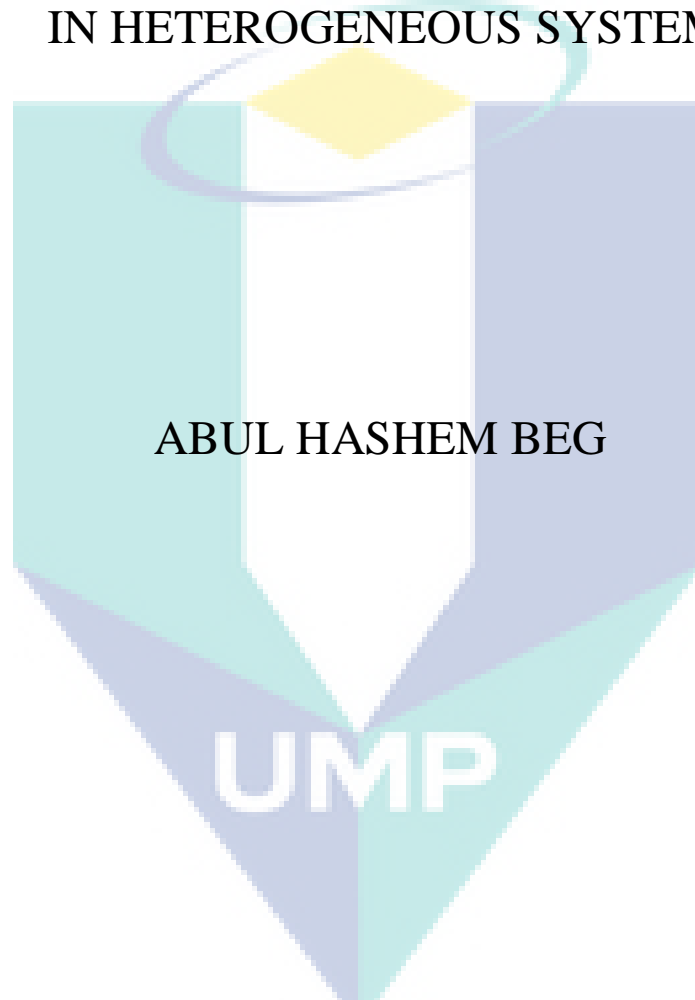


ADAPTIVE PERSISTENCE LAYER FOR
SYNCHRONOUS REPLICATION (PLSR)
IN HETEROGENEOUS SYSTEM



MASTER OF SCIENCE (COMPUTER)

UNIVERSITI MALAYSIA PAHANG

**ADAPTIVE PERSISTENCE LAYER FOR SYNCHRONOUS REPLICATION
(PLSR) IN HETEROGENEOUS SYSTEM**



ABUL HASHEM BEG

**Thesis submitted in fulfillment of the requirements
For the award of the degree of
Master of Science (Computer)**


UMP


**Faculty of Computer Systems & Software Engineering
UNIVERSITI MALAYSIA PAHANG**

JULY 2011

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion this thesis is satisfactory in terms of scope and quality for the award of the degree of Master of Science (Computer).



Signature : 

Name of Supervisor : DR. NORAZIAH BINTI AHMAD

Position : SENIOR LECTURER
FACULTY OF COMPUTER SYSTEMS & SOFTWARE
ENGINEERING, UNIVERSITI MALAYSIA PAHANG


Date : JULY 11, 2011



UMP

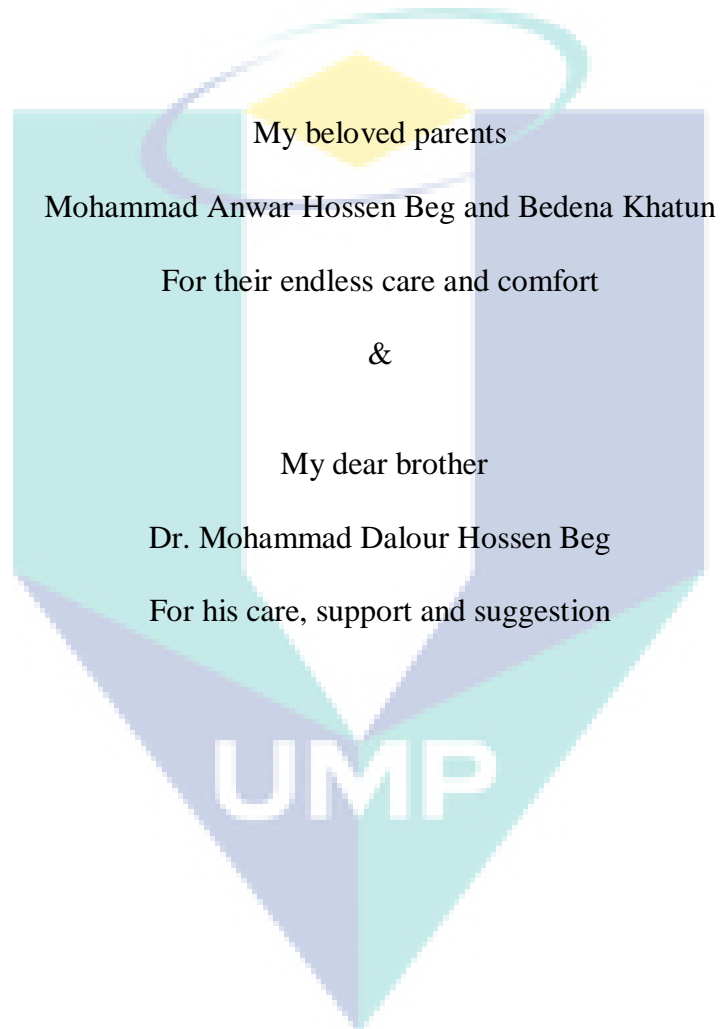
STUDENT'S DECLARATION

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged. The thesis has not been accepted for any degree and is not concurrently submitted for award of other degree.

Signature : 
Name : ABUL HASHEM BEG
ID Number : MCC09004
Date : JULY 11, 2011



This thesis is dedicated to



ACKNOWLEDGEMENTS

I would like to express my most sincere gratitude to the supervisory committee Dr. Noraziah Binti Ahmad for her continuing support, professional guidance and for giving me an opportunity to learn what research is all about. Special gratitude also to Associate Professor Dr. Ahmed N Abdalla for his contributions, guidance and time towards this research.

Sincerely thanks should be forwarded to Vice Chancellor University Malaysia Pahang (UMP), Profesor Dato' Dr. Daing Nasir Ibrahim for the GRS scholarship and also sponsorships to the International Conference on ICSCCT 2010.

I extend my deepest gratitude to Dr. Nubli, Dr. Anwar and Dr. Tutut Herawan for their great effort and support during my study period.

Special gratitude also to my family, especially to my father, Mohammad Anwar Hossen Beg; my mother, Bedena Khatun; my sister Aleya Akter; my brothers Dr. Dalour Hossen Beg, Altab Hossen Beg; my sister in-law Afroza Khanom Lovely and niece Nilanchol Chhnooa Beg; for their patience and morale support.

Finally, I thank to all my friends especially Khandaker Fazley Rabbi, Ainul Azila Che Fauzi, Nawsher Khan, Noriyani Mohd Zin, El Rasheed Sultan and Mohammed Fakheraldien who have contribute this research.

The logo of Universiti Malaysia Pahang (UMP) is a large, stylized letter 'U' shape. The top part of the 'U' is a light blue oval. The two vertical sides of the 'U' are light blue. The bottom part of the 'U' is a light blue inverted triangle. The letters 'UMP' are written in white, bold, sans-serif font across the bottom of the 'U' shape.

UMP

ABSTRACT

Nowadays, in the grid community, distributed and clustering system, a lot of work has been focused on providing efficient and safe replication management services through designing of algorithms and systems. For many reasons, businesses or specially enterprise business or industrial business use replication. Therefore, replication is a useful technique for distributed systems. It can improve the performance and the reliability of a database application. In addition, it can be considered as a data backup method in case of hardware failure, software corruption or even a natural disaster. A change of the main database is reflected, forwarded and applied at each of the replicated server which might be in a remote location. Replication in the heterogeneous system is a very promising and challenging platform which is a compound of multi environment. Proper mechanism is significantly required in order to manage the complex heterogeneous data replication. In this research, Persistence Layer for Synchronous Replication (PLSR) has been proposed to support heterogeneous systems. The main objective of this technique is to develop an adaptive persistence layer which consisted of reliable and smooth replication. This technique also introduces a multi thread based persistence layer, which supports early binding and parallel connection to the servers. All the replication servers established its connection through interfaces. Furthermore, similar with the Service Oriented Architecture (SOA) and the structure is flexible enough to modify i.e.; adding and removing replication server. The PLSR is proposed based on the multithreading technique in order to avoid the dependency of replicated server from the main server and to make the enterprise software more enhanced so that the system will never be unstable during system up-gradation or system crashes. Consequently, the implementation of this technique will be applicable to enterprise application such as bank, insurance, group of companies as well as a small and medium organization such as NGO. The new replication process will also be used in e-commerce application to secure user transaction information. The motivation of implementation is to make sure the data replication is easy to maintain and cost effective. The PLSR architecture, model, workflow and algorithms are described. The PLSR has been developed using Java Programming language. The system requirements also have been elaborated. The experimental main server and replication servers were established in Windows and Linux platform using the local area network (LAN). Finally, series of experiments have been carried out by using different servers. The snapshot of implementation showed that the proposed framework works successfully with replicating data in different operating systems. The result shows that PLSR performs outstandingly and the value is 83.2 % and 2.49% than SQL server for transactional insert and synchronization in compare to time (seconds).

ABSTRAK

Dewasa ini, di dalam komuniti grid, sistem teragih dan sistem klustering, banyak usaha telah difokuskan untuk menyediakan servis replikasi yang cekap dan selamat dengan merekabentuk algoritma dan sistem. Perniagaan atau khususnya perniagaan enterprise atau perniagaan industri banyak menggunakan replikasi disebabkan pelbagai faktor. Oleh kerana itu, replikasi adalah teknik yang berguna untuk sistem teragih Ianya dapat meningkatkan prestasi dan kebolehpercayaan terhadap aplikasi pangkalan data. Selain daripada itu, ia juga boleh dianggap sebagai kaedah sandaran data sekiranya berlaku kegagalan peranti keras, kerosakan perisian mahupun bencana alam. Perubahan dari pangkalan data utama akan diteruskan dan digunakan pada setiap pelayan yang mungkin terletak pada lokasi berjauhan. Replikasi dalam sistem heterogen adalah platform yang mencabar serta menjanjikan masa depan yang cerah yang mana terdiri daripada persekitaran pelbagai. Mekanisme yang tepat diperlukan untuk menguruskan replikasi data heterogen yang kompleks. Dalam kajian ini, Persistence for Layer Synchronous Replication (PLSR) telah dicadangkan untuk menyokong sistem heterogen. Tujuan utama teknik ini adalah untuk membangunkan lapisan persistensi adaptif yang terdiri dari pada replikasi yang boleh dipercayai. Teknik ini juga memperkenalkan lapisan persisten berasaskan multi bebenang yang menyokong sambungan awal dan sambungan selari ke pelayan. Semua pelayan replikasi mendirikan sambungannya melalui antaramuka, menyerupai Service Oriented Architecture (SOA) dan strukturnya cukup fleksibel untuk diubahsuai seperti menambah dan membuang pelayan replikasi. Dalam kajian ini, PLSR dicadangkan berdasarkan pada teknik multi bebenang untuk mengelakkan pelayan direplikasi bergantung dengan pelayan utama serta untuk meningkatkan taraf perisian Enterprise sehingga sistem itu tidak akan menjadi tidak stabil pada masa sistem dinaik taraf atau sistem terjadinya kerosakan sistem. Oleh itu, pelaksanaan teknik ini sesuai untuk aplikasi enterprise seperti sekumpulan syarikat, insurans, bank organisasi kecil dan sederhana seperti NGO. Proses replikasi baru juga akan digunakan dalam aplikasi e-dagang untuk melindungi maklumat transaksi pengguna. Motivasi dari pelaksanaan tersebut adalah untuk memastikan replikasi data mudah untuk penyelenggaraan dan pengurangan kos. Reka bentuk model, alur kerja dan algoritma PLSR dijelaskan. PLSR telah dibangunkan dengan menggunakan bahasa pengaturcaraan Java. Keperluan sistem juga telah dihuraikan. Server utama yang diuji dan pelayan replikasi dibangunkan di platform Windows dan platform Linux dengan menggunakan rangkaian kawasan tempatan (LAN). Akhir sekali, suatu siri percubaan telah dilakukan dengan menggunakan pelayan yang berbeza. Hasil kajian dengan snapshot menunjukkan bahawa model yang dicadangkan berfungsi dengan baik bagi mereplikasi data dalam sistem operasi yang berbeza. Keputusan menunjukkan PLSR platform berfungsi dengan hebat dengan nilai 83.2% dan 2.49% daripada SQL server untuk memasukkan transaksi dan sinkronisasi dengan perbandingan masa (saat).

TABLE OF CONTENTS

	Page
SUPERVISOR’S DECLARATION	ii
STUDENT’S DECLARATION	iii
DEDICATION	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
ABSTRAK	vii
TABLE OF CONTENTS	viii
LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF ABBREVIATIONS	xiv
CHAPTER 1 INTRODUCTION	
1.1 Introduction	1
1.2 Data Replication	2
1.3 Problem Statement	5
1.4 Objectives of Research	6
1.5 Scopes of Research	7
1.6 Organization of Thesis	7
CHAPTER 2 LITERATURE REVIEW	
2.1 Introduction	8
2.2 Replication Model	8
2.2.1 Synchronous Replication	14
2.2.2 Peer-to-Peer Replication	16
2.3 Persistence Layer	18
2.4 Heterogeneous system	21

2.5	Transaction	23
2.6	SQL server replication	25
2.7	Multi-threading technique	27
2.8	Conclusion	28

CHAPTER 3 METHODOLOGY

3.1	Introduction	29
3.2	Framework of PLSR model	29
3.3	Complete flowchart of PLSR model	30
3.4	Structure of persistence layer	32
3.4.1	Exception Handler	33
3.4.2	Global configuration	33
3.4.3	Connecting sting	33
3.4.4	Look up service	33
3.4.5	Heterogeneous System	35
3.5	PLSR algorithm	37
3.5.1	Persistence layer algorithm	38
3.5.2	Connection string algorithm	39
3.5.3	Lookup service algorithm	40
3.5.4	Utility algorithm (add previous record)	41
3.5.5	Utility algorithm (synchronous data)	42
3.6	Replication time calculation	43
3.7	Conclusion	43

CHAPTER 4 IMPLEMENTATION AND RESULTS

4.1	Introduction	44
4.2	Programming implementation	44
4.3	Hardware and Software components	51

4.4	PLSR Environment	53
4.4.1	Experiment 1	53
4.4.2	Experiment 2	54
4.5	PLSR implementation	56
4.6	Result and Discussion	65
4.7	Conclusion	71

CHAPTER 5	CONCLUSION AND FUTURE WORK	
5.1	Introduction	72
5.2	Conclusion	72
5.3	Future Work	73
REFERENCES		75
APPENDIX		83
BIODATA OF THE AUTHOR		88
LIST OF PUBLICATIONS		89



UMP

LIST OF TABLES

Table No.	Title	Page
2.1	Basic comparison of replication time between transaction and merge insert	27
3.1	PLSR algorithm variable and definition	37
4.1	Server main components specifications	52
4.2	System development tools specifications	52
4.3	The local IP address for each server based on SQL Server	54
4.4	The local IP address for each server based on MySQL Server	55
4.5	Transactional insert time between SQL Server and PLSR (SQL Server)	66
4.6	Transactional insert time between SQL Server and PLSR (MySQL Server)	67
4.7	Total replication process time between SQL Server and PLSR (SQL Server)	68
4.8	Total replication process time between SQL Server and PLSR (MySQL Server)	69

A large, semi-transparent watermark logo for UMP (Universitas Muhammadiyah Palembang) is centered on the page. It features a shield-like shape with a yellow top section and blue bottom sections, with the letters 'UMP' in white at the bottom.

