HRS techniques. In conclusion, BVAGQ-AR has the lowest communication cost compared to those of BSCA, PRA, ROWA, HRS and BRS.

4.4.3 Replication Job Execution Time Comparison

Two series of experiments were executed in order to identify the job execution time for each technique. The first experiment was executed using the minimum number of replication servers for each replication technique. Table 4.11 shows the comparison of job execution time for the first experiment.

Table 4.11 Comparison of job execution time for the minimum number of replication servers

Replication Techniques	Min. number of servers	Initiate Lock (ms)	Propagate Lock (ms)	Obtain Majority Quorum (ms)	Database Fragmentation & Commit (ms)	Total time taken:
BSCA	3	4.044	48.481	3.864	39.743	88.404
PRA	3	4.136	46.998	3.882	41.695	96.711
ROWA	9	4.275	144.522	8.187	105.259	262.243
HRS	9	3.956	147.227	7.870	98.875	257.928
BRS	8	4.523	64.268	8.112	60.254	137.157
BVAGQ-AR	3	3.905	16.369	3.890	42.384	66.548

Table 4.11 shows the execution time comparison between BSCA, PRA, ROWA, HRS, BRS and BVAGQ-AR in their minimum replication servers. The results, as shown in Table 4.12, have provided evidence that BVAGQ-AR would require the shortest duration of time to complete a transaction. A complete transaction took only 66.548 ms to finish. The second and the third in the order of the duration of execution time are BCSA with 88.404 ms and PRA with 96.711 ms, respectively. PRA would take longer time due to user prefetching data from other servers. Next is BRS, which would take 137.157 ms to complete the replication process. ROWA and HRS would take the longest of execution time with more than 250 ms. As shown in Table 4.12, there are huge differences in the total job execution time between BSCA and PRA with those of ROWA, BRS and HRS. This is because the data in ROWA, BRS and HRS were not mined because the original techniques did not consider the data correlation.

The second experiment was executed using the maximum number of replication servers for each technique. Table 4.12 shows the comparison of job execution time for the second experiment.

Table 4.12 Comparison of job execution time for the maximum number of replication servers

Replication Techniques	Min. number of servers	Initiate Lock (ms)	Propagate Lock (ms)	Obtain Majority Quorum (ms)	Database Fragmentation & Commit (ms)	Total time taken:
BSCA	9	4.097	75.272	8.433	105.172	192.974
PRA	9	3.974	81.250	9.214	97.170	191.608
ROWA	9	4.275	147.498	8.002	107.912	267.687
HRS	9	4.152	146.136	9.107	107.536	266.931
BRS	9	4.480	64.864	8.835	93.993	172.172
BVAGQ-AR	5	4.280	23.808	3.950	51.830	83.868

Table 4.12 shows the comparison of execution time between BSCA, PRA, ROWA, HRS, BRS and BVAGQ-AR for maximum replication servers. The results, as shown in Table 4.13, have provided evidence that BVAGQ-AR would require the shortest duration of time to complete a transaction as the number of maximum replication servers in this technique was only five. BVAGQ-AR took only 83.868 ms to complete a transaction. The second lowest execution time is PRA with 191.608 ms. This is followed by BSCA, which took 192.974 ms. Next is BRS, which took 185.172 ms to complete the replication process. ROWA and HRS each took the longest execution time which was more than 250 ms. Compared to other techniques, BRS would need less time to complete a transaction because the data in this technique were fragmented and allocated at several different sites while other techniques replicated all data to all sites.

Table 4.13 Improvement shown by BVAGQ-AR in terms of job execution time (%)

REPLICA SERVERS	BSCA	PRA	ROWA	HRS	BRS
Minimum	31.19	24.72	74.62	74.20	51.48
Maximum	56.54	56.23	68.67	68.58	51.23

As shown in Table 4.13, BVAGQ-AR has improved 31.19% compared to BCSA when experiment was executed using minimum number of replication servers and has shown 56.54% of improvement using maximum number of replication servers. This is followed by PRA where BVAGQ-AR has improved 24.72% using minimum number of replication servers and has shown 56.23% of improvement using maximum number of replication servers. The improvement in BSCA and that of PRA have made a huge difference because in BVAGQ-AR, the minimum and maximum numbers of replication servers would be 3 and 5, respectively, while in BSCA and PRA, they would be 3 and 9, respectively. BVAGQ-AR has improved 74.62% compared to ROWA and 74.20% compared to HRS using minimum number of servers, while showing 68.67% and 68.58% of improvement using maximum number of replication servers, respectively. There has not been much difference in the results because ROWA and HRS used 9 replication servers in both experiments. Finally, BVAGQ-AR has improved 51.48% compared to BRS using minimum number of replication servers and has shown 51.23% of improvement using maximum number of replication servers. The percentages are much higher in ROWA, HRS and BRS compared to those of BSCA, PRA and BVAGQ-AR because ROWA, HRS and BRS did not take into consideration the correlations between the data. For BSCA, PRA and BVAGQ-AR, the data mining process has been executed before proceeding with the data replication process. Hence, the processing time for each of them would be shorter. In conclusion, BVAGQ-AR has the lowest job execution time to complete a transaction compared to those of BSCA, PRA, ROWA, HRS and BRS.