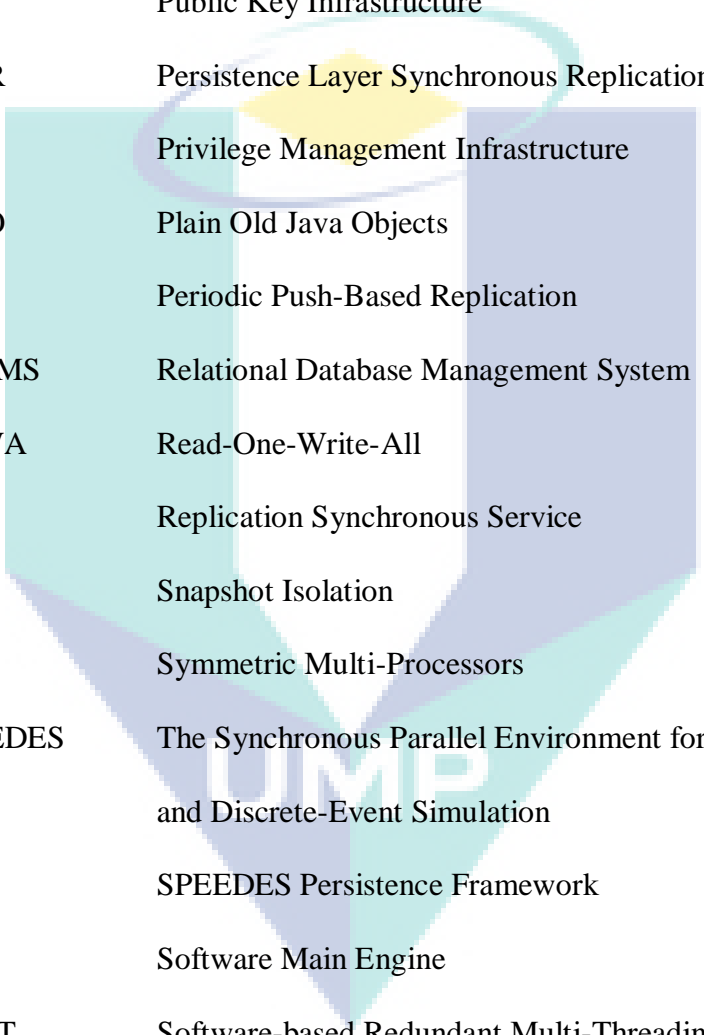
LIST OF FIGURES

Figure No.	Title	Page
2.1	Data distribution	10
2.2	Data consolidation	11
2.3	Bidirectional replication	12
2.4	Synchronous replication	15
2.5	Peer-to- Peer replication	17
2.6	Persistence layer in J2EE applications	21
2.7a	COMMIT balances BEGIN TRANSACTION by reducing the @@TRANCOUNT one	24
2.7b	A single ROLLBACK undo the entire transaction	25
2.8	SQL server replication architecture	26
3.1	Structure of replication process	30
3.2	Flow chart of persistence layer replication process	31
3.3	Structure of persistence layer for synchronous data replication	32
3.4	Lookup service in PLSR	35
3.5	Heterogeneous replication process	36
3.6	Persistence layer algorithm	39
3.7	Connection string algorithm	40
3.8	Lookup Service algorithm	41
3.9	Utility algorithm (add previous record)	42
3.10	Utility algorithm (synchronize data)	43
4.1	Product authority information	45
4.2	Project view of persistence layer APP	46
4.3	Demonstration of server configuration file	47
4.4	Demonstration of connection string file	48
4.5	Source code view of persistence layer Engine	49
4.6	The Initial view of the application currently supporting data	50

	insertion	
4.7	The view after data insertion	51
4.8	Data replication on heterogeneous system with 3 replication servers and SQL Server	53
4.9	Data replication on heterogeneous system with 3 replication servers and MySQL server	55
4.10	The inserted data on the table in the main server (SQL Server)	56
4.11	IP address information of main Server (SQL Server)	57
4.12	The inserted data on the table in a replication server (MS Access)	58
4.13	IP address information of MS Access	59
4.14	The inserted data on the table in a replication server (MySQL in Linux)	60
4.15	IP address information of MySQL Server (Linux)	61
4.16	The inserted data on the table in a replication server (MySQL in Windows)	62
4.17	IP address information of MySQL (Windows Server)	63
4.18	Showing permalink SQLyog link under wine in Ubuntu	64
4.19	Running SQLyog in Ubuntu	65
4.20	Comparative time for transactional insert	69
4.21	Comparative time for synchronization	70

LIST OF ABBREVIATIONS

BHR	Bandwidth Hierarchy Replication
BSL	Boost Serialization Library
CRUD	Create, Read, Update and Delete
CSCW	Computer Supported Cooperative Learning/Work
DAG	Direct Acyclic Graph
DDMS	Distributed Database Management System
DDS	Distributed Database System
DLL	Dynamic Link Library
DML	Data Modification Language
eFRD	efficient Fault-Tolerant Reliability- Driven
EJB	Enterprise Java Bean
FTP	File Transfer Protocol
GUI	Graphical User Interface
HDC	Heterogeneous Distributed Computing
HCS	Hierarchical Cluster Scheduling
HRS	Hierarchical Replication Strategy
IDE	Integrated Development Environments
LAN	Local Area Network
MT	Multi-Threading
MSE	Multi-Staged Engine
ODR	On Demand Replication



ORM	Object/Relational Mapping
OS	Operating System
P2P	Peer-to-Peer Replication
PADS	Parallel and Distributed Simulation
PKI	Public Key Infrastructure
PLSR	Persistence Layer Synchronous Replication
PMI	Privilege Management Infrastructure
POJO	Plain Old Java Objects
PPR	Periodic Push-Based Replication
RDBMS	Relational Database Management System
ROWA	Read-One-Write-All
RSS	Replication Synchronous Service
SI	Snapshot Isolation
SMP	Symmetric Multi-Processors
SPEEDES	The Synchronous Parallel Environment for Emulation and Discrete-Event Simulation
SPF	SPEEDES Persistence Framework
SME	Software Main Engine
SRMT	Software-based Redundant Multi-Threading
URL	Uniform Resource Locator
TRLS	Tree-based Replica Location Scheme
TSPS	Tree-Structured Persistence Server

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Nowadays, in many fields, such as scientific experiments and technological applications generate a huge amount of data. The appropriate use of data sharing and collaboration, these generated data should be shared and distributed in wide area networks. Therefore, the effective management of these wonderful sources of information shared and distributed is becoming a very important topic of scientific research and commercial applications. Thus, the data replication is a very useful technique to manage the large scale data across the widely distributed networks (Sriram and Cliff, 2010; Li and Shen, 2009; Lei et al., 2008).

Several researchers have been carried out regarding the replication process from the last decade. Among them were those by Ibison (2010), Pucciani et al. (2010), Tanga et al. (2010), Lou et al.(2009), Sato et al. (2009), Tong and Shu (2009), Elghirani et al. (2007), Boyera and Hura (2005) and Ma et al. (2004). Those articles revealed that data replication in heterogeneous system are one of the current issues that still unsolved in distributed system. Therefore; the study on this basis is initiated.

A database can be defined as a common set of logically related data that can be designed to meet the necessary information from an organization and to be used by multiple users (Post, 2006; Connolly and Begg, 2005). The emergence of network facilities and communication added to a database system can be taken from centralizing

to a decentralized concept (Bell and Grimson, 1992). Distributed Database System (DDS) is one of the main developments in the field of database, where it moves from centralization that led to the monolithic gigantic database to greater decentralization (Deris et al., 2004; Deris et al., 2001). DDS is defined as a set of multiple independent databases that run on two or more computers that are connected and share data over the network (Post, 2006; Connolly and Begg, 2005). Meanwhile, a Distributed Database Management System (DDMS) can be defined as software that manages the distributed database and makes the distribution transparent to the user (Connolly, and Begg, 2005). Even the commercial system databases such as Oracle Database (Version 11g) provide the necessary support for data distribution and inter database communication (Freeman et al., 2005). These concepts allow higher degrees of distributed and flexibility in distributed databases (Mavromoustakis and Karatza, 2008; Deris et al., 2004). With the development of distributed processing and distributed computing, database research community makes a lot of work to solve the data distribution, distributed design, distributed query processing, distributed transaction management.

The heterogeneous computing system is a very promising and challenging platform that combines with the case of multi-environment. The single parallel architecture based systems is not sufficient enough for a running application to exploit the parallelism. In some cases, Heterogeneous Distributed Computing (HDC) systems can achieve better performance than the single super computer system; moreover, it puts the lower cost than the super computer. However, the HDC system is more exceptions oriented that may put a negative impact on the running application (Tanga et al., 2010).

1.2 DATA REPLICATION

Data replication provides an important role in this involving world of the DDS. Through this technique, an object request (read and write) will be accessed from multiple locations such as Local Area Network (LAN) or in the worldwide distributed network. For example, the results of a student in college will be read and updated by lecturers from various departments. The price of financial instruments will be read and

updated from around the world (Noraziah et al., 2009; Chidambaram et al., 2008; Gu et al., 2008).

Replication environment commonly use two approaches, namely asynchronous or synchronous. With asynchronous replication, changes are made one after a certain time with a lot of data from the master/main database to the different other database. Synchronous means changes made immediately once some data transaction occurs to the master/main database. Using synchronous replication, an update of transaction results immediately replicates the update to all other databases (Urbano et al., 2003). Thus, synchronous replication provides tight consistency between data stores. The meaning is that, the latency between data consistency is zero. If any copy is updated, at the same time the update applied to all other copies within the same transaction. So the data at all sites is always the same and accurately consistent and there is no matter from which replica the updated organized. On the other hand, asynchronous replication provides loose consistency between data stores. The meaning is that, the latency between data consistency is always greater than zero. If one copy is updated, the changes will be broadcasted and applied to the other copies within separate transactions. These copy changes can occur seconds, minutes, hours or even days later. Therefore, a certain degree of lags always exists between the organizing transaction that has committed and the effects of the transaction available on the other replicas (Buertha, 1997).

Replication is a process that copies and maintains database object, such as tables, in multiple databases or even codes into a distributed environment (Urbano et al., 2003). Since database replication maintains the same copy to other remote locations, thus it can improve the performance of a database application. Replication can also be considered as a data backup method which supports to the database application during hardware failure, software corruption or even a natural disaster. A change of main database is reflected, forwarded and applied at each of the replicated servers that usually located on a remote environment (Filip et al., 2009; Caviglione and Cervellera, 2007; Kim et al., 2007). Although, the definition of replicated database and a distributed database similar in some case, there have some divergence. In the definition of distributed database, data

is available at many locations, but a particular table resides at only one location (Wang and Li, 2006; Kosar and Livny, 2005). For example, the employees table resides at only the *pah.employee* database in a distributed database system that also includes the *kl.employee* and *kn.world* databases. However, replication means that the 100% same data at other locations (Bost et al., 2009; Jianfeng et al., 2008). Moreover, Data replication can be driven by programs which transport data to some other location and then loaded at the receiving location which may be filtered and transformed during the process. It should be considered in a replication process that must not interfere with existing applications and should have the minimal impact on production systems. Thus the replication processes, need to be managed and monitored (Gu et al., 2002). The definition of database replication can be stated as improvement of data access time, transaction time and provides fault tolerance by maintaining and managing multiple copies of data (e.g. files, objects, databases or parts of databases) at different locations (Ibej et al., 2005).

Replication balances the data transaction, and it provides fast, local access to shared data over multiple databases (Hao.W et al., 2008; Lin, 2007). Thus replication can also perform as load balancing. Recently, database replication protocols have been designed using Snapshot Isolation (SI) replicas and follow the Read-One-Write-All (ROWA) approach as well as the transactions are, firstly executed in a delegate replica and their updates (if required) are propagated to the rest of the replicas at the committed time (Inigo et al., 2011).

1.3 PROBLEM STATEMENT

Data replications used in the various field such as bank, insurance, group of industries to protect their secure data to prevent unwanted crashes. Data replication in terms of duplication of data creates a backup copy of the data on the different servers. In the current enterprise software system, typically there used a persistence layer which persists in different current objects, which in terms help the application to avoid fault tolerance. So basically, on an enterprise system, replication helps to avoid fault of the data server system. Currently, in the data replication system consist of the following impediments:

- i. Usually replication process depends on the main server
- ii. Introducing the up-gradation of the replication process usually mute or pause the system for a routine of time
- iii. Fail or cashes of the main server, usually make the entire system stop working
(For a database driven system)

In the grid environment, Sato et al. (2009) proposed an approach for the clustering base replication algorithm. The goal is to create a technique to determine optimal file replication strategies. Their approach outperformed groups file stored in a grid file system according to the relationship of simultaneous file access and determines locations and movement of replicas of file clusters from the observed performance data of file access and implementation specification was in Linux 2.6.1.8. However, researchers do not consider the heterogeneous system and also the replication in the grid environment needs a lot of inter connection speed, which is in gigabyte.

Elghirani et al. (2007) proposed an approach in an intelligent replication framework for data grid. The main goal of their approach is to create a replica management service that interrogates replica placement optimization mechanisms and dynamic replication techniques, coupled with computation and job scheduling

algorithms for better performance in data grids. They use dynamic ordinary replication strategies and replica placement schemes. The result shows that their approach improves the job execution time by 10-23%. However, the replica management is only coupled with computational job scheduling, which actually better performed in Symmetric Multi-Processors Server (SMP).

Persistence data layer is one of an important part in the information system design. It is the foundation of the system performance and its migration ability. Lou et al. (2009) studied a reflected persistence data layer framework based on O/R mapping. They presented five modules: data loadable module which are data write module, database services module, primary key cache module and paging cache module for persistence layer. However, the reflection is not native to the OS. A lot of execution handling mechanisms should be included into the system. Besides replication using the reflection mechanism is a very slow process and takes a lot of memory and sometimes causes a buffer overflow.

1.4 OBJECTIVES OF RESEARCH

This research concentrates on the synchronous replication in the heterogeneous environment. Persistence Layer for Synchronous Replication (PLSR) is proposed based on the multithreading technique in order to avoid the dependency of replicated server from the main server and to make the Enterprise software more enhanced so that the system will never be unstable during system up-gradation or system crashes. This framework supports the heterogeneous system. It can be implemented in SQL Server, MySQL, and MS Access in Linux or Windows environment. Therefore, it is easy to maintain and cost effective.

The objectives of this research are as follows:

- i. To propose and to develop a new framework and algorithm of persistence layer for synchronous replication in heterogeneous system

- ii. To analyze the performance of the proposed framework and algorithm

1.5 SCOPES OF RESEARCH

The scopes of this research are as follows:

- i. Design a new framework using multi-threading technique for synchronous replication
- ii. Develop a new algorithm for synchronous replication and implemented in the heterogeneous environment
- iii. Develop new replication process which will be applicable for enterprise application such as bank, insurance and group of companies
- iv. The new replication process will be used in e-commerce application to secure users transaction information
- v. The new replication also can be used in small and medium organization such as NGO, Institutes to make their data more reliable and portable

1.6 ORGANIZATION OF THESIS

This thesis has been prepared to give details on the facts, observations, arguments, and procedures in order to meet its objectives. Chapter 1 generally gives the brief background of database replication, the problem statement, objectives and scope of the research. Chapter 2 presents the literature review of replication model, synchronous replication, peer-to-peer replication, persistence layer, heterogeneous system, transaction, data grid solution and multi-threading technique. Chapter 3 carries out the structure of the proposed persistence layer. Its different modules and algorithm of the PLSR and definition of the notation have been described. Chapter 4 addresses the implementation of PLSR framework and compares the performance with other replication techniques. The conclusions of the present research are summarized and presented in Chapter 5. Suggestion and recommendations for the future work are also presents in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter reviews some of the foremost techniques namely replication model, synchronous replication, Peer-to-Peer (P2P) replication, persistence layer, heterogeneous system, transaction, data grid solution and Multi-Threading (MT) Technique. A review of other relevant research studies is also provided.

2.2 REPLICATION MODEL

Replication is mainly used to store some or all data items redundantly at multiple sites. The main objective of replication is to increase the system reliability and application performance. In the grid community, distributed and clustering system, a lot of work has been focused on providing efficient and safe replication management services through the designing of algorithms and systems. Replication technology creates data replication on the right node from where the data transmission becomes faster. As such a network is in some remote location separated from the main server, and the data transmission rate is too high. Thus a replication server can be created on that remote location which in terms helps the remote system to reduce data transmission impediments and improve visit delay, bandwidth consumption and system reliability (Gao and Liu, 2007).

Businesses or specially Enterprise business or industrial business use replication for many reasons. The business requirements can be categorized (Gu et al., 2002).

- i. Distribution of data to other locations
- ii. Consolidation of data from other locations
- iii. Bidirectional exchange of data with other locations

i. Distribution of data to other locations

Distribution of data involves to move all or a subset of data to one or more locations. The data is copied from a central data warehouse. Subsets of data can be copied to the different data station to afford different groups of users with local access. It is therefore, possible to use enterprise data with business intelligence tools, while maintaining safety and performance of production applications. Distribution of data can be used for other applications in the same or different environments. This is called simply copying data from the main server to another replication and / or main server. It may be necessary a complex data transmission for a new application. The new application can be a web application, bought package, or a distributed desktop application.

Data replication can also be used to provide a scalable application when migrating from one environment to another. It can also be used for Legacy data copied to the new environment for the reference by new applications and, unless the legacy applications are migrated to the new environment. Figure 2.1 shows two target servers replicating from a single source server (Gu et al., 2002).

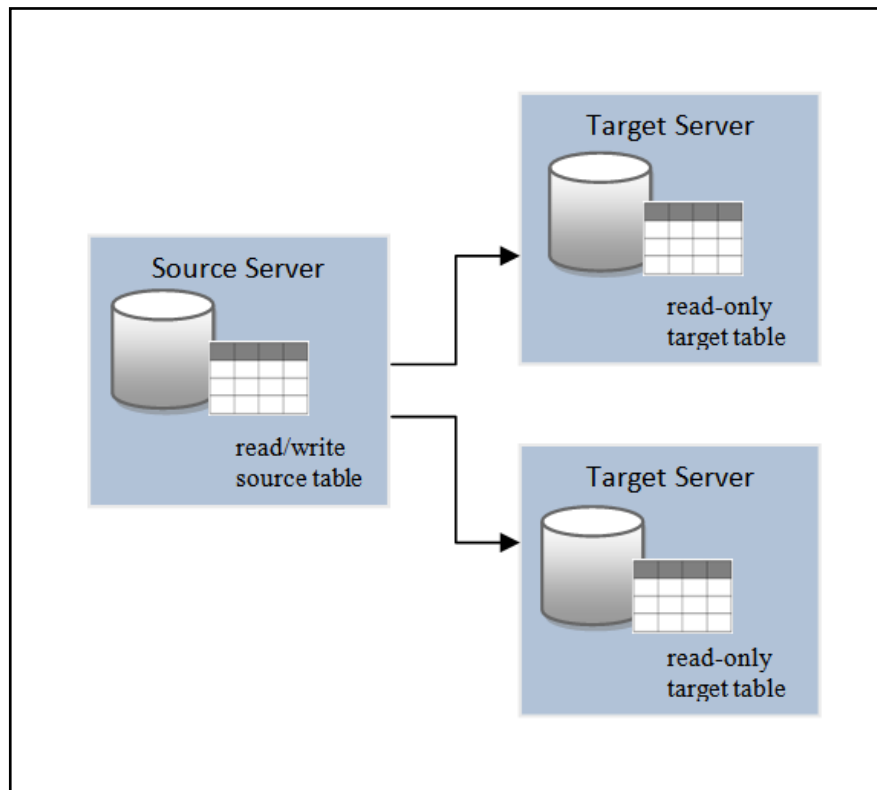


Figure 2.1: Data distribution (Gu et al., 2002)

The two target servers are copying different subsets or transformations of the data on different schedules.

ii. Consolidation of data from remote systems

An enterprise application can obtain data on many distributed systems. Retail stores have data at each store. Manufacturing companies have data on each plant. Insurance companies contain data at each branch office or on each sales person's laptop or computer. Therefore, replication can copy changes from each of the individual distributed sites to a central point for analysis, reporting, and enterprise application processing. Figure 2.2 shows a target server replication from two source servers (Gu et al., 2002).