4.5 Summary

This chapter presents detailed descriptions and the summary of the results from the prototype of the BVAGQ-AR technique. A detailed performance evaluation has been shown as well. Results have clearly shown that managing replication and transaction in fragmented database through proposed BVAGQ-AR would be able to preserve data consistency and availability with the lowest communication cost and the shortest duration of time for job execution compared to those of BSCA, PRA, ROWA, HRS and BRS. However, BVAGQ-AR technique has a few limitations. Hence, much improvement would still need to be made. This will be discussed in the next chapter.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Introduction

This work has proposed a replication technique called Binary Vote Assignment Grid Quorum with Association Rule (BVAGQ-AR) to produce data consistency with low communication cost and short processing time in managing replication for fragmented distributed database. In order to achieve the purpose of this research, a data mining technique has been integrated with the replication process. This chapter summarizes the key findings of the present research. It also includes some suggestions for future work in each of the areas covered during this research.

5.2 Conclusion

The aims of the present research was to design, implement and evaluate. From the analysis in Chapter 2, it has been evident that there is much room for improvement in the existing techniques. The objectives of this research were as follows:

- i. To design and develop data replication algorithms in distributed database environment with low communication cost and processing time for a transaction.
- ii. To enhance the data consistency technique in objective 1 for synchronous replication.

In order to achieve the first objective of this research, an algorithm called Binary Vote Assignment Grid Quorum with Association Rule (BVAGQ-AR) was proposed as described in Chapter 3. BVAGQ-AR would only consider the neighbours to obtain a data copy. Hence, the number of replication servers would be limited. The neighbours would be assigned with vote one and zero. A copy would be allocated to a site if and only if the vote assignment to the site would be equal to one.

In BVAGQ-AR, not only data replication would be considered but the present research was also concerned about the correlations between the data. Hence, the granularity for this proposed algorithm would be higher than the existing algorithm and the data would be replicated at the suitable servers. In addition, the existing replication techniques would tend to abort the transaction that failed to get a majority of the quorum in the first place. While in this proposed algorithm, it would place the second transaction in the queue and it could continue to precede with its transaction once it succeeded in obtaining a majority quorum.

These second and third objectives have been achieved and presented in Chapter 4. A series of experiments were conducted in order to prove this technique could preserve data consistency. After that, an analysis of BVAGQ-AR has been presented in terms of communication cost. It shows that BVAGQ-AR technique has provided a convenient approach to achieve data consistency for distributed database replication in real time environment by allowing only one transaction to be executed at one time. When a transaction committed its process, it would update the data in all its replication servers at the same time. Therefore, at the end of the process, all servers would have the same data. After comparing BVAGQ-AR with Pre-Fetching Based Dynamic Replication Algorithm (PRA), Based on Support and Confidence Dynamic Replication Algorithm (BSCA), Read-One-Write-All (ROWA), Hierarchy Replication Scheme (HRS) and Branch Replication Scheme (BRS), it has been shown that BVAGQ-AR is one of the techniques that would require the lowest communication cost for an operation.

Handling database update operations with less computational time becomes crucial for synchronous replication in real time. BVAGQ-AR has resolved this by setting the lock with small quorum size before updating and committing transaction synchronously to the sites having the same fragmented data. Because this technique has used small size of

quorum, less computational time would be needed to send and receive messages from the neighbours' replicas. In addition, maintaining data consistency would also be easier compared to other techniques because BVAGQ-AR has low communication cost. This is because less computational time would be required for the locking of the small quorum size in synchronization process.

5.3 Contributions to Knowledge

This research has provided novel contributions to knowledge in the related areas. Firstly, the present research has provided evidence that the data replication technique called Binary Vote Assignment Grid Quorum with Association Rule (BVAGQ-AR) could be employed with low communication cost and short replication processing time. BVAGQ-AR would apply some data to some sites mechanism. The data would only be replicated to the neighbour servers. Hence, BVAGQ-AR would use a maximum number of only five replication servers. In addition, the data would also be fragmented before being allocated at the replication servers. This technique also has been proven to have the shortest duration of job processing time compared to those of ROWA, HRS and BRS because the data in BVAGQ-AR has been mined to find the frequent itemsets.

Secondly, the combination of BVAGQ-AR replication technique and the transaction management has formed a reliable system. This technique would be able to manage a synchronous database replication in real time environment.

Finally, the BVAGQ-AR Transaction Manager (BTM) would solve the problem of distributed concurrency transactions in databases. BTM would allow only one transaction to be executed at one time. When a transaction commits its process, it would update the data in all its replication servers at the same time. Thus, at the end of the process, all servers would have the same data.

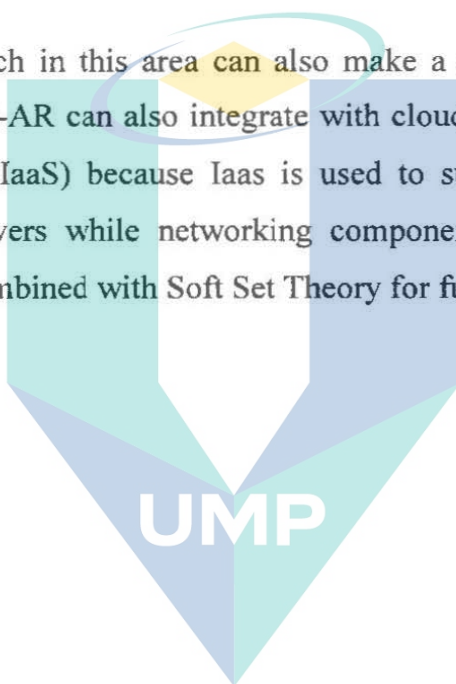
5.4 Future Work

BVAGQ-AR can be improved in many different ways. Currently, BVAGQ-AR does not support handling fragmented database replication transaction management by considering failure cases. In the future, it is anticipated that BVAGQ-AR would be able

take this challenge to handle fragmented database failure case and fault tolerance such as system crashes, statement failure, application software errors, network failure and media failure in real time distributed database system in real time environment.

The prototype tool has few fields to update data into the different replication servers. To make this more user-friendly, various forms of module can be introduced. In the present research, only two modules were developed to test the algorithm. Therefore, many modules can be added in the future to improve this prototype. It also can be tested in Big Data environment.

In the future, research in this area can also make a significant improvement for commercial usage. BVAGQ-AR can also integrate with cloud computing service focus in Infrastructure-as-a-Service (IaaS) because IaaS is used to support operations, including storage, hardware, and servers while networking components can also be introduced. BVAGQ-AR can also be combined with Soft Set Theory for further analysis of the data.



اونيورسيتي مليسيا قهغ

UNIVERSITI MALAYSIA PAHANG

REFERENCES

- Al-Ekram, R. and Holt, R. 2010. OSSR: Optimal Single Site Replication. *Parallel and Distributed Processing with Applications (ISPA)*. 433 – 441.
- Al-Mistarihi, H.E., and Yong, C. 2008. Replica Management in Data Grid. *International Journal of Computer Science and Network Security*. 8: 22 – 32.
- Al-Mistarihi, H.H.E., and Yong, C.H. 2009. On Fairness, Optimizing Replica Selection in Data Grids. *IEEE Transactions on Parallel and Distributed Systems*. 20 (8): 1102 – 1111.
- Amir, L. A., Helder, M. R., Nooruldeen, A. D., Qaderd, N. 2018. A Data Replication Algorithm for Groups of Files in Data Grids. *Journal of Parallel and Distributed Computing*. 113: 115-126.
- Amir, S. S. and Rahmaniab, M. 2018. Systematic Survey of Big Data and Data Mining in Internet of Things. *Computer Networks*. 139: 19-47
- Amjad, T., Sher, M., Daud, A. 2012. A Survey of Dynamic Replication Strategies for Improving Data Availability in Data Grids. *Future Generation Computer System*. 28 (2): 337 – 349.
- Amza, C., Cox, A., and Zwaenepoel, W. 2003. Conflict-Aware Scheduling for Dynamic Content Applications. *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*.
- Aouiche, K. Darmont, J. Boussaïd, O. Bentayeb, F. 2005. Automatic Selection of Bitmap Join Indexes in Data Warehouses. *Data Warehousing and Knowledge Discovery*. 64 – 73.
- Apers P. M. G. 1988. Data Allocation in Distributed Database Systems. *ACM Transaction on Database System*, 13: 263 – 304.
- Asghari, S. and Navimipour, N. J. 2016. Review and Comparison of Meta-Heuristic Algorithms for Service Composition in Cloud Computing. *Majlesi Journal of Multimedia Processing*. 4(4): 28 – 34.
- Ashouraie, M. and Navimipour, N. J. 2015. Priority-Based Task Scheduling on Heterogeneous Resources in the Expert Cloud. *Kybernetes*. 44(10) 1455–1471.
- Baiao, F., Mattoso, M., Zaverucha, G. 2000. Horizontal Fragmentation in Object DBMS: New issues and Performance Evaluation. *Proceeding of IEEE International Conference on Performance, Computing, and Communications*. 108 – 114.
- Barr, M. and Bellatreche, L. 2010. A New Approach Based on Ants for Solving the Problem of Horizontal Fragmentation in Relational Data Warehouses. *Proceeding of International Conference on Machine and Web Intelligence*. 411 – 415.