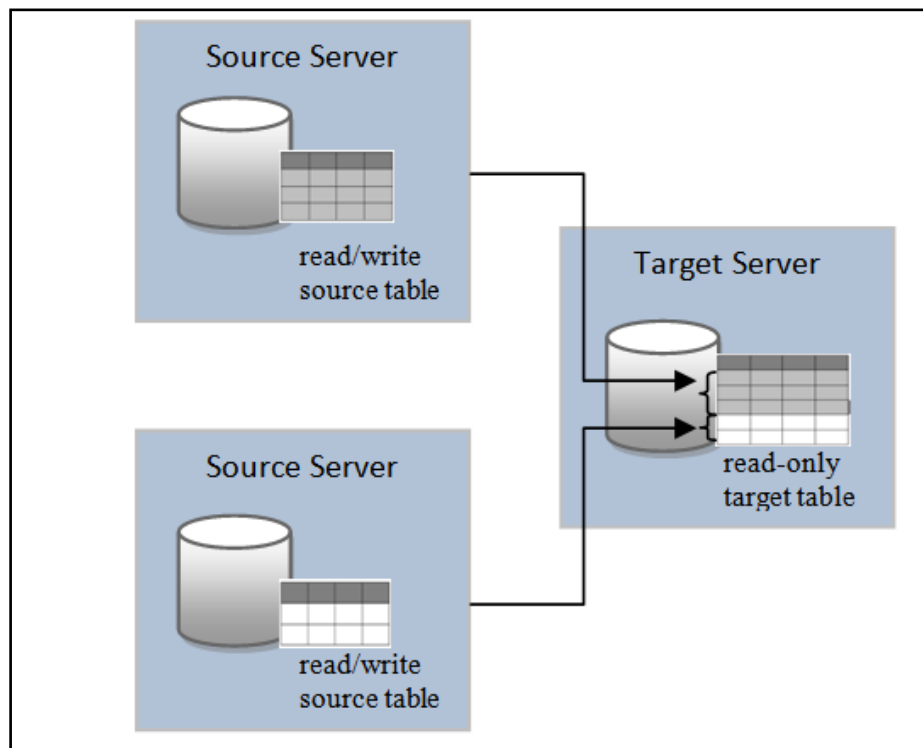


Figure 2.2: Data consolidation (Gu et al., 2002)

iii. Bidirectional exchange of data

This type of data replication functions as a coordinated fashion. In this system, one location works as the master location and distributes changes to the target's locations. Changes made at the target flow to other destination sites by the master. Bidirectional replication can be used for mobile applications, where the goal may be a computer at a branch or a delivery truck. The connection can be made through telephone lines, while efficiency is important. Figure 2.3 illustrates the bidirectional replication with a designated master (Gu et al., 2002).

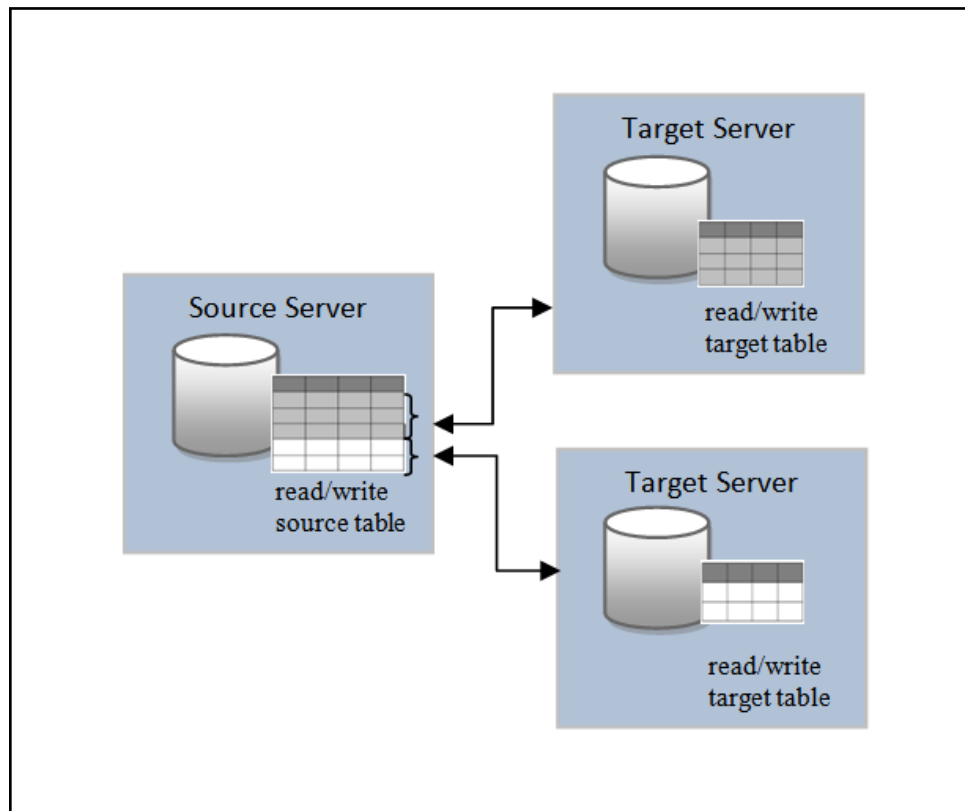


Figure 2.3: Bidirectional replication (Gu et al., 2002)

Ahmad N et al. (2010) proposed new Neighbour Replica Transaction Failure Framework (NRTFF) in data grid. They developed a reliable system for managing transaction on neighbor replication data grid (NRDG) to maintain data availability, fault tolerance and avoid the deadlock. When two neighbour replicas have failures at a specific given point of time, the transaction can perform without waiting to obtain a majority quorum. In particular, $T_{\gamma x q, 1} \in T_{\gamma x q}$ at a primary replica becomes as $\cdot, 1 T_{\gamma x q}$. $\cdot, 1 T_{\gamma x q}$ will change an access permission mode of data file x . Then it acknowledged the client for an updated the process and also commits the transaction changes. Their implementation also showed that managing transactions on NRTFF provided fault tolerance capabilities that allow it to withstand failure both in handling quorum locking and the transaction execution.

Tang et al. (2006) proposed architecture for data replication and job scheduling to reduce the job turnaround time remarkably and evaluate the performance of the scheduling heuristics combination with different replication algorithms. The adaptive object replication algorithm (Wujuan and Veeravalli, 2008), replicate on line request for serving random arriving requests in a distributed network system and consider the costs involved that occur in servicing requests, like the I/O cost, control-message and data-message transferring cost. In another paper (Zhoua and Xu, 2007), the authors proposed an efficient algorithm of video replication and placement on a cluster of streaming servers. The goal of their video replication strategy is to duplicate videos according to their popularity levels and also places the replicas entirely on servers. Litke et al., (2007) present a fault tolerant model. For this model, they used four different algorithms and implement their model based on task replication for task scheduling in Mobile Grid systems. The efficient task was designed for diverse failure probabilities of the resources and operates in Grid middleware. The authors introduced an indirect replication algorithm (Wang and Li, 2006) followed the inherent characteristic of a distributed storage system and the P2P model. The characteristic of the algorithm is to provide less granularity of replication, less bandwidth and storage costs, and also provides higher availability, durability, and security.

Some research activities have been investigated MANET-specific solutions to bind/rebind to new discover distributed resources, thus enabled wireless clients to automatically redirect requests to service components. Bellavista et al. (2005) proposed REDMAN middleware to manage, retrieve, and disseminate replicas of data/service components to cooperated nodes in a dense MANET.

Sashi and Thanamani (2011) proposed a modified Bandwidth Hierarchy Replication (BHR), which reduces the data access time by avoiding unnecessary replication in the data grid network. The modified BHR can increase the data availability by replicating the files within the region to region header and also stored them in the site where the file has been accessed frequently. It can minimize the job execution time. In

another paper (Horri et al., 2008), the authors proposed another BHR algorithm, and they used three hierarchical structures. They concentrated to the problem of replication and scheduling.

There are some mixed approaches to use of static and dynamic replica placement. The static replica placement algorithm are used to optimize the average response time and dynamic replica placement algorithm are used to re-allocate replicas to new candidate sites if a performance metric degrades extensively (Rahman et al., 2006). P2P strategy (Shena and Zhu, 2009) and parallel transmission (Wang.C et al., 2007) have been also used for replica placement in data grids. Nam et al. (2004) have been proposed a Tree based Replica Location Scheme (TRLS) to decide the replica locations. The main objective of TRLS is to minimize the sum of storage cost and communication cost of the replication and used the linear programming for problem solved. Youn et al. (2002) proposed hybrid protocol using trees and grid replication. P2P and DHTS approaches have also been practices for replication in grid systems. Knezevic et al. (2006) proposed a DHT based replication protocol that can adjust autonomously the number of replicas to deliver a configured data availability guarantee.

2.2.1 Synchronous Replication

Synchronous remote data replication is the right solution for organizations looking for the fastest possible data recovery, minimal data loss, and protection against the problems of integrity of the database. It ensures that a remote copy of data, which is identical to the primary copy, is created when the primary copy is updated. In synchronous replication, an input or output updates operations is not considered done until the end of confirmation both primary and mirrored sites. An incomplete operation is rolled back at both locations to ensure that the remote copy is always an exact mirror of the primary as shown in Figure 2.4 (Hitachi data system, 2007).

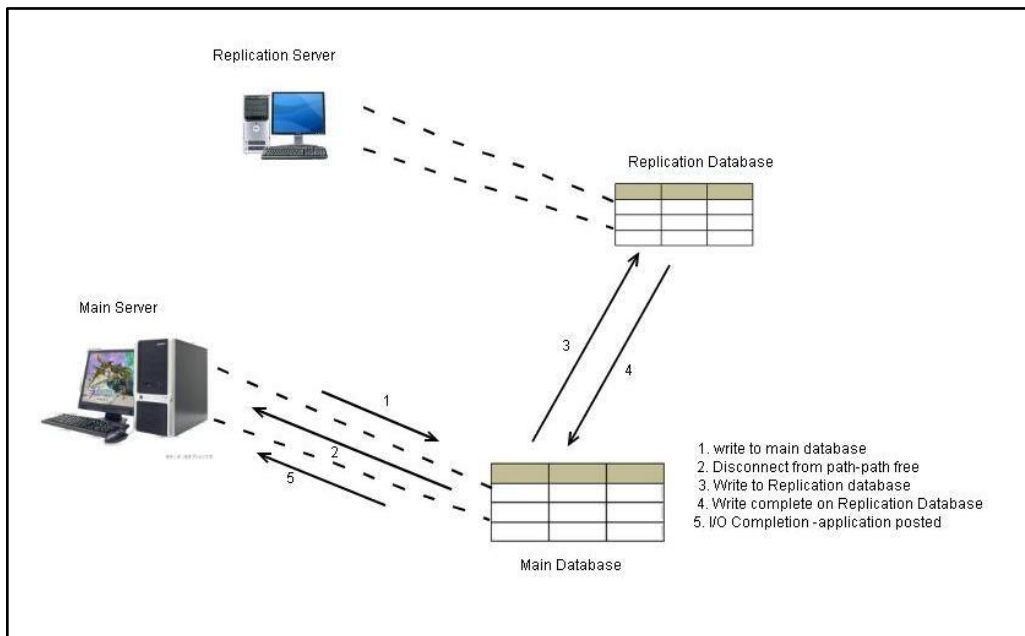


Figure 2.4: Synchronous replication (Hitachi data system, 2007)

The main benefit of synchronous replication is that data can be recovered quickly. Operations in the remote, mirror site may begin immediately when the primary site stopped should operations at the primary site be disrupted. Only some operations in the process at the instant of disruption may be lost. Because neither the primary nor a remote site has a record of these transactions, the database rolls back to the last committed state (Hitachi data system, 2007).

In synchronous replication there have several schemes, including all-data-to-all-sites (full replication) and some-data-item-to-all sites & some-data-item-to-some-sites (partial replication). Among all of these ROWA (read one write all) is one of the simplest techniques. An object is allowed to read by the read operation and write operation writes all the copies of the object (Stockinger, 2001).

Ahmad et al. (2010) presented an algorithm to manage replication and synchronization transaction system using ROWA-MSTS. They deployed their algorithm in the real time application in the distributed environment. Read-One-Write-All

Monitoring Synchronization Transactions Systems (ROWA-MSTS) have been developed based on ROWA technique. The ROWA-MSTS techniques handle each site either it is operational or down and to communicate each other. The researcher used VSFTPD (GPL licensed FTP server for UNIX systems) as an agent communication between replicated servers.

A set of services for the replica content synchronization has been presented (Ciglan and Hluchy, 2007). The researchers designed their system to support relational and XML data resources to provide the interoperability of heterogeneous systems. The main objectives of their approach are virtualization of underlying data resource heterogeneity, provision of rich functionality that can enable the implementation of a number of proven consistencies protocols. Experiment shows that they are able to replicate relational and XML databases and can manage the distinct replicas in different systems.

The applications based on the distributed system became more and more popular by the benefit of the development of the network technology. In distributed application, same copy of data stored in different databases thus data synchronization is very important. Hao.Y et al. (2008) analyze the data synchronization technology and describe the advantages of the Oracle stream mechanisms. They compare the result based on CPU utilization and processing time. The result shows that for insertion, update and delete operation, the CPU utilization rate of synchronous replication is very low compare to asynchronous replication and Oracle streams.

2.2.2 Peer-to-Peer (P2P) Replication

This is another type of bidirectional replication that does not have a designated master. Each site copies changes from all other sites directly. This is called multi-master or P2P replication. It can be used to maintain disaster recovery, providing fail-over systems for high availability and load balancing queries across multiple sites. Figure 2.5 shows a P2P replication (Gu et al., 2002).

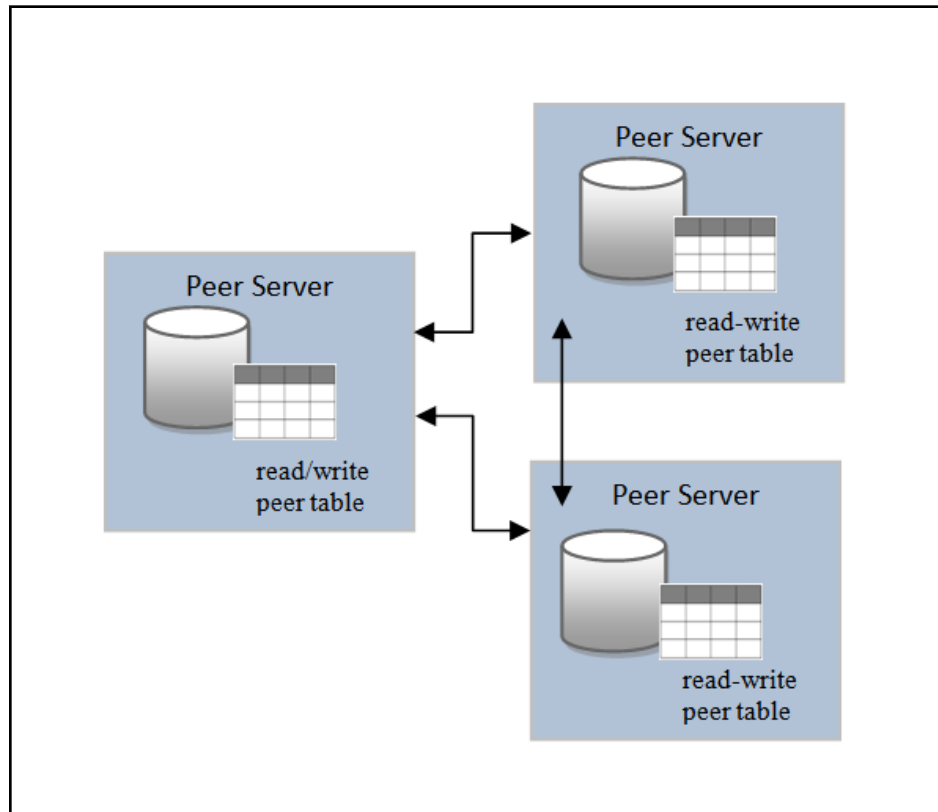


Figure 2.5: Peer-to-Peer replication (Gu et al., 2002)

In the P2P network environment, using dynamic replication proposed a load sharing technique (Chidambaram et al., 2008) providing and improving access performance. The authors proposed two load sharing techniques, which use data replication. At the first technique there has been used a Periodic Push-based Replication (PPR) to reduce the hop count (the number of legs traversed by a packet) and at the second technique it uses On Demand Replication (ODR) that performs and improves access frequency. However, they proposed two algorithms: improve access performance on a P2P network. File replication facilitates efficient file consistency maintenance and is a widely used technique for high performance in P2P content delivery networks. Caviglione and Cervellera (2007) introduce a P2P based system for content replication. They evaluate the process through a discrete-time system, where decisions are taken by the tracker at starting of the each temporal stage, and they also consider a single-stage

optimization because peers can enter and leave the system in every time unpredictably (Caviglione and Cervellera, 2007). Secure content access and replication in pure P2P networks addressed in (Palomar et al., 2008) through the idea of attribute certificates that does not rely on the existence of a Public Key Infrastructure (PKI), Privilege Management Infrastructure (PMI), or any other form of centralized authority for content authentication and access control in pure P2P networks. Optimization of a P2P system for efficient content replication has been proposed in (Cervellera and Caviglione, 2009). They introduced a model to addressing single peer quantities such as bandwidths and chunk completion and also present a procedure for the optimal control of the replication process, through the characterization of connections among peers and the management of their bandwidth shares.

2.3 PERSISTENCE LAYER

Persistence layer provides an abstract interface to the data access layer which is a part of an information storage mechanism. Such an interface is abstract and independent of storage technology. The typical features include:

- i. Store and/or Retrieve of the whole database objects
- ii. Abstraction of the database cursor with all instances of a given type
- iii. All available transaction support, including open, commit, abort and rollback
- iv. Data session management
- v. Data querying support

Usually persistence layers are building from at least two internal layers of an application: It first includes an abstract interface and second is a set of binding to each targeted database. In implementation there may have more than two internal divisions between the logic layer and storage mechanism layer (Open EHR, 2007).

Zhang et al. (2010) studied on data persistence layer. Based on the data persistence layer, they established data dictionary persistence layer and business layer of data dictionary to reduce the workload of database management to attain the maintenance of the database model and data persistence layer. In their model they showed that they can maintain the synchronization of the database and data persistence layer with doubly persistence layer while database model changed. The persistence layer framework considered mainly of two parts. One was data map known as Sql Map and the other was data access object. In their data dictionary persistence layer, they configured different XML files according to different data dictionaries to map and realized access and operation of data dictionary. During the changes of database model, firstly they changed corresponding list or field in the database. Secondly they, changed corresponding XML file according to the corresponding relationship between sqlMaps and XML files in the data persistence layer.

In another paper, Wu et al. (2010) presented a data persistence layer of multitier web application, which enabled the developers to interact with a relational database by an object-oriented programming surface. To implement the data persistency, they use the Object/Relational Mapping (ORM). The feature was to map from java classes to database tables, which provided data query and relational facilities. Their approach outperformed the improvement of data access efficiency and ensures the Web application's quality.

Persistence layer is not only used in data dictionary and load balancing systems but also used in simulation checkpoint and restart, and it is very important in parallel and distributed system. Qiao et al. (2006) describe two frameworks SPEEDES Persistence Framework (SPF) and Boost Serialization Library (BSL) and applied persistence framework in parallel and distributed systems. The main contribution of their work tested a C++ template based persistence framework BSL in a Parallel and Distributed Simulation (PADS) application.

In a collaborative computing environment, the collaborative applications required a simple and transparent persistence middleware to deal with complex data accesses. Wang.C.M et al. (2005) proposed a data persistence mechanism and implement a persistence server, called Tree-Structured Persistence Server, which known as TSPS. Their TSPS allowed states of collaborative applications to stored in a tree fashion beside tables. Their main goal to develop the TPS is to serve as a persistence layer to support a project of computer collaborative work.

Essmann et al. (2007) proposed a distributed persistence layer for Computer Supported Cooperative Learning/Work (CSCW) application. Their concern is to provide a unified persistence layer for all applications accessing the distributed knowledge spaces. That can allow developers to implement views that access and change distributed objects like local ones. Their approach is quite similar to the concept of generative communication from parallel computing.

Nowadays, J2EE technology has been widely used in enterprise applications. Usually, a multilayered model is used to encompass client layer, web presentation layer, business logic layer, and database layer. To access durable stores, usually, relational databases, J2EE have two means including JDBC. The first one is the standard API provided by J2SE for relational database management system access, and the second one is the entity beans, an Enterprise Java Bean (EJB) component type dedicated to model a persistent entity (Di et al., 2010). In another paper, Zhou et al. (2010) proposed an Object Relational Mapping (ORM) to provide a transparent persistence layer for Plain Old Java Objects (POJO). Their main objective is to improve the performance and availability of J2EE systems. In their system persistence layer is separately interconnected between business logic layer and databases, shown in Figure 2.6.

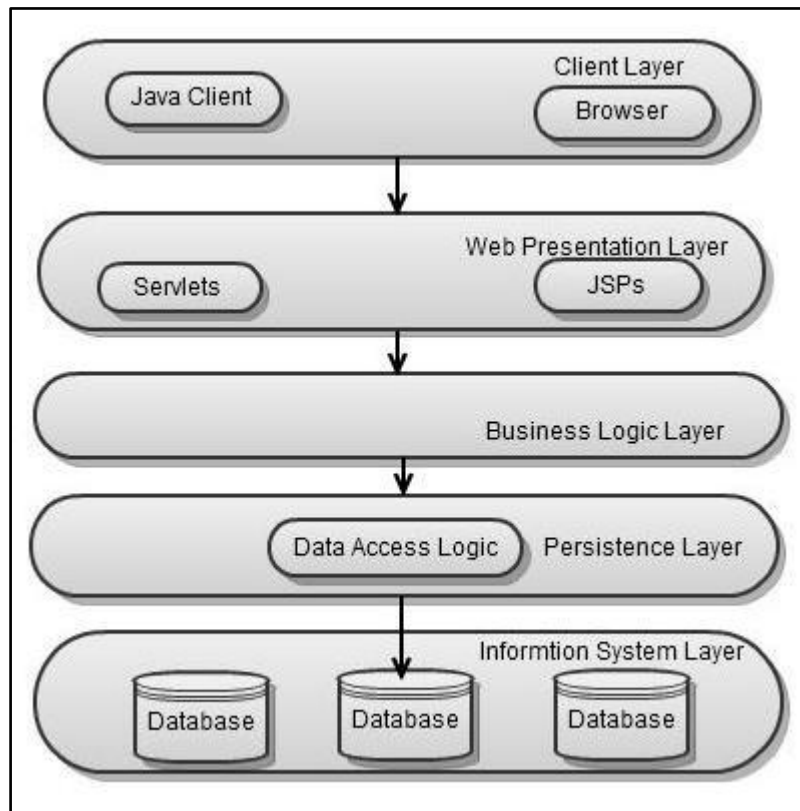


Figure 2.6: Persistence layer in J2EE applications (Zhou et al., 2010)

2.4 HETEROGENEOUS SYSTEM

A Distributed Heterogeneous Computing (DHC) is a collection of autonomous dissimilar computing machines that are linked by a network and synchronized by software that functioning as a single powerful computing facility. As computer tasks can be broken into different parts so it is possible to distribute the task for parallel execution. A DHC system has some advantages over homogeneous computing because some parts of an application perform better in some system, and some parts may perform better in another system (Boyera and Hura, 2005). The homogeneous computing system is easier to control as the processing times are not dependent and identically with an arbitrary distribution (Tong and Shu, 2009).

Several systems have been increasingly used in scientific applications and commercial applications, such as real-time safety-critical applications. Qin and Jiang (2006) proposed an efficient fault-tolerant reliability-driven algorithm known as eFRD for precedence constrained tasks in real-time heterogeneous systems. In their work, they comprehensively address the issues of fault-tolerance, reliability, real-time, task precedence constraints on the heterogeneous system.

Heterogeneous database management systems are also used in various administrative domains in the grid environment. Pucciani et al. (2010) presented a data consistency service that is called CONStanza. They designed and implemented a grid based data replication and synchronization system for updating and synchronizing of both flat as well as a relational database. Their system allows for replicating updates from a commercial system (Oracle) to an open source system (MySQL). Their service can synchronize four wide area distributed MySQL slave databases with an Oracle master database, and their result shows that for 1000 rows insertion time taken near about 12 seconds.

Recently, heterogeneous systems have emerged as a major high-performance computing platform in parallel and distributed system. Chen et al. (2008) proposed an isospeed-efficiency model based on mark speed for heterogeneous computing. Chi et al. (2006) proposed App.Net.P2P architecture to implement effective content delivery on peer-to-peer networks for heterogeneous system. The main objective of their App.Net.P2P is to allow delivering intermediate objects to other peers as well as the final presentations. Therefore, the recipient peers can share the intermediate objects and adapt their presentations for other peers using the associated service logic.

In another paper, Ho et al. (2007) studied of large scale video streaming for heterogeneous network. They explore the impact on the broadcasting scheme coupled with proxy and developed an analytical model, guidelines for resources allocation and transmission strategy to evaluate the system performance in a highly heterogeneous environment. The main concern of their strategy is to make their model applicable for

different system configurations such as centralized/distributed, unicast/broadcast as well as replication/layering.

Ma et al. (2004) proposed a task scheduling algorithm for parallel real-time jobs in heterogeneous system. The main objective of their work is to build a real time model that is applicable to dynamically scheduling multiple Direct Acyclic Graph (DAG) s in the heterogeneous environment. In their model, an assigned processor known as center scheduler is responsible for dynamically scheduling the real-time jobs when they arrive.

2.5. TRANSACTION

A transaction is a group containing a set of tasks which inherent part of any application that collects or manipulates data. SQL server has to make sure the data integrity. That means two users should not modify the same piece of data at the same time. In the SQL server, a transaction is a single statement or multiple Data Manipulation Language (DML) statements executed together. In a transaction, all statements are treated as a group of works. If one of the statements fails then the transaction treated as a fail transaction and the whole transaction roll back to the previous state. As a result, none of the changes are saved. On the other hand, if all the statements are succeeded then the transaction treated as a succeed transaction and commit i.e. save to the database. Therefore, transaction has only two consequences; Success or failure (Tim Ford, 2010; Dewald and Kline, 2002). User can create two groups or more transact-SQL statements into a single transaction by using the following statements:

- i. Begin Transaction
- ii. Rollback Transaction
- iii. Commit Transaction

If anything exception occurs on any of the grouped statements then all changes need to abort. This reversing change process is called rollback. On the other hand, if

everything in the statement within a single transaction occurs successfully, the changes are recorded together in the database. Then these changes are called committed. SQL server contains nest transactions. In nest transaction, a new transaction can start although the previous one is not complete. This transaction operation is happened by issuing the BEGIN TRAN commands. The @@ TRANCOUNT is an automatic variable, queried to conclude the level of nesting. TRANCOUNT 0 indicates no nesting where TRANCOUNT > 1 indicates the level of nesting (where TRANCOUNT is 3 shown in the Figure 2.7a and Figure 2.7b). In the beginning of the nest transaction, the @@ TRANCOUNT automatic variable count increases from 0 to Level of Nesting. On the other hand, in the COMMIT statements, the count decreases by 1 and in ROLLBACK statements, the count is reduced to 0. That means the behavior of COMMIT and ROLLBACK is not symmetric. Figure 2.7a shows that in the nest transactions, COMMIT always decreases the nesting level by 1 and Figure 2.7b shows that in the ROLLBACK command, rollback the entire transaction (Poddar, 2003).

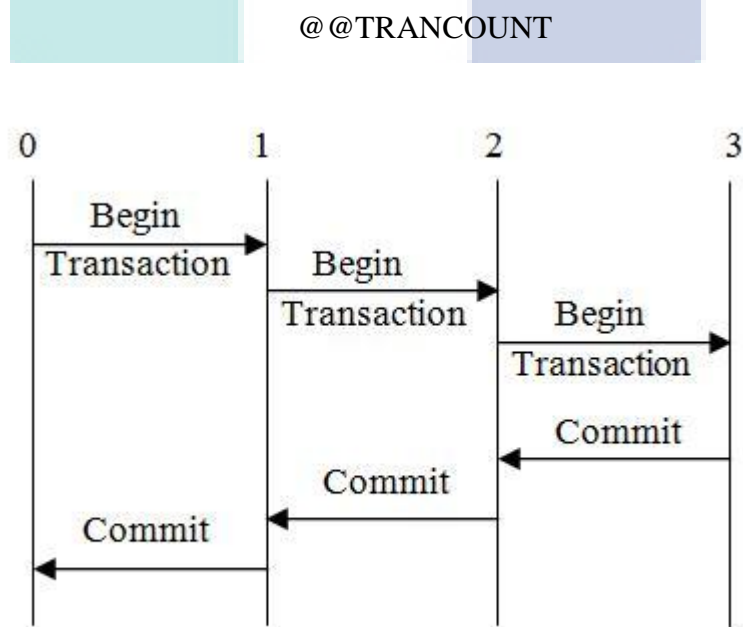


Figure 2.7a: COMMIT balances BEGIN TRANSACTION by reducing the @@TRANCOUNT one (Poddar, 2003)

@@TRANCOUNT

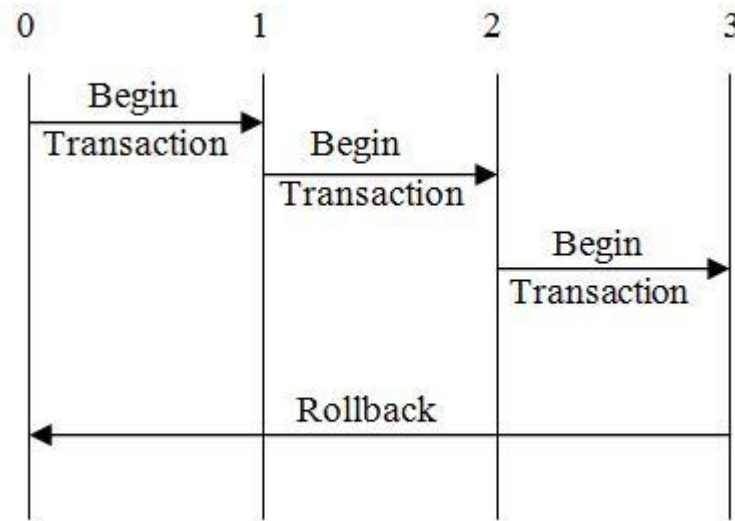


Figure 2.7b: A single ROLLBACK undoes the entire transaction (Poddar, 2003)

2.6 SQL SERVER REPLICATION

SQL server is defined as a Relational Database Management System (RDBMS) from Microsoft that is designed for the enterprise environments. It's run on Transact-SQL (T-SQL), a set programming extension from Sybase and Microsoft. The original SQL Server code has been developed by Sybase in the 1980's, Microsoft. Sybase and Ashton-Tate have collaborated to produce the first product version, SQL Server 4.2 for OS / 2. Later, Sybase and Microsoft SQL Server provide the products. In November 2005, SQL Server 2005 was released. The features of the products were to offer flexibility, scalability, reliability and security of database applications and were easier to build and deploy, reduced the complexity and tediousness involved in database

management. This version also includes a more administrative support (SearchSQLserver.com, 2006).

Gutzait.M (2007) has been described SQL server database design, replication design and architecture. The architectures shows that the databases are replicated with a common structure and the changes are replicated to the subscribers with fewer publications. During the replication consolidated information, users can publish a piece of information about the specific site. This allows users to create one publication and add WHERE clause according to the site or database code. The database servers are connected with the main server. The main database also connected with replication servers. Figure 2.8 show the SQL server replication.

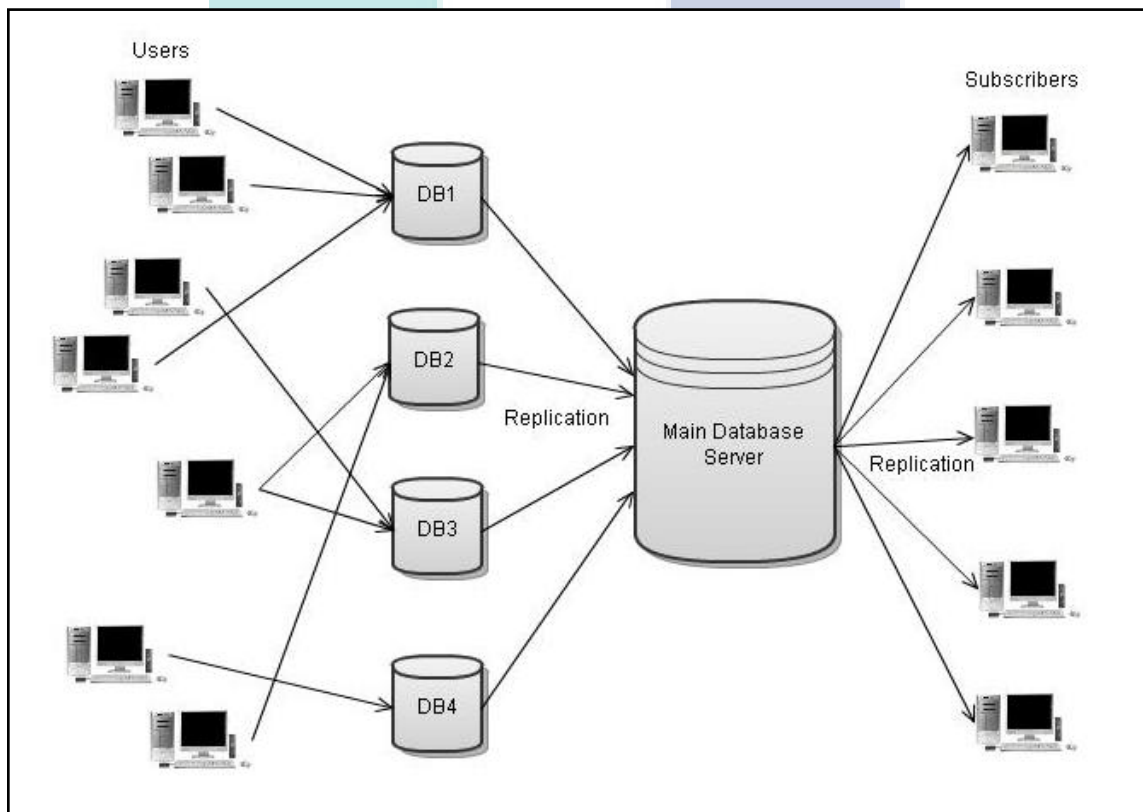


Figure 2.8: SQL server replication architecture (Gutzait.M., 2007)

Ibison (2010) described a basic comparison of SQL server replication times between Merge and Transactional insert and synchronization shown in Table 2.1. The table used a sample table having fixed width columns: The fixed-length data types were chosen because this table was also used to calibrate the problems of network bandwidth. The table data shows that for the inserts, merge takes significantly longer than transactional and snapshot. For a small number of insertions transactional and snapshot are quite the same. When the number of row increases, transactional begins to take longer than the snapshot. Conversely, there are a few rows is not much difference, but as the increase in the number of rows, transactional and snapshot stay very similar, while the merge becomes much larger.

Table 2.1: Basic comparison of SQL Server replication time between Transaction and Merge insert

# rows	Snapshot	Sync	Transactional	Sync	Merge	Sync
	Inserts	Time	Inserts	Time	Inserts	Time
100	0	4	0	0	1	2
500	1	4	1	1	3	6
1000	2	5	2	1	5	12
5000	6	7	7	2	21	42
10000	13	12	26	5	42	96

2.7 MULTI-THREADING TECHNIQUE

Multi-threading is the ability to run multiple processor threads. It seems, at the same time. CPUs are very fast to execute instructions. Modern PCs can run almost one billion instructions per second. Instead of running the same program for a second, the processor executes a program to perhaps a few hundred microseconds, and then switch

to another and work for a short period and so on. It is also possible for an application to have multiple parts that run simultaneously. For example, a background task could be to respond to the mouse, while a file is loaded into RAM, and second task updates a progress bar on the screen (Bolton, 2011).