- Beigrezaei, M., Haghghat, A. T., Meybodi M. R., Runiassy, M. 2016. PPRA: A New Pre-Fetching and Prediction Based Replication Algorithm in Data Grid. *Proceeding of International Conference on Computer and Knowledge Engineering*. 257 – 262.
- Berman, F., Hey, A. J. G. and Fox, G. C. 2003. Grid Computing: Making the Global Infrastructure a Reality. *Wiley Series in Communications Networking & Distributed Systems*.
- Bernstein, P., Hadzilacos, V., and Goodman, N. 1987. Concurrency Control and Recovery in Database Systems. *Addison-Wesley*.
- Bernstein, P., and Goodman, N. 1983. The Failure and Recovery Problem for Replicated Databases. *Proceedings of the 2nd Annual ACM symposium on Principles of Distributed Computing*, 114 – 122.
- Bhar, S. and Barker, K. 1995. Static Allocation in Distributed Object Base Systems: A Graphical Approach. *Proceeding of International Conference on Information Systems Data Management*. 92 – 114.
- Birman, K. P. 1985. Replication and Fault-Tolerance in The ISIS System. *Proceedings of the 10th ACM symposium on Operating Systems Principles*. 79 – 86.
- Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., Zomaya, A. 2013. Energy-Efficient Data Replication in Cloud Computing Datacenters. *Proceeding of IEEE Globecom Workshops*. 446 – 451.
- Bsoul, M., Al-Khasawneh, A., Eddien Abdallah, E., and Kilani, Y. 2011. Enhanced Fast Spread Replication strategy for Data Grid. *Journal of Network Computer Application*. 34 (2): 575 – 580.
- Budiarto, S.N., Tsukamoto, M. 2002. Data Management Issue in Mobile and Peer to Peer Environment. *Data Knowledge Engineering*. 41: 183 – 204.
- Buyya, R. and Venugopal. 2005. A Gentle Introduction to Grid Computing and Technologies. *Computer Society of India*. 29(1): 9 – 19.
- Carballeira, F.G., Carretero, J., Calderon, A., Garcia, J.D., and Sanchez, L.M. 2007. A Global and Parallel File System for Grids. *Future Generation Computer Systems*. 116 – 122.
- Ceri, S. and Pelagatti, G. 1984. Distributed Databases: Principles and Systems. *New York: McGraw-Hill*.
- Charrada, B. F., Ounelli, H. and Chettaoui, H. 2010. An Efficient Replication Strategy for Dynamic Data Grids. *Proceedings of IEEE International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. 50 – 54.
- Charrada, B. F., Ounelli, H. and Chettaoui, H. 2010. Dynamic Period vs Static Period in Data Grid Replication. *Proceeding of IEEE International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. 565 – 568.

- Chiregi, M. and Navimipour, N. J. 2016. A New Method for Trust and Reputation Evaluation in the Cloud Environments Using the Recommendations of Opinion Leaders' Entities and Removing The Effect Of Troll Entities. *Computers in Human Behavior*. 60: 280–292.
- Connolly, T. and Begg, C. 2015. Database System a Practical Approach to Design, Implementation and Management, *International Edition, Fifth Edition, Pearson Education*.
- Cui, Z., Zuo, D., Zhang, Z. 2013. Based on Support and Confidence Dynamic Replication Algorithm in Multi-Tier Data Grid. *Proceeding of International Journal of Computational Information Systems*. 10: 3909 – 3918.
- Daudjee, K., and Salem, K. 2004. Lazy Database Replication with Ordering Guarantees. *Proceedings of Data Engineering*. 424 – 435.
- Davidson, B., Molina, H. G. and Skeen, D. 1985. Consistency in a Partitioned Network: A Survey. *ACM Computing Surveys*. 17 (3): 341 – 370.
- Deris, M.M., Abawajy, J.H., Taniar, D., Mamat, A. 2009. Managing Data using Neighbour Replication on a Triangular-Grid Structure. *International Journal of High Performance Computing and Networking*. 6 (1): 56 – 65.
- Deris, M. M., Evans, D. J., Saman, M. Y., Noraziah A. 2003. Binary Vote Assignment on Grid for Efficient Access of Replicated Data. *International Journal of Computer Mathematics*, 80 (12): 1489 – 1498.
- Deris, M.M., Mamat, R., Noraziah, A., Suzuri, H.M. 2003. High Service Reliability for Cluster Server Systems. *Proceeding of IEEE International Conference on International Conference on Cluster Computing*. 280–287.
- Doraimani, S. 2007. Filecules: A New Granularity for Resource Management in Grids. *Master thesis. University of South Florida, USA*.
- Dullmann, D., Hoschek, W., Jaen-Martinez, J. and Segal, B. 2001. Models for Replica Synchronization and Consistency in a Data Grid. *Proceeding of IEEE Symposium on High Performance on Distributed Computing*. 67 – 75.
- Foster, C. K. and Tuecke. S. 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*. 15 (3): 200 – 222.
- Foster, I. and Kesselman, C. 1999. The Grid: Blueprint for a New Computing Infrastructure. *Morgan Kaufmann*.
- Gao, L., Dahlin, M., Zheng, J., Alvisi, L. and Iyengar, A. 2010. Dual-Quorum: A Highly Available and Consistent Replication System for Edge Services Dependable and Secure Computing. *Proceeding of IEEE Transactions on Dependable and Secure Computing*. 7 (2): 159 – 174.