Wang.C.H et al. (2007) present a Software-based Redundant Multi-Threading (SRMT) approach for transient fault detection. Their technique use compiler to automatically generate redundant threads, and they can run on general-purpose chip multi-processors (CMPs). The result shows that their technique can provide a flexible program execution environment where flexible scheduling legacy binary codes and reliability-enhanced codes can co-exist in a fashion mix-and-match according to the desired level of reliability and software compatibility.

Wei et al. (2009) present the Multi-Staged Engine (MSE) for high performance and flexibility in the application of concurrent continuous query processing, using the pipeline strategy and departs from the continuous query processing on three parallel phases: preprocessing, execution and dispatching modules to improve the parallelism with multi-threaded technology. They developed an algorithm multi-threaded (MTCNN) for k nearest massive continuous query processing. MTCNN algorithm uses parallel threaded workload and cache-conscious implementation of the reorganization to improve spatial and temporal locality.

2.8 CONCLUSION

The definition and functionality of data replication and review of other relevant research studies are covered. Synchronous replication, P2P replication, persistence layer, transaction, heterogeneous system, and Multi-Threading (MT) technique are reviewed with great details.

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION

This chapter carries out the framework of PLSR. The flow chart and structure of PLSR with all possible diagrams have been shown. All the associate modules PLSR have also been described. The details algorithms have been shown as a pseudo code.

3.2 FRAMEWORK OF PLSR MODEL

In the proposed system PLSR modifies the persistence layer to adopt multi processing use of multi-threading. The persistence layer is connected with different database servers, from those one will be the main server that will take care of Create Read, Update and Delete (CRUD) with the entire system and the several others are known as the replicated server. A thread with higher priority, i.e.; main thread keeps the consistent connection with the main server. *Thread* is defined as the single stream of execution within a process. Programs that execute within its own address space are known as *Process*. The persistence layer creates one thread for each of the replication servers. One thread is the higher priority used for main server, and rest of all has the lower priority. As the main server is maintained by a high priority thread, thus the data should be saved or deleted or modified immediately with high priority. On the other hand, replication servers are maintained by a low priority thread. Consequently, the data transaction will be kept in a queue, and then it makes its own copy and does the transaction with that thread. Figure 3.1 shows the structure of the replication process.

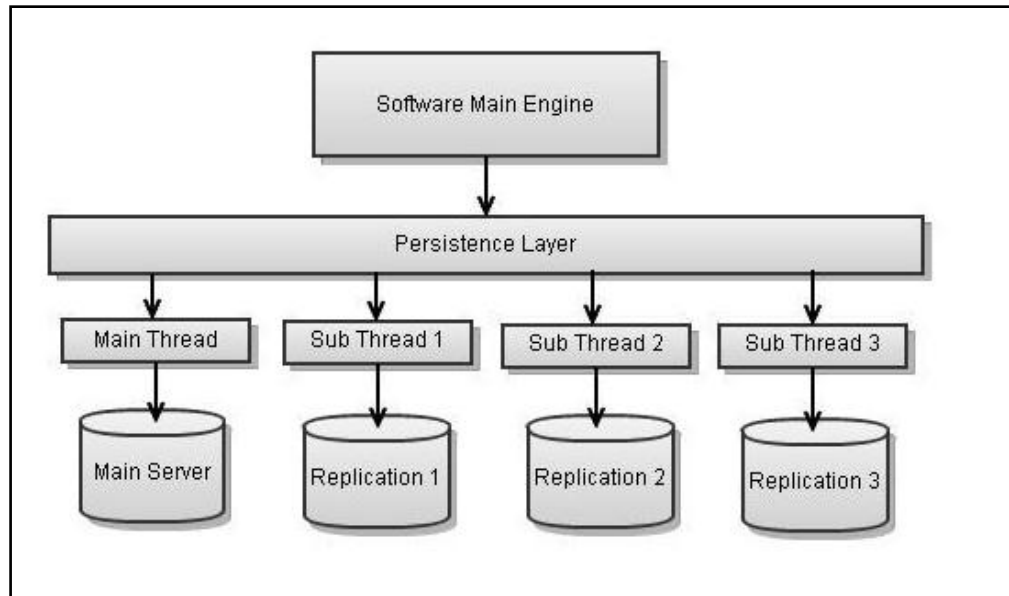


Figure 3.1: Structure of replication process

3.3 COMPLETE FLOWCHART OF PLSR FRAMEWORK

In this technique, the system promotes a GUI which facilitates users to insert data to the system. From GUI, data send to the persistence layer. To configure the servers i.e. to populate the server's information persistence layer check the configuration and connection string file either, which is existed and readable. Exception handler generates messages. Therefore, send to the user. Alike with configuration file Exception handler covers connection string, ether the connection string is readable or not. An unreadable connection string shows a message to the user, other than that the system read connection string where the database connection URL stored as a XML file. The persistence layer creates multithread based on the configuration file, i. e. the definition of main server and the replication servers along with the numbers (number of replication servers). For the main server, persistence layer creates a high priority thread. The high priority thread is responsible for the transactions to the main server. Along with this, for the X number of replication server, persistence layer creates X number of low priority

threads, which provide service for the transaction to the replication servers. A notification is sent to the end user/ administrator when the entire process is finished. Figure 3.2 shows the flowchart of the PLSR.

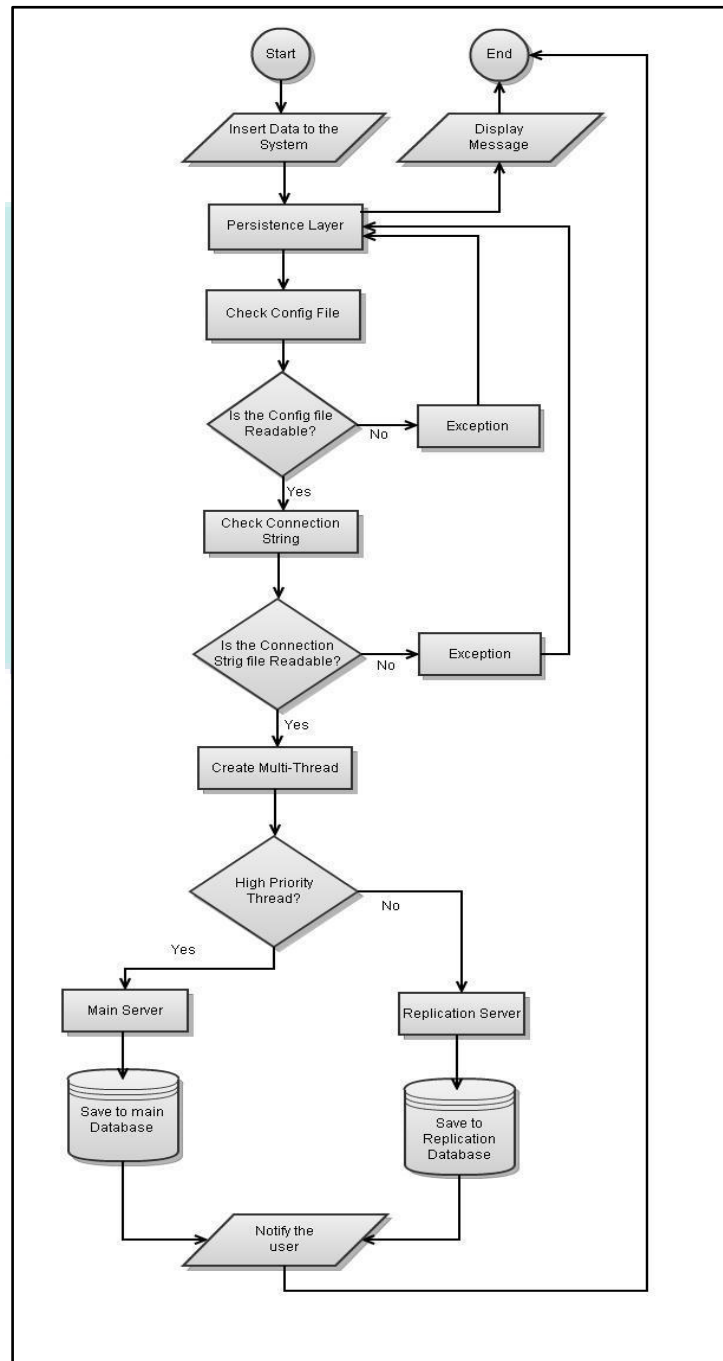


Figure 3.2: Flowchart of persistence layer replication process

3.4 STRUCTURE OF THE PERSISTENCE LAYER

The persistence layer queues all the database transaction for the replicated server which runs in a low priority multi-thread. Despite of this, the persistence layer contains another higher priority thread, which makes a sound transaction with the main server using a higher priority thread. The entire database server's connection defines the connection properties from a XML configuration. The XML architecture is flexible enough to make an addition of other replication servers even doesn't need to harm the whole system. When the database server crashes then the persistence layer start using transaction from one of the replication servers and sends system alert messages to the administrators. It makes no down time to the entire system. The structure of persistence layer for synchronous data replication (Figure 3.3) and then implement of the corresponding function has been described below.

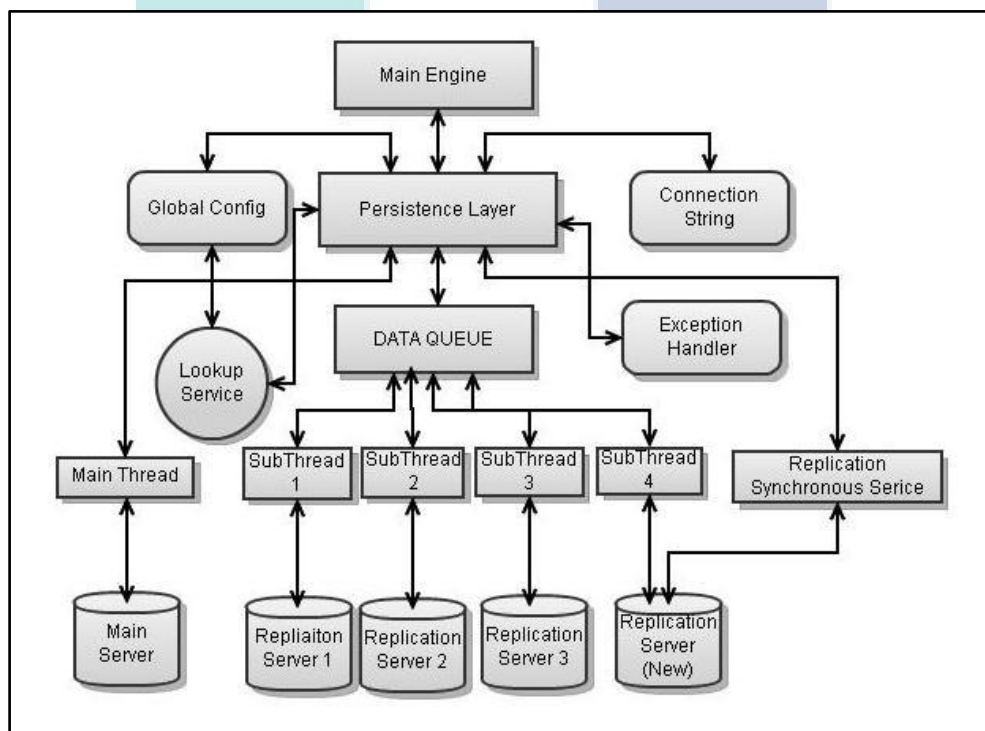


Figure 3.3: Structure of persistence layer for synchronous data replication

3.4.1 Exception Handler

The module/layer Exception is responsible to handle all kinds of exception of the entire system. A database exception handler layer also incorporated within this layer. In any system, typically an exception handler is defined by the occurrence of exceptions, special conditions that change the normal flow of program execution. Such as, memory overflow, null pointer, divided by zero is the special conditions which in terms handled by the Exception layer. In the proposed algorithm, when a database server becomes to overload and/or when main or replication server crashes, is treated as an exception. Thus it creates its own defined exceptions to handle database crashes. As an example, when the main server crashes, the exception layers get acknowledged, and it immediately passes the exception type to the main engine which takes the proper actions.

3.4.2 Global Configuration

Global Configuration is a system resource where all the static data, class, etc. is introduced by its values. As an example in a web enterprise application of the data flow/routing class and the URL has been defined in the global configuration section.

3.4.3 Connection String

Connection string is the technical definition of a software development practice. Connection string is a character string expression that uniquely identifies the data store to use for a particular query or set of queries and methods for connecting. Connection string holds all the connection expression of all the replication servers and main server.

3.4.4 Look up Service

The entire architecture can help an enterprise system to become more secure, reliable through database replication. All the data are synchronously replicated to the

different database servers. The entire system becomes flexible enough to handle any kind of database server. When a new replication server will be added to the system then needs to add some connection string and the API details in the global configuration. It also consists of a lookup service on that configuration file. If something adds or removes from the configuration file then the lookup service updates the system and consist of another service, which known as replication synchronous service. The replication synchronous service updates the newly added replication server with the bulk of data from the main server. This service adds all the data from the main server and updates the newly added replication server.

The synchronous replication runs by the proposed algorithm and perform the entire task with the use of multi-threading. A thread with higher priority keeps the consistent connection with the main server. The persistence layer creates other many threads for all the replication servers. One thread has the higher priority, and all other have the lower priority of the system. Therefore, data that should be saved or deleted or modified will be kept in a queue, and then it makes its own copy and does the transaction with that thread. Figure 3.4 shows how the lookup service looks up the global configuration file and notifies the persistence layer.

The logo for UMP (Universitas Muhammadiyah Purwokerto) is a large, stylized shield shape. It is divided into four quadrants by a white cross. The top-left and bottom-right quadrants are light blue, while the top-right and bottom-left quadrants are light green. In the center of the shield, the letters "UMP" are written in a bold, white, sans-serif font. The shield is positioned in the background, partially overlapping the text of the second paragraph.

UMP

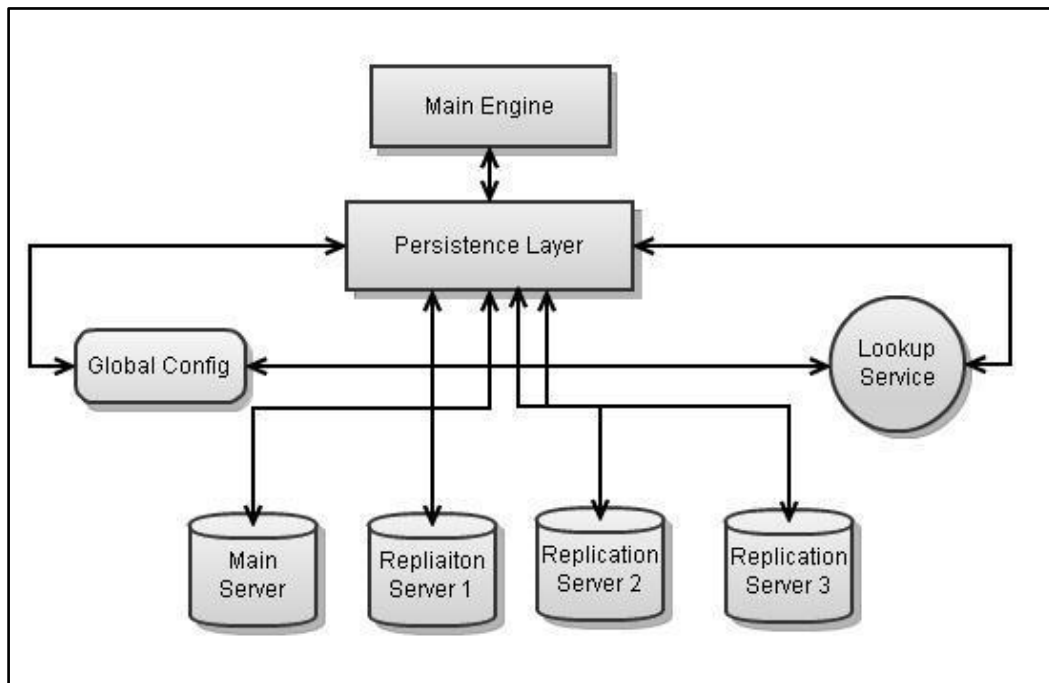


Figure 3.4: Lookup service in PLSR

3.4.5 Heterogeneous Replication Process

In the heterogeneous system, all the replication servers are connected from the persistence layer through the Data Queue (DQ). A multithreading is introduced from the DQ. In this system, data is flow from persistence layer to the main server in a high priority thread and to the different replication server through lower priority thread. The network architecture constructs through different OS. The main server contains the software main engine (SME) which contains the independent Dynamic Link Library (DLL) in C# or jar files in Java to solve complex business logics. On the persistence layer; the data connectivity, complex query execution and the ORM executes. The persistence layer is also responsible for more different service like DQ) and Replication Synchronous Service (RSS) that supports the heterogeneous system as shown in Figure 3.5.

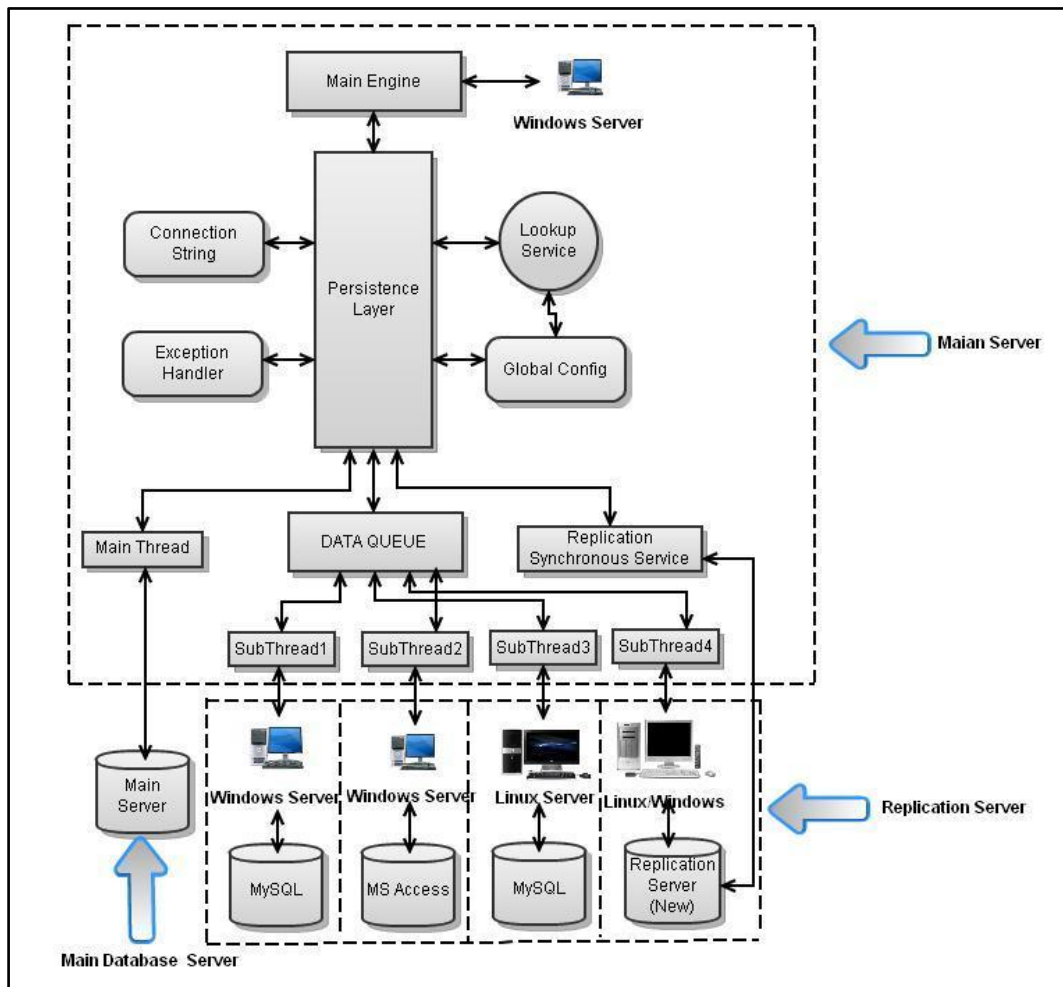


Figure 3.5: Heterogeneous replication process

In the heterogeneous system, all the replication servers are connected from the persistence layer through the DQ. A multithreading is introduced from the DQ. Thus data flows from persistence layer to the main server through high priority thread and to the different replication server through lower priority thread.

3.5 PLSR ALGORITHM

To execute the PLSR framework has been developed different algorithms for different function. These algorithms control most of the significant part of the PLSR. Table 3.1 has been described the nomenclature which has been used in PLSR algorithm.

Table 3.1: Nomenclature of PLSR algorithm

Variable	Definition
N_S	Main Server Name
N_L	Main server Location
MS	Main server in the XML tag
Rep	Replication server in the XML tag
Loc	Location of the replication server in the XML tag
R_S	The list of replication server name
R_L	The list of replication server location which has been read from XML file
Name	Name represents the name of the server
Location	Location represents the network location of the server
Serv	Serv represent the server
C_S	Main server's connection string
C_L	Main Server's operating information
S_N	Replication server's connection string
S_L	Replication server's operating information
T_D	The current data time of the main server
T_F	The lowest traffic information from the database
T_C	From the config it can find the information that when the file created then it represented by T_C
T_M	From the config file, when the file has been modified then it represented by T_M

T_{LM}	The last modifications date represented by T_{LM}
P_C	P_C represent as the set of SQL comamnd
OSInfo	Operating System information
F_{log}	Log file information
C_{sql}	SQL command
$Sync_{list}$	Synchronous SQL command list
$Config_{text}$	Configuration text (string value)

3.5.1 Persistence Layer algorithm

Definition: The x number of main server is equal to x number of thread. Where $x = constant$. For Y number of replication server create Y number of thread. Where Y is 1 to ∞ .

Persistence layer algorithm (Figure 3.6) parses the configuration file and establishes the connection among all the replication database servers and the main server. It makes to queue all database transactions for the replication server and the main server.

PERSISTENCE_LAYER

```

1:   Input: Config.XML
2:   Output: [OutputPersistence]
3:   If ( Config.XML  $\nexists$  ) do
4:       [OutputPersistence]  $\leftarrow$  ( File not found )
5:   end if
6:   Create  $N_S \leftarrow$  null
7:    $N_L \leftarrow$  null
8:    $R_S \leftarrow$  null
9:    $R_L \leftarrow$  null
10:  ConfigText  $\leftarrow$  null
11:  Read Config.XML
12:  ConfigText  $\leftarrow$  Config.XML
13:  for (  $N_S \in$  ConfigText ) do
14:       $N_S \leftarrow$  ( REP  $\rightarrow$  Serv  $\rightarrow$  MS  $\rightarrow$  Name )
15:  end for loop
16:  for (  $N_L \in$  ConfigText ) do

```

```

17:      NL ← ( REP→Serv→MS→Loc )
18:  end for loop
19:      [OutputPersistence]←NS
20:      [OutputPersistence]←NL
21:  for ( RS∈ ConfigText ) do
22:      RS ← ( Rep→Serv→ServerName )
23:      [OutputPersistence]←RS
24:  end for loop
25:  for ( RL∈ ConfigText ) do
26:      RL ← ( Rep→Serv→Location )
27:      [OutputPersistence]←RL
28:  end for loop

```

Figure 3.6: Persistence layer algorithm

3.5.2 Connection String algorithm

Definition: The c number of connection string depends on the addition of main server a , and replication server n . Where $n = 0$ to ∞ .

The connection string basically stores the way to connect to the server. In this system, the connection strings located into an XML file (Figure 3.7). Thus, the connection string algorithm parses the connection string from the XML file and sends the string value to the persistence layer algorithm to establish the connectivity. In this algorithm has been defined the operating information for the main server and the replication server because some exceptions are operating system dependent, like some threading is safe in Linux but usually not safe in XP.

CONNECTION_STRING

```

1:  Input: ConnectionString.XML
2:  Output: [OutputConnection]
3:  If (ConnectionString.XML  $\nexists$  ) do
4:      [OutputConnection]← (File not found)
5:  end if
6:  Create Cs← null

```

```

7:   CL ← null
8:   SN ← null
9:   SL ← null
10:  ConnectionText ← null
11:  Read ConnectionString.XML
12:    ConnectionText ← ConnectionString.XML
13:  for ( CS ∈ ConnectionText ) do
14:    CS ← ( REP → Serv → ConnectionString )
15:  end for loop
16:  for ( CL ∈ ConnectionText ) do
17:    CL ← ( REP → Serv → OSInfo )
18:  end for loop
19:    [OutputConnection] ← CS
20:    [OutputConnection] ← CL
21:  for ( SN ∈ ConnectionText ) do
22:    SN ← ( Rep → RepServ → ConnectionString )
23:    [OutputConnection] ← SN
24:  end for loop
25:  for ( SL ∈ ConnectionText ) do
26:    SL ← ( Rep → RepServ → OSInfo )
27:    [OutputConnection] ← SL
28:  end for loop

```

Figure 3.7: Connection string algorithm

3.5.3 Lookup Service algorithm

Definition: Lookup service L_S is a subset of traffic info SET T_F . Config created date SET T_c . Configuration modified date SET T_M and last modified SET T_{LM} .

In this system, lookup service (Figure 3.8) is the special layer from where lots of synchronization occurred. The system keeps track of lowest traffic information into the database and do a routine check that at when it can do synchronization. Synchronization is necessary if the config file gets changed or if a new replication server added.

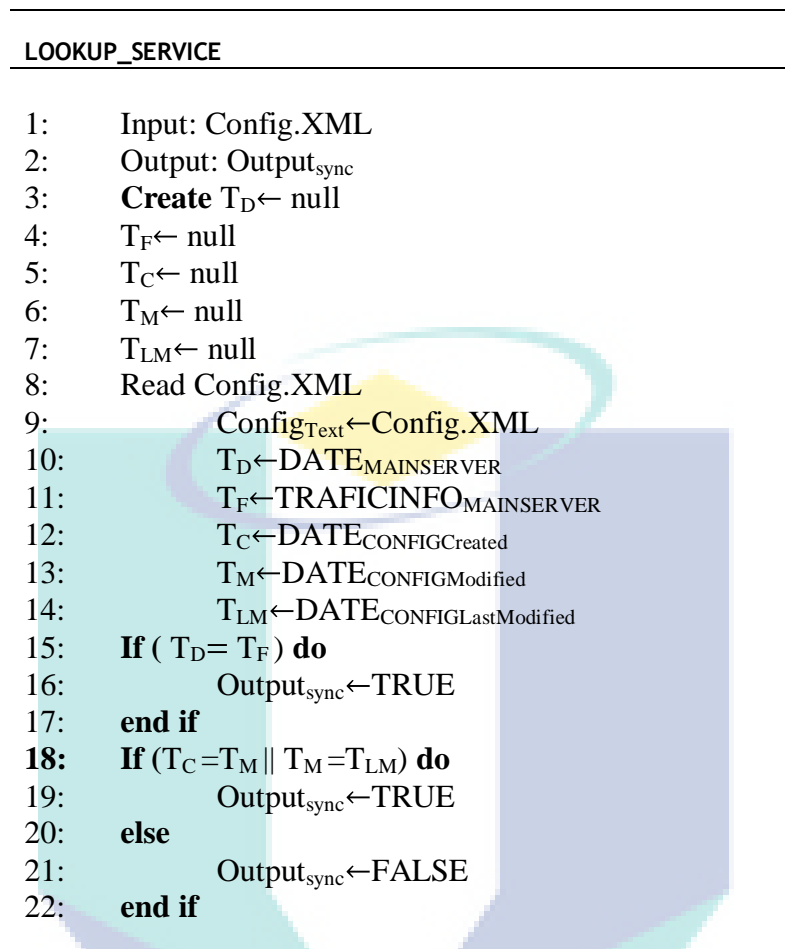


Figure 3.8: Lookup service algorithm

3.5.4 Utility algorithm (add previous record)

Definition: The N_C number of sql command is proportional to the number sql text, T is the log file.

The utility functions show how synchronization happened between the main server and a newly added replication server (Figure 3.9). The system creates a *log.txt* file which stores all the SQL commands for all tables and data. After that it executes those SQL commands to the newly added replication server.

UTILITY_ADD_PREVIOUS_RECORD

```

1:   Input: Replicationserver
2:   Output: [Synclist]
3:   Create TD ← null
4:       Flog ← null
5:       [Csql] ← null
6:   Read Replicationserver → Flog
7:       Flog → [Csql]
8:   for (P ∈ Flog) do
9:       P → [Synclist]
10:  end for loop

```

Figure 3.9: Utility algorithm (add previous record)

3.5.5 Utility Algorithm (Synchronize data)

Definition: The S_c numbers of executable sql command are exists in command XML file.

This is another utility function to synchronous data between main server and replication server (Figure 3.10). When any data could not replicate to the replication server then it stores to an XML file as a SQL command. After that at the lowest traffic time it parses all the SQL command and execute to the replication server.

UTILITY_SYNCHRONIZATION_DATA

```

1:   Input: CommandXMLFile.XML
2:   Output: [Outputsql]
3:   Create PC ← null
4:       ConfigText ← null
5:   Read CommandXMLFile.XML
6:       ConfigText ← CommandXMLFile.XML
7:   for ( PC ∈ ConfigText ) do
8:       [Outputsql] ← PC

```

```
9:   end for loop
```

Figure 3.10: Utility algorithm (synchronize data)

3.6 REPLICATION TIME CALCULATION

The total replication time (RT) has been calculated by using equation (1) (Beg et al., 2011).

$$RT = \sum (TT+ST) \dots \dots \dots (1)$$

Here, RT represents the replication time, TT represents Transactional time and ST represents Synchronous time.

For 10000 rows of data insertion in SQL server, [refer Table 2.1 in section 2.5]

$$\begin{aligned} RT &= (26+5) \text{ Seconds} \\ &= 31 \text{ Seconds} \end{aligned}$$

For 10000 rows of data insertion in PLSR (SQL) server, [refer Table 4.5 in section 4.5]

$$\begin{aligned} RT &= (4.786+0) \text{ Seconds [ST=0, Because PLSR TT and ST done at the same time]} \\ &= 4.786 \text{ Seconds} \end{aligned}$$

3.7 CONCLUSION

The framework and flow chart of PLSR are presented in this chapter. The structure of the persistence layer and heterogeneous replication process also has been described. The detail's algorithms have been shown as a pseudo code. These algorithms control most of the significant part of the PLSR. After describing the details PLSR design, these algorithms make a better understanding on an entire PLSR approach.

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 INTRODUCTION

This chapter described the implementation of the heterogeneous synchronous replication. The purposes of this implementation are to illustrate the algorithms described in the previous chapter and to show that PLSR supports the heterogeneous systems and can use in practical applications. Finally, the results have been compared with other replication process.

4.2 PROGRAMMING IMPLEMENTATION

The programming implementation has been developed using Java programming language. NetBean 6.9.1 has been used to write the source code of persistence layer. This is because NetBean is comfortable Integrated Development Environments (IDE) to implement Java source code. NetBean is developed and maintained by Sun Microsystems. After installing NetBean, it can be accessible to the start menu. NetBean has its own file format to maintain the source code. MySQL, SQLServer and MSAccess have been used as the Database. To implement the SQL Query testing in MySQL has been used SQLyog, which is a free tool. SQL Server has its own SQL IDE. To connect to the database from Java has been used JavaMySQL connector API, and to connect to MSSQL has been used JavaSQLServer connector API. Finally, Java can easily connect to MS Access from the user control panel. The screen shot and the usability of the experiment tool has shown and described below:

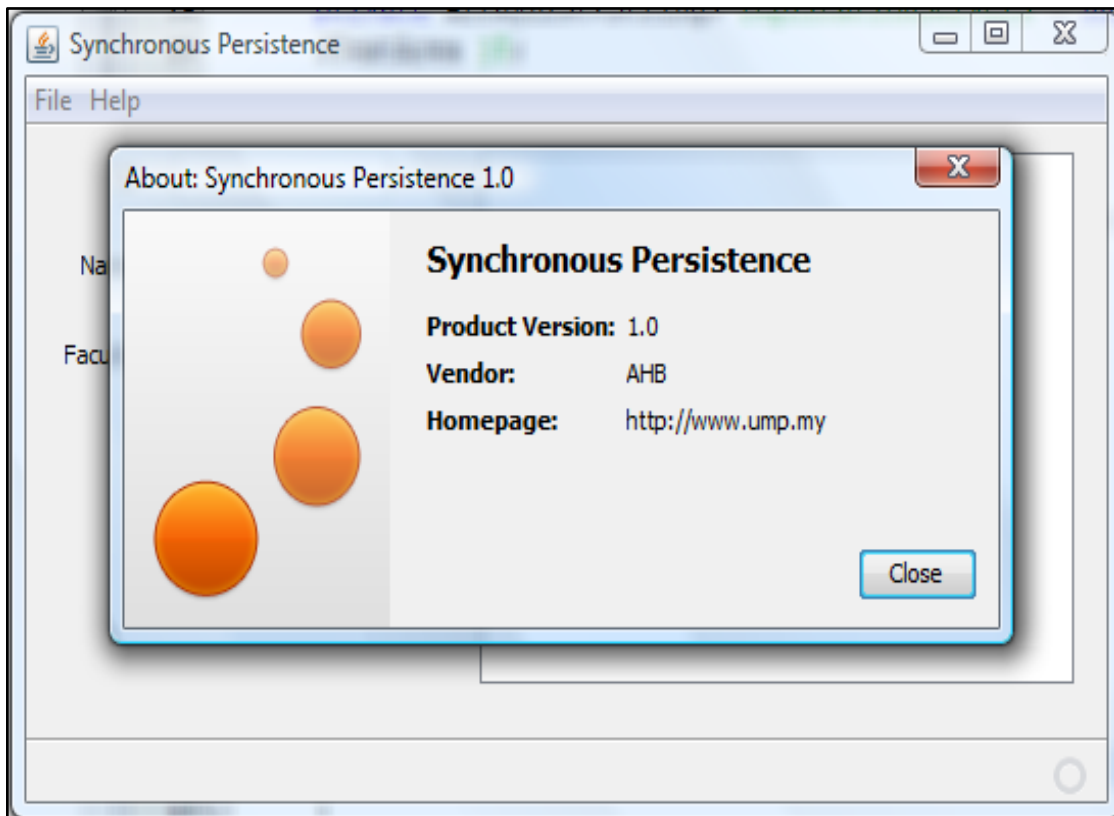


Figure 4.1: Product authority information

Figure 4.1 shows the first page of tool information and the authorization of the application which belongs to the help menu. It is showing the product version, vendor info and the home page of the copyright owner. The product version is 1.0 and the vendor name has been assigned AHB.