- Gao, M. D., Nayate, A., Zheng, J. and Iyengar, A. 2005. Improving Availability and Performance with Application-Specific Data Replication. *Proceeding of IEEE Transaction on Knowledge and Data Engineering*. 17 (1):106 – 200.
- Golding, R. A. 1992. Weak-consistency group communication and membership. *Ph.D. thesis, Computer and Information Sciences Board, University of California, Santa Cruz*.
- Grace, R.K. and Manimegalai, R. 2014a. Data Access Prediction and Optimization in Data Grid Using SVM and AHL Classifications. *International Review Computers and Software*. 9(7):1188 – 1194.
- Grace, R.K. and Manimegalai, R. 2014b. Dynamic Replica Placement and Selection Strategies in Data Grids A Comprehensive Survey. *Journal of Parallel and Distributed Computing*. 74(2): 2099 – 2108.
- Gray, J., Helland, P., O’Neil, P. E., and Shasha, D. 1996. The Dangers of Replication and A Solution. *Proceedings of ACM SIGMOD International Conference on Management of Data*. 173 – 182.
- Gullo, F. 2015. From Patterns in Data to Knowledge Discovery: What Data Mining Can Do. *Physics Procedia*. 62: 18 – 22 .
- Hamrouni, T., Slimani, S., Charrada, F. B. 2015a. A Critical Survey of Data Grid Replication Strategies Based on Data Mining Techniques. *Proceeding of Procedia Computer Science*. 51: 2779 – 2788.
- Hamrouni, T., Slimani, S., Charrada, F. B. 2015b. A Survey Of Dynamic Replication and Replica Selection Strategies Based on Data Mining Techniques in Data Grids. *Proceeding of Engineering Applications of Artificial Intelligence*. 48: 140 – 158.
- Han, J., Cheng, H., Xin, D., Yan, X. 2007. Frequent Pattern Mining: Current Status and Future Directions. *Data Mining and Knowledge Discovery*. 15: 55 – 86.
- Han, J., Kamber, M., Pei, J. 2011. Data Mining: Concepts and Techniques. *Morgan Kaufmann Publishers*.
- Han, J., Kamber, M., Pei, J. 2012. Data Mining: Concepts and Techniques. Reprint. *Morgan Kaufmann Publishers*
- Harandi, M. T., Hou, J., Gupta, I. 2011. Transaction with Replication. *Nikita Borisov, University of Illinois*.
- Hoschek, W., Jaen-martinez, J., Samar, Stockinger, A., Stockinger, H. 2000. Data Management in An International Data Grid Project. *Proceeding of ACM International Workshop on Grid Computing*. 77 – 90.
- Hosseini, R., Parand, F., Riahi, D. 2016. Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases. *Applied Computing and Informatics*.

- IBM. 1999. DB2: Replication Guide and Reference. *New Orchard Road, Armonk, NY 10504 (USA). Number SC26-9642-00.*
- Jemal, H. A. and Deris, M. M. 2014. Data Replication Approach with Consistency Guarantee for Data Grid. *IEEE Transactions on Computers.* 63: 2975 – 2987.
- Jin, H., Cortes, T., and Buyya, R. 2002. High Performance Mass Storage and Parallel I/O: Technologies and Applications. *IEEE Press/Wiley.*
- Joseph, J. and Fellenstein, C. 2004. Grid Computing. *Pearson Education.*
- Kemme, B. and Alonso, G. 1998. A Suite of Database Replication Protocols Based on Group Communication Primitives. *Proceeding of International Conference on Distributed Computing Systems.* 156 – 163.
- Ko, S.Y., Morales, R., Gupta, I. 2007. New Worker-Centric Scheduling Strategies for Data-Intensive Grid Applications. *Proceedings of the International Conference on Middleware.* 121 – 142.
- Krishnamurthy, S., Sanders, W. H. and Cukier, M. 2003. An Adaptive Quality of Service Aware Middleware for Replicated Services. *Proceedings of IEEE Transactions on Parallel and Distributed Systems.* 14 (11): 1112 – 1125.
- Ladin, R., Liskov, B., Shrira, L. and Ghemawat, S. 1992. Providing High Availability Using Lazy Replication. *Proceedings of ACM Transactions on Computer Systems.* 10 (4): 360 – 391.
- Laura C. V., Schuldt, H., Breitbart, Y. and Schek, H. J. 2009. Replicated Data Management in The Grid: The Re:GRiDiT Approach. *Proceedings of ACM Workshop on Data grids for eScience.* 7 – 16.
- Linesch, M. 2007. HP.: Grid – Distributed Computing at Scale an Overview of Grid and the Open Grid Forum. *Copyright © Open Grid Forum 2006, 2007. All Right Reserved. GFD-112.*
- Ma, H., Schewe, K.D and Kirchberg, M. 2006. A Heuristic Approach to Vertical Fragmentation Incorporating Query Information Databases and Information Systems. *Proceedings of International Baltic Conference on Databases and Information Systems.* 69 – 76.
- Ma, J., Liu, W. and Glatard, T. 2013. A Classification of File Placement and Replication Methods On Grids. *Future Generation Computer System.* 29(6): 1395 – 1406.
- Milani, B. A. and Navimipour N. J. 2016. A Comprehensive Review of the Data Replication Techniques in the Cloud Environments: Major Trends And Future Directions. *Journal of Network and Computer Applications.* 64: 229 – 238.
- Milani, A. S. and Navimipour, N. J. 2016. Load Balancing Mechanisms and Techniques in The Cloud Environments: Systematic Literature Review and Future Trends. *Journal of Network and Computer Applications.* 71: 86 – 98