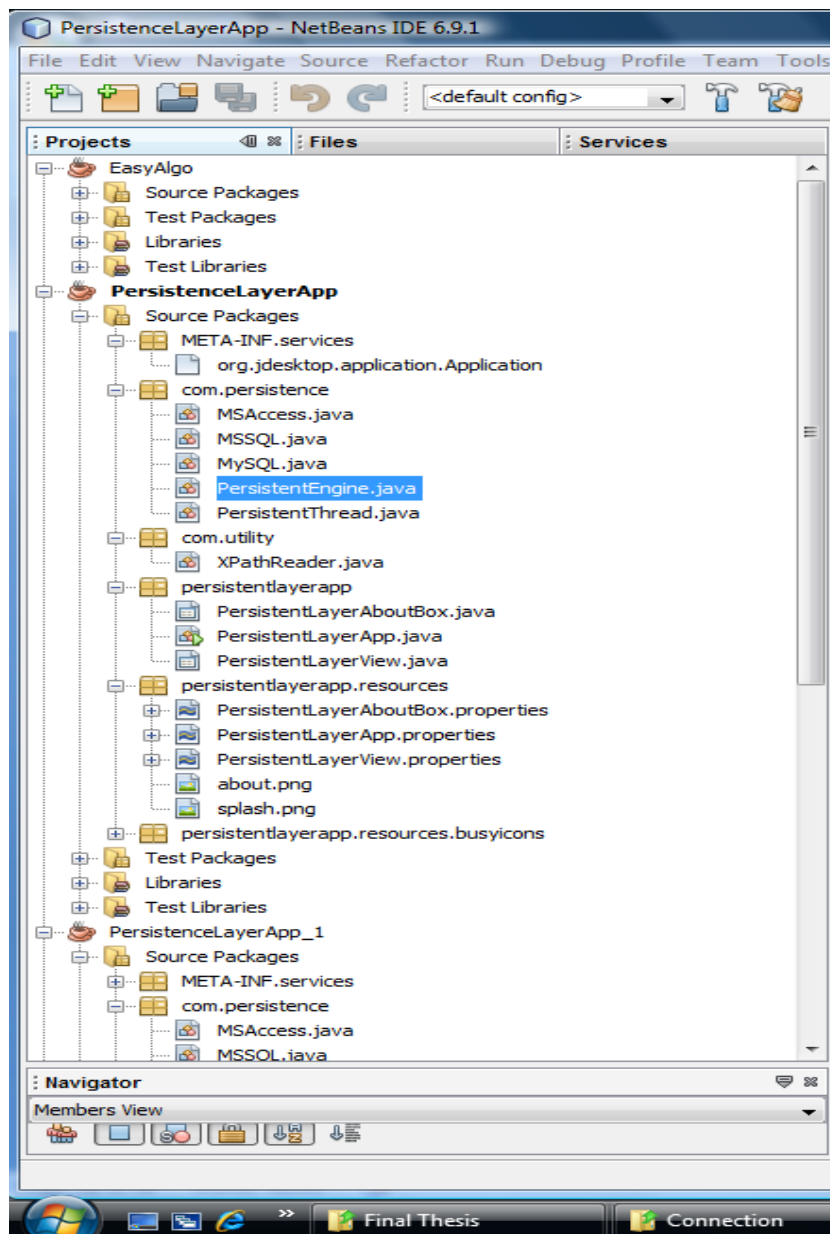
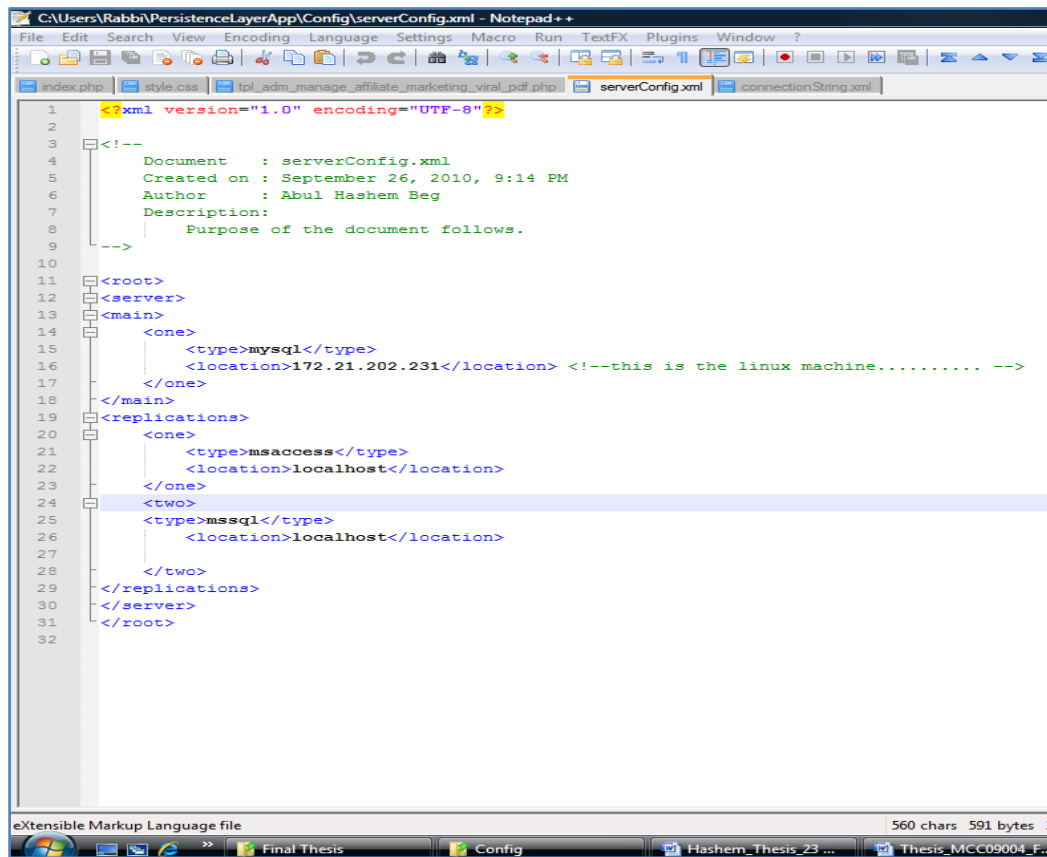


Figure 4.2: Project view of persistence layer APP

Figure 4.2 shows the project and different package view of persistence layer application. This snapshot shown in NetBeans and Java programming. This window appears when the project is loaded on the NetBeans. The file system of NetBeans provides various packages such as “source package,” “test package,” “libraries” and “test libraries.” The source code typically appears on “source package”.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!--
4 Document : serverConfig.xml
5 Created on : September 26, 2010, 9:14 PM
6 Author : Abul Hashem Beg
7 Description:
8 Purpose of the document follows.
9 -->
10
11 <root>
12 <server>
13 <main>
14 <one>
15 <type>mysql</type>
16 <location>172.21.202.231</location> <!--this is the linux machine..... -->
17 </one>
18 </main>
19 <replications>
20 <one>
21 <type>msaccess</type>
22 <location>localhost</location>
23 </one>
24 <two>
25 <type>mssql</type>
26 <location>localhost</location>
27 </two>
28 </replications>
29 </server>
30 </root>
31
32
```

Figure 4.3: Demonstration of Server configuration file

Figure 4.3 shows the Server configuration file. The main purpose of the file is to identify the server's location and the type. The persistence layer read this configuration file to determine which server is main server and which the replication servers are. Besides the hosting address of the main server and replication server is also added within the XML file.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <!--
4      Document   : connectionString.xml
5      Created on  : September 28, 2010, 1:48 AM
6      Author      : Abul Hasem Beg
7      Description :
8          Purpose of the document follows.
9  -->
10
11 <root>
12 <connectionString>
13   <mssql>
14     <connection>jdbc:sqlserver://localhost:1433;databaseName=ump;integratedSecurity=true;</connection>
15     <driver>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver>
16   </mssql>
17   <msaccess>
18     <driver>sun.jdbc.odbc.JdbcOdbcDriver</driver>
19     <url>jdbc:odbc:ump</url>
20   </msaccess>
21   <mysql>
22     <connection>jdbc:mysql://172.21.202.231:3306/ump</connection> <!-- this is called mysql connection string -->
23     <username>root</username> <!-- this is the user name for the linux -->
24     <password>123456</password> <!-- this is the password for the linux machine -->
25   </mysql>
26 </connectionString>
27 </root>
28

```

Figure 4.4: Demonstration of connection string file

Figure 4.4 shows the connection string XML file. This file is responsible to demonstrate the connection definition of server types. The persistence layer needs the address to connect with the main server and replication servers. This file has all the necessary information to get connected to the servers from the persistence layer.

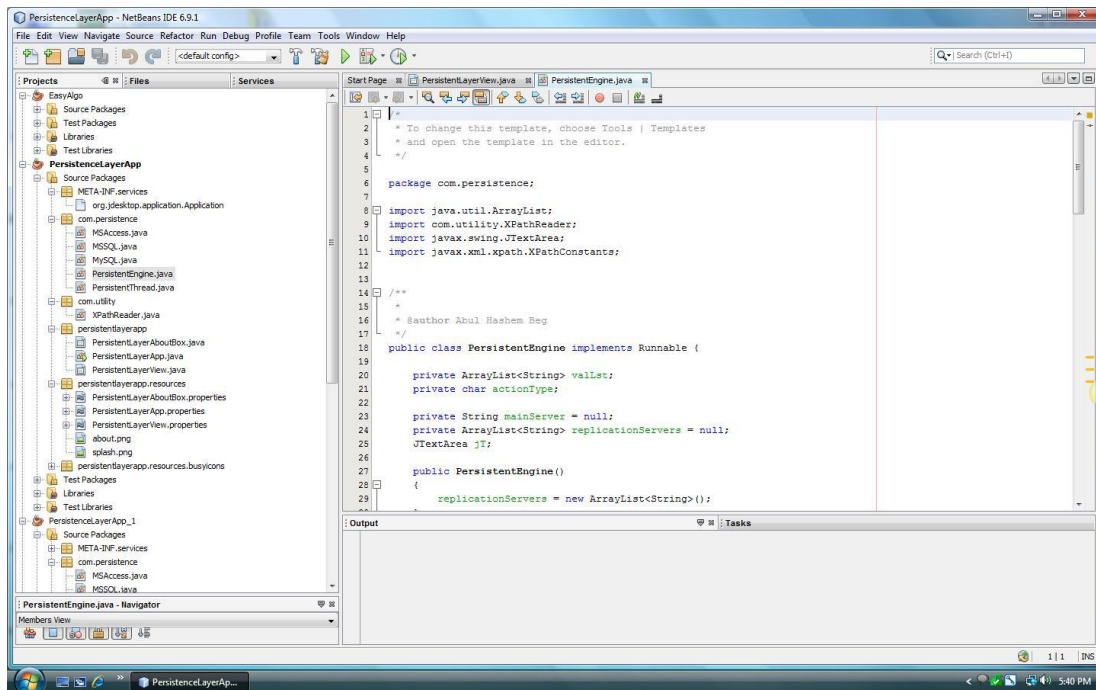


Figure 4.5: Source code view of persistence layer Engine

Figure 4.5 shows the source code of persistence layer. The source code has been written in Net Beans, and the screen shot is shows the part of the source code (persistence layer APP) which has been written in Java. In the persistence layer APP contains the persistence layer engine. When the program has been run, the persistence engine generates the GUI to insert data. After inserting data, the persistence check the config and connection string XML file and then it insert and replicate to the main and replication server.

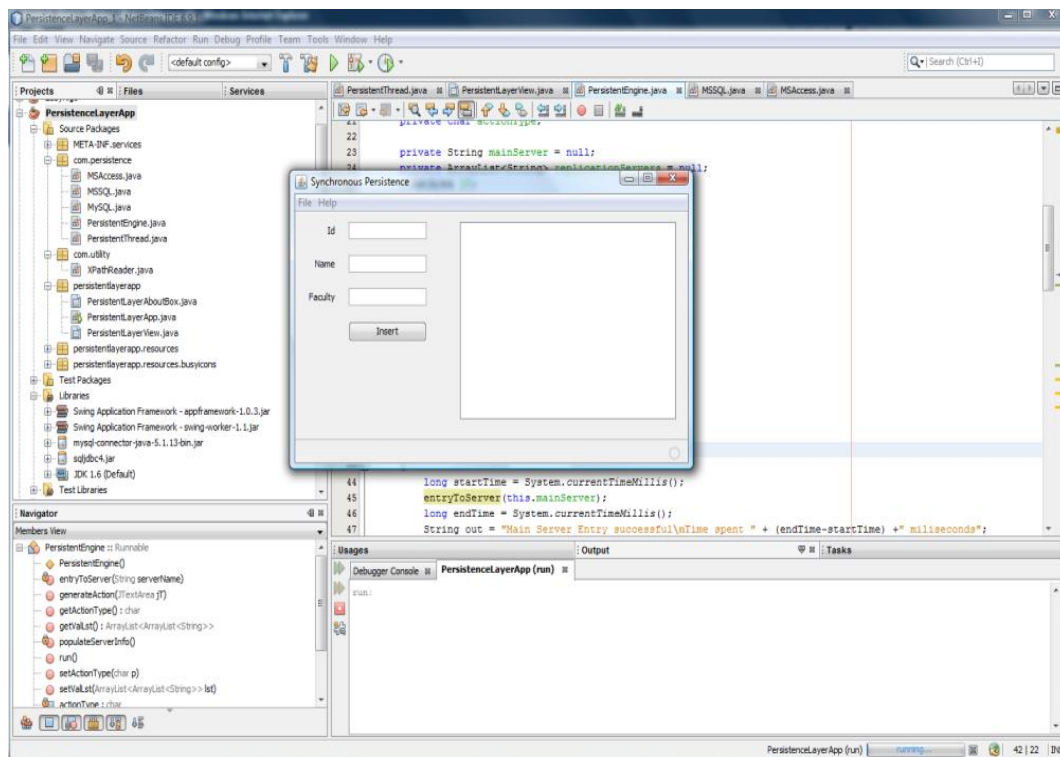


Figure 4.6: The Initial view of the application currently supporting data insertion

Figure 4.6 shows the initial view of the developed tool which currently allows users to insert data through the persistence layer, thus a “Insert” button is shown at the left side of the GUI containing 3 fields. The 3 fields are mapped with a database containing 3 columns. After clicking on the “Insert” button saves the values on the “Id”, “Name”, “Faculty” to the database remains to the main server and replication servers.

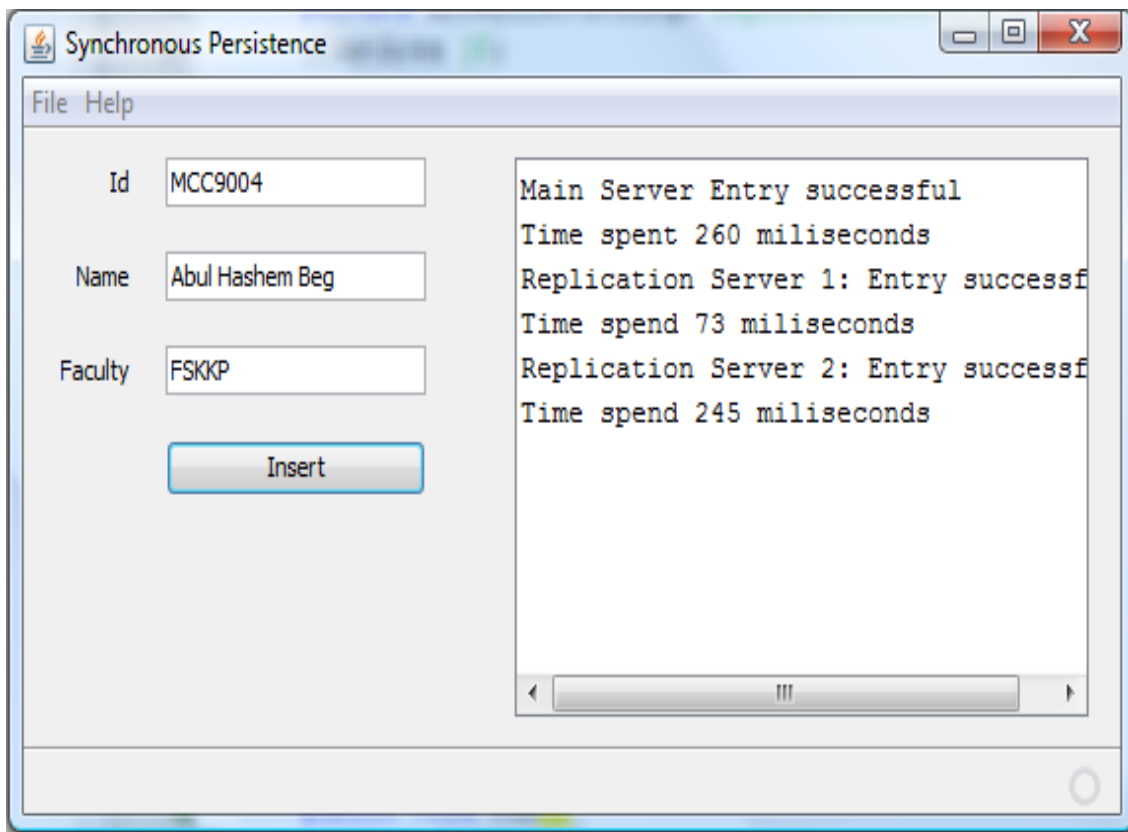


Figure 4.7: The view after data insertion

Figure 4.7 shows the summary of the output on the system when data successfully insert on the main server and on the replication servers. It shows the time for data insertion to different servers.

4.3 HARDWARE AND SOFTWARE COMPONENTS

The implementation of PLSR requires some minimum hardware and software specifications. To demonstrate PLSR system, prototypes across three replication servers as in Figure 4.1 and Figure 4.2 are deployed. Each server or node has been connected to one another through a fast Ethernet switch hub. Theoretically, each of the replication servers and the main server should have to be connected each other logically. The

hardware specifications are shown in Table 4.1 was used in each replication server for implementation.

Table 4.1: Server main components specifications

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

The implementation of the PLSR was carried out by using Java programming language and has been deployed in different OS environment. Table 4.2 shows the system development tools specification for this implementation.

Table 4.2: System development tools specifications

System Development Software	Specifications
Java	SE (Jdk 6.0)
SQLyog	Community Edition 8.53
MySQL Server	Version 5.0.89
Ubuntu	Version 10.0.4
NetBeans IDE	Version 6.9.1
Wine	Version 1.14
Windows Vista	TM Business
SQL Server	Version 2008
MSAccess	Version 2007

4.4 PLSR ENVIRONMENT

From the user's perspective, the functionality offered by PLSR framework for heterogeneous system. Nevertheless, PLSR is tested in different Server under the local area network (LAN) for this implementation. There have been done two experiments. In first experiment SQL Server was the main Server and meanwhile in second experiment MySQL server in Linux OS was the main server.

4.4.1 Experiment 1

In this experiment, the heterogeneous replication has been done with one main server and three replication servers. The entire servers are connected with the persistence layers. SQL server is the main server and connected with the persistence layer with high priority thread. Ms Access is the first replication server. MySQL in Windows OS is the second replication server and MySQL in Linux OS is the third replication server that is connected with the persistence layer with low priority thread as shown in Figure 4.1.