- Milani, B. A. and Navimipour, N. J. 2017. A Systematic Literature Review of the Data Replication Techniques in the Cloud Environments. *Big Data Research*. 10: 1-7
- Mistarihi, H.E. and Yong, C. 2008 Replica Management in Data Grid. *International Journal Computer Science Network Security*. 8(6): 22 – 32.
- Mokadem, R. and Hameurlain, A. 2015. Data Replication Strategies with Performance Objective in Data Grid Systems: A Survey. *International Journal of Grid and Utility Computing*. 6(1): 30 – 46.
- Moore, R., Baru, C., Marciano, R., Rajasekar, A., Wan, M. 1999. Data-Intensive Computing in the Grid: Blueprint for a New Computing Infrastructure. 105 – 29.
- Mulk., V. N. and Mohamed, A. M. 2017. A Prediction Based Dynamic Replication Strategy for Data Intensive Applications. *Computers & Electrical Engineering*. 57: 281-293
- Naseera, S., and Murthy, K.V.M. 2009. Agent Based Replica Placement in a Data Grid Environment. *Proceedings of International Conference on Computational Intelligence, Communication Systems and Network*. 426 – 430.
- Navathe, S. B. and Ra, M. 1989. Vertical Partitioning for Database Design: A Graphical Algorithm. *SIGMOD Record*. 14: 440 – 450.
- Navathe, S., Ceri, S., Wiederhold, G. and Dou, J. 1984. Vertical Partitioning Algorithms for Database Design. *Proceedings of ACM Transaction Database System*. 9: 680 – 710.
- Navimipour, N. J. and Milani, F. S. 2014. A Comprehensive Study of the Resource Discovery Techniques in Peer-To-Peer Networks. *Peer-To-Peer Networking and Applications*. 8(3): 474–492
- Navimipour, N. J., Navin, A. H., Rahmani, A. M., Hosseinzadeh M. 2015. Behavioral Modeling and Automated Verification of a Cloud-Based Framework to Share The Knowledge and Skills Of Human Resources. *Computers in Industry*. 68: 65-77
- Nicola, M. and Jarke, M. 2000. Performance Modeling Of Distributed and Replicated Databases. *Proceedings of IEEE Transactions on Knowledge Data Engineering*. 12(4): 645 – 672.
- Noraziah, A., Deris, M.M., Ahmed, N.A., Norhayati, R., Saman, M.Y., Zeyed, M.Y. 2007. Preserving Data Consistency through Neighbour Replication on Grid Daemon. *American Journal of Applied Science*. 4(10): 748 – 755.
- Noraziah, A., Abdalla, A. N. and Roslina M.S. 2010. Data Replication Using Read-One-Write-All Monitoring Synchronization Transaction Systems in Distributed Environment. *Journal of Computer Science*. 6 (10): 1033 – 1036.
- Noraziah, A., Ainul Azila, C.F, Roslina, M.S., Noriyani, M.Z. and Beg, A.H. 2010. Lowest Data Replication Storage of Binary Vote Assignment Data Grid. *Proceedings of International Conference on Networked Digital Technologies*. 466 – 473.

- Noraziah, A., Deris, M. M., Norhayati, R., Rabiei, M. and Shuhadah, W.N.W. 2008. Managing Transaction on Grid-Neighbour Replication in Distributed System. *International Journal of Computer Mathematics*. 86 (9): 1 – 10.
- Olston, C. and Widom, J. 2000. Offering a Precision-Performance Tradeoff for Aggregation Queries over Replicated Data. *Proceedings of the International Conference on Very Large Data Bases*. 1 – 12.
- Oracle Corporation. 1998. 500, Oracl8i Advanced Replication, *Oracle Technical White Paper*.
- Pedone, F., Wiesmann, M., Schiper, A., Kemme, B. and Alonso, G. 2000. Understanding Replication in Databases and Distributed Systems. *Proceedings of the International Conference on Distributed Computing Systems*. 464 – 474.
- Pérez, J. M., Carballeira, F. G., Carretero, J., Calderón, A., and Fernández, J. 2010. Branch Replication Scheme: A New Model for Data Replication in Large Scale Data Grids. *Future Generation Computer Systems*. 26: 12 – 20.
- Plattner, C. and Alonso, G. 2004. Ganymed: Scalable Replication for Transactional Web Applications. *Proceedings of the International Conference on Middleware*. 155 – 174.
- Ram, S. and Narasimhan, S. 1994. Database Allocation in A Distributed Environment: Incorporating A Concurrency Control Mechanism and Queuing Costs. *Management Science*. 40: 969 – 983.
- Ram, S. and Narasimhan, S. 1995. Incorporating The Majority Consensus Concurrency Control Mechanism into the Database Allocation Problem. *ORSA Journal of Computing*. 7: 244 – 256.
- Ranganathan, K. and Foster, I. 2001. Identifying Dynamic Replication Strategies for A High-Performance Data Grid. *Proceedings of the International Grid Computing Workshop, Springer*. 2242: 75 – 86.
- Ranganathan, K. and Foster, I. 2001. Identifying Dynamic Replication Strategies for a High-Performance Data Grid. *Proceedings of Second International Workshop on Grid Computing, Springer-Verlag*. 75 – 86.
- Ren, K., Li, Z., and Wang C. 2010. LBDRP: A Low-bandwidth Data Replication Protocol on Journal-based Application. *Computer Engineering and Technology*. 89 – 92.
- Ritu Garg, R. and Singh, A. K. 2015. Adaptive Workflow Scheduling in Grid Computing Based on Dynamic Resource Availability. *Engineering Science and Technology*. 18(2): 256 – 269.
- Sacca, D. and Wiederhold, G. 1985. Database partitioning in a cluster of processors. *Transaction Database System*. 10: 29 – 56.

- Sánchez, A., Montes, J., Dubitzky, W., Valdés, J.J., Pérez, M.S., Miguel, P.D. (2008). Data Mining Meets Grid Computing: Time To Dance. In: Data Mining Techniques in Grid Computing Environments. *John Wiley & Sons*. 1 – 16.
- Sashi, K., and Thanamani, A. S. 2010. Dynamic Replica Management for Data Grid. *International Journal of Engineering and Technology*. 2 (4).
- Sashi, K., and Thanamani, A.S. 2010. A New Replica Creation and Placement Algorithm for Data Grid Environment. *Data Storage and Data Engineering*. 265 – 269.
- Sathya, S. S., Kuppaswami, S. and Ragupathi, R. 2006. Replication Strategies for Data Grids. *Proceedings of International Conference on Advanced Computing and Communications*. 123 – 128.
- Satio, Y. and Shapiro, M. 2005. Optimistic Replication. *Proceedings of ACM Computing Survey*. 37 (1): 1 – 44.
- Schneider, F. B. 1993. Replication Management Using The State-Machine Approach. *Distributed systems (2nd Edition)*, ACM Press/Addison-Wesley Publishing.
- Shorfuzzaman, M., Graham, P., and Eskicioglu, R. 2010. Distributed Popularity Based Replica Placement in Data Grid Environments. *Proceedings of International Conference on Parallel and Distributed Computing, Applications and Technologies*. 66 – 77.
- Shorfuzzaman, M., Graham, P., and Eskicioglu, R. 2011. QoS-Aware Distributed Replica Placement in Hierarchical Data Grids. *Proceedings of IEEE International Conference on Advanced Information Networking and Applications*. 291 – 299.
- Stockinger, H. 2001. Distributed Database Management Systems and the Data Grid. *Proceeding of IEEE Symposium on Mass Storage Systems and Technologies*. 1 – 1.
- Souri A. and Navimipour, N. J. 2014. Behavioral Modeling and Formal Verification of a Resource Discovery Approach in Grid Computing. *Expert Systems with Applications*. 41(8): 3831–3849.
- Stockinger, H., Samar, A., Allcock, B., Foster, I., Holtman, K., and Tierney, B. 2001. File and Object Replication in Data Grids High Performance Distributed Computing. *Proceedings of IEEE International Symposium*. 76 – 86.
- Tackett, J.J., Gunter, D., and Brown, L. 1995. Special Edition Using Linux. *Que Corporation USA*.
- Tamhankar, A.M., Ram, S. 1998. Database Fragmentation and Allocation: An Integrated Methodology and Case Study. *Proceeding of IEEE Transactions Systems, Man and Cybernetics, Part A: Systems and Humans*, 28 (3): 288 – 305.
- Tang, M., Lee, B.S., Tang, X. and Yeo, C.K. 2006. The Impact on Data Replication on Job Scheduling Performance in the Data Grid. *International Journal of Future Generation of Computer Systems*. 22: 254 – 268.

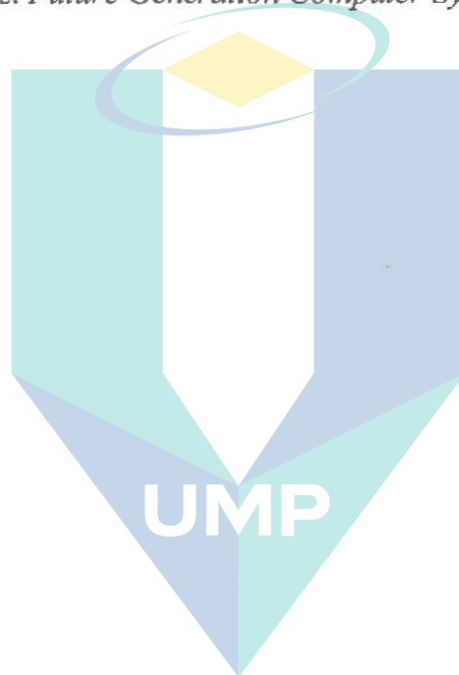
- Terry, D., Demers, A., Petersen, K., Spreitzer, M., Theimer, M. and B. Welsh. 1994. Session Guarantees for Weakly Consistent Replicated Data. *Proceedings of 3rd International Conference on Parallel and Distributed Information Systems*. 140 – 149.
- Thomas, R. H. 1979. A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases. *Proceedings of ACM Transactions on Database Systems*. 4 (2): 180 – 209.
- Tian, T., Luo, J., Wu, Z., Song, A. 2008. A Pre-Fetching-Based Replication Algorithm in Data Grid. *Proceedings of the 3rd International Conference on Pervasive Computing and Applications*. 526 – 531.
- Tu, M. 2006. A Data Management Framework for Secure and Dependable Data Grid. *Ph.D. dissertation. University of Texas, USA*.
- Tu, M., Li, P., I-Ling Yen, Thuraisingham, B., and Khan, L. 2010. Secure Data Objects Replication in Data Grid. *Proceedings of IEEE Transactions on Dependable and Secure Computing*. 7 (1): 50 – 64.
- Vadim, K. 2018. Overview of Different Approaches to Solving Problems of Data Mining. *Procedia Computer Science*. 123: 234-239
- Vazhkudai, S., Schopf, J. 2003. Using Regression Techniques to Predict Large Data Transfers. *International Journal on High Performance Computer Application*. 17(3): 249 – 268.
- Wang, J., Wu, H., Wang, R. 2017. A New Reliability Model in Replication-Based Big Data Storage Systems. *Journal of Parallel and Distributed Computing*. 108: 14 – 27.
- Wiesmann, M., Pedone, F., Schiper, A., Kemme, B., and Alonso, G. 2000. Database Replication Techniques: A Three Parameter Classification. *Proceedings of IEEE Symposium on Reliable Distributed Systems*. 206 – 215.
- Yin, J., Alvisi, L., Dahlin, M. and Lin, C. 1999. Volume Leases for Consistency in Large-Scale Systems. *Proceedings of IEEE Transaction on Knowledge and Data Engineering*. 11 (4): 536 – 576.
- Yu, H. and Vahdat, A. 2002. Design and Evaluation of a Conit-Based Continuous Consistency Model for Replicated Services. *Proceedings of ACM Transactions on Computer Systems*. 20 (3): 239 – 282.
- Yuan, Y., Wu, Y., Yang, G. and Yu, F. 2007. Dynamic Data Replication based on Local Optimization Principle in Data Grid and Cooperative Computing. *Proceedings of International Conference on Grid and Cooperative Computing*. 815 – 822.
- Zaki, M.J. and Meira, W. Jr. 2014. Data Mining and Analysis: Fundamental Concepts and Algorithms. *Cambridge University Press*.

Zhang, C. and Zhang, Z. 2003. Trading Replication Consistency for Performance and Availability: an Adaptive Approach. *Proceedings of the International Conference on Distributed Computing Systems*. 687 – 695.

Zhao, W., Xu, X., Wang, Z., Zhang, Y. and He, H. 2010. A Dynamic Optimal Replication Strategy in Data Grid Environment. *Proceedings of International Conference on Internet Technology and Applications*. 1 – 4.

Zhao, W., Xu, X., Xiong, N., and Wang, Z. 2008. A Weight-Based Dynamic Replica Replacement Strategy in Data Grids. *Proceeding of IEEE on Asia-Pacific Services Computing Conference*. 1544 – 1549.

Zhou, W., Wang L., and Jia, W. 2004. An Analysis of Update Ordering in Distributed Replication Systems. *Future Generation Computer Systems*. 20 (4): 565 – 590.



اونيورسيتي ملايسيا قهغ

UNIVERSITI MALAYSIA PAHANG