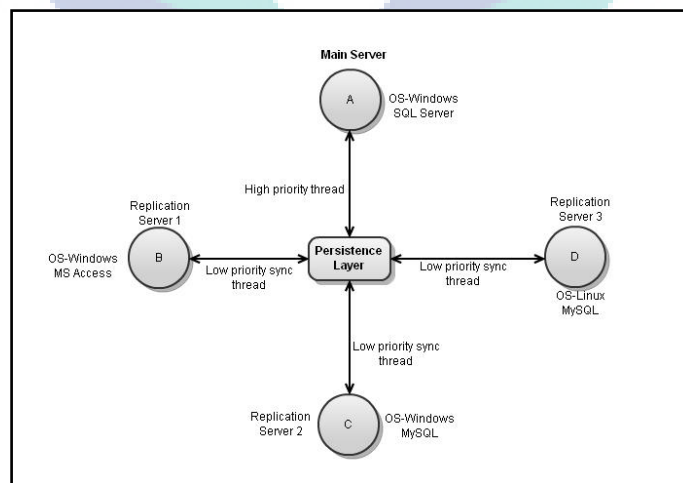


Figure 4.8 Data replication on heterogeneous system with 3 replication server and SQL Server

The host name and IP address for each server depicted in Table 4.3. Server A with IP address 172.21.202.232 is the main server. Server B with IP address 172.21.202.235 is the first replication server. Server C with IP address 172.21.202.231 is the second replication server, and Server D with IP address 172.21.202.230 is the third replication server.

Table 4.3: The local IP address for each server based on SQL Server

Server	Host Name	IP address	Operating System	Software
A	Main Server	172.21.202.232	Windows Vista	SQL Server
B	Replication Server 1	172.21.202.235	Windows Vista	MS Access
C	Replication Server 2	172.21.202.231	Windows Vista	MySQL
D	Replication Server 3	172.21.202.230	Linux (Ubuntu)	MySQL

4.4.2 Experiment 2

In this experiment, heterogeneous replication has been done with one main server and three replication servers. The entire server is connected with the persistence layers. MySQL in Linux OS is the main server and connected with the persistence layer with high priority thread. MS Access is the first replication server. MySQL in Windows OS is the second replication server and SQL Server is the third replication server that is connected with the persistence layer with low priority thread as shown in Figure 4.2.

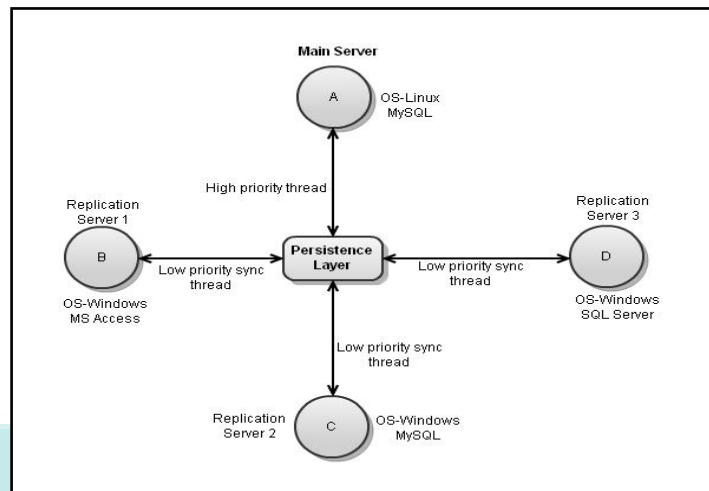


Figure 4.9 Data replication on heterogeneous system with three replication servers and MySQL server

On the other hand, in this experiment, the host name and IP address for each server depicted in Table 4.4. Server A with IP address 172.21.202.230 is the main server. Server B with IP address 172.21.202.235 is the first replication server. Server C with IP address 172.21.202.231 is the second replication server and Server D with IP address 172.21.202.232 is the third replication server.

Table 4.4: The local IP address for each server based on MySQL Server

Server	Host Name	IP address	Operating System	Software
A	Main Server	172.21.202.230	Linux (Ubuntu)	MySQL
B	Replication Server 1	172.21.202.235	Windows Vista	MS Access
C	Replication Server 2	172.21.202.231	Windows Vista	MySQL
D	Replication Server 3	172.21.202.232	Windows Vista	SQL Server

4.5 PLSR IMPLEMENTATION

PLSR has been implemented in the heterogeneous environment. The screen shot and the usability of the implementation to the different replication server and main server has shown and described below:

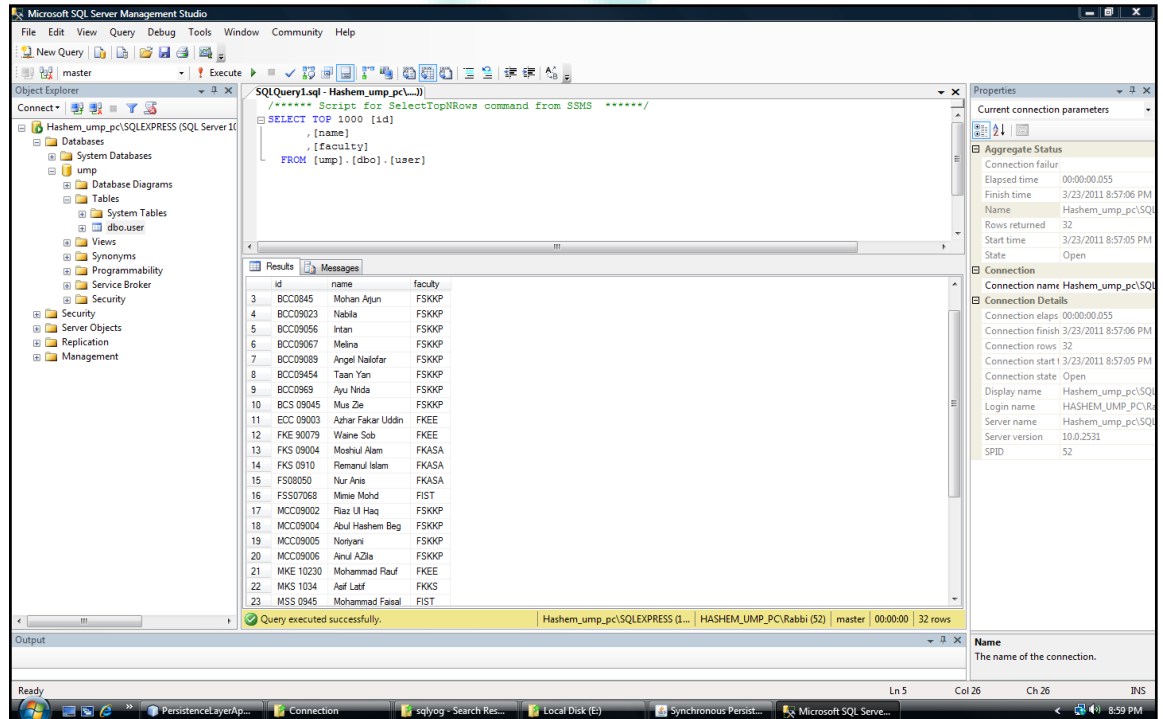
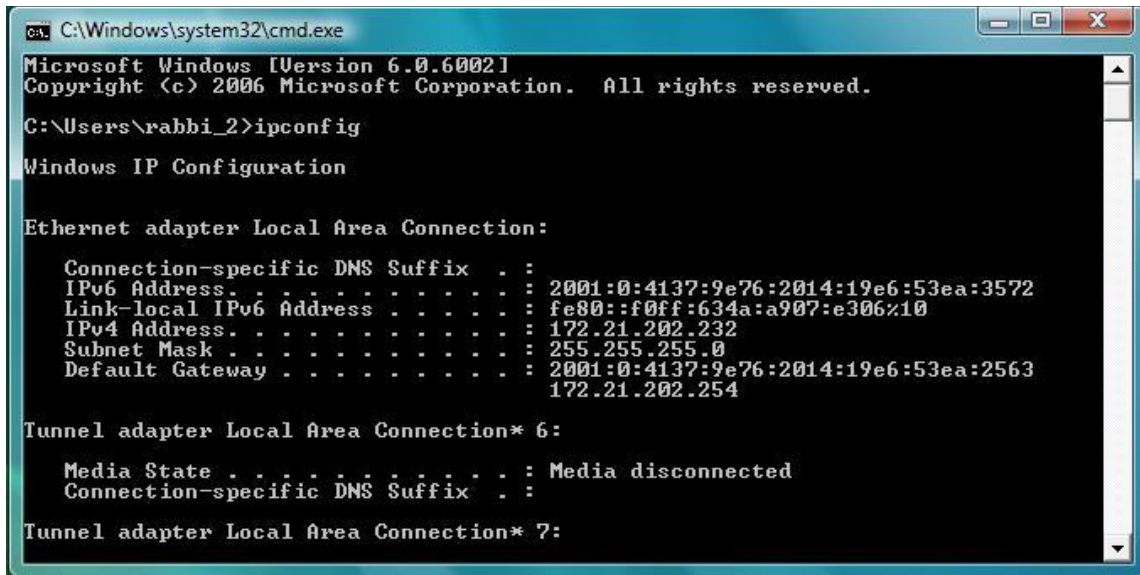


Figure 4.10: The inserted data on the table in the main server (SQL Server)

Figure 4.10 demonstrates the table exists on SQL Server in windows OS, which used as the main server in the developed tool. The data is saved in column “Id”, “Name” and “Faculty” stores the data in SQL Server.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\rabbi_2>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : 
    IPv6 Address . . . . .           : 2001:0:4137:9e76:2014:19e6:53ea:3572
    Link-local IPv6 Address . . . . . : fe80::f0ff:634a:a907:e306%10
    IPv4 Address . . . . .           : 172.21.202.232
    Subnet Mask . . . . .           : 255.255.255.0
    Default Gateway . . . . .       : 2001:0:4137:9e76:2014:19e6:53ea:2563
                                      172.21.202.254

Tunnel adapter Local Area Connection* 6:

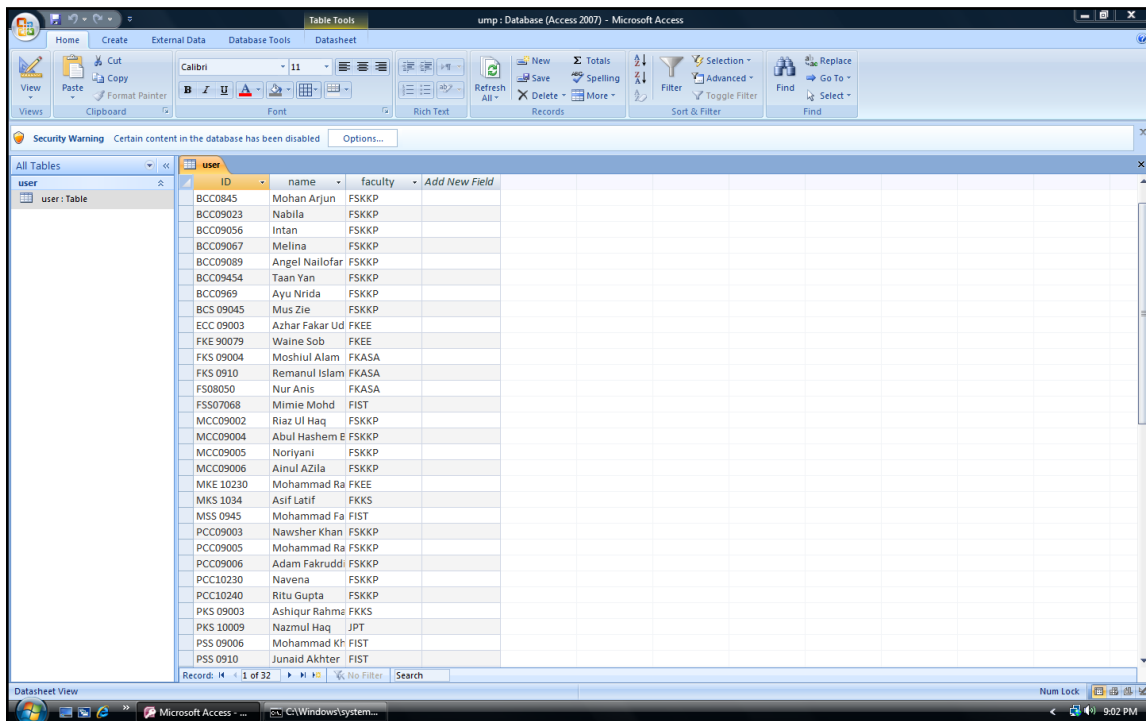
    Media State . . . . .           : Media disconnected
    Connection-specific DNS Suffix  . : 

Tunnel adapter Local Area Connection* 7:
```

Figure 4.11: IP address information of main Server (SQL Server)

Figure 4.11 shows the IP address information of main server (SQL Server). The IP address is 172.21.202.232, Subnet mask 255.255.255.0 and Default gateway 172.21.202.254.

UMP



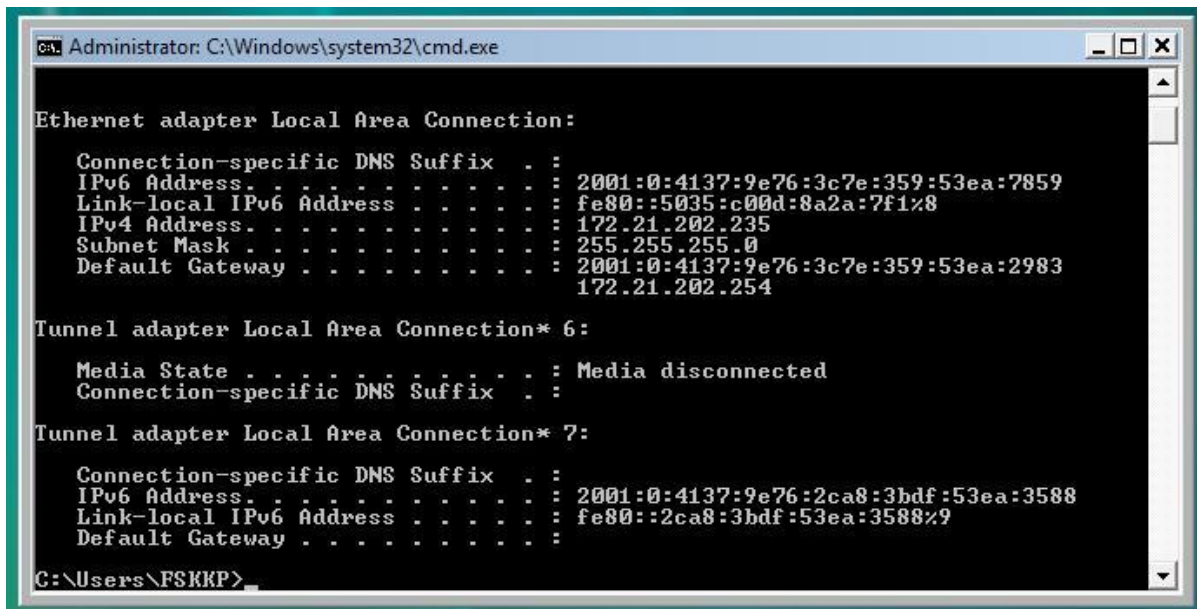
The screenshot shows the Microsoft Access 2007 interface. The title bar reads 'ump : Database (Access 2007) - Microsoft Access'. The ribbon includes 'Home', 'Create', 'External Data', 'Database Tools', and 'Datasheet'. A security warning is visible at the top. The 'All Tables' pane on the left shows a table named 'user'. The main area displays a datasheet view of this table with the following data:

ID	name	faculty
BCC0845	Mohan Arjun	FSKPP
BCC09023	Nabila	FSKPP
BCC09056	Intan	FSKPP
BCC09067	Melina	FSKPP
BCC09089	Angel Nallofar	FSKPP
BCC09454	Taan Yan	FSKPP
BCC0969	Ayu Nrida	FSKPP
BCC09045	Mus Zie	FSKPP
ECC 09003	Azhar Fakar Ud	FKEE
FKE 90079	Waine Sob	FKEE
FKS 09004	Moshiul Alam	FKASA
FKS 0910	Remanul Islam	FKASA
FS08050	Nur Anis	FKASA
FSS07068	Mimie Mohd	FIST
MCC09002	Riaz Ul Haq	FSKPP
MCC09004	Abul Hashem E	FSKPP
MCC09005	Noriyani	FSKPP
MCC09006	Ainul AZila	FSKPP
MKE 10230	Mohammad Ra	FKEE
MKS 1034	Asif Latif	FKKS
MSS 0945	Mohammad Fa	FIST
PCC09003	Nawsher Khan	FSKPP
PCC09005	Mohammad Ra	FSKPP
PCC09006	Adlam Fakruddi	FSKPP
PCC10230	Navena	FSKPP
PCC10240	Ritu Gupta	FSKPP
PKS 09003	Ashiqur Rahma	FKKS
PKS 10009	Nazmul Haq	JPT
PSS 09006	Mohammad KH	FIST
PSS 0910	Jumaid Akhter	FIST

Figure 4.12: The inserted data on the table in a replication server (MS Access)

Figure 4.12 demonstrates the table exists on MS Access, which used as a replication server in the developed tool. The data stores in column “Id”, “Name” and “Faculty” in MS Access.

UMP



```
Administrator: C:\Windows\system32\cmd.exe

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : 
    IPv6 Address. . . . .           : 2001:0:4137:9e76:3c7e:359:53ea:7859
    Link-local IPv6 Address . . . . .: fe80::5035:c00d:8a2a:7f1%8
    IPv4 Address. . . . .           : 172.21.202.235
    Subnet Mask . . . . .           : 255.255.255.0
    Default Gateway . . . . .       : 2001:0:4137:9e76:3c7e:359:53ea:2983
                                      172.21.202.254

Tunnel adapter Local Area Connection* 6:

    Media State . . . . .           : Media disconnected
    Connection-specific DNS Suffix  . : 

Tunnel adapter Local Area Connection* 7:

    Connection-specific DNS Suffix  . : 
    IPv6 Address. . . . .           : 2001:0:4137:9e76:2ca8:3bdf:53ea:3588
    Link-local IPv6 Address . . . . .: fe80::2ca8:3bdf:53ea:3588%9
    Default Gateway . . . . .       : 

C:\Users\FSKKP>
```

Figure 4.13: IP address information of MS Access

Figure 4.13 shows the IP address information of MS Access. The IP address is 172.21.202.235, Subnet mask 255.255.255.0 and Default gateway 172.21.202.254.

UMP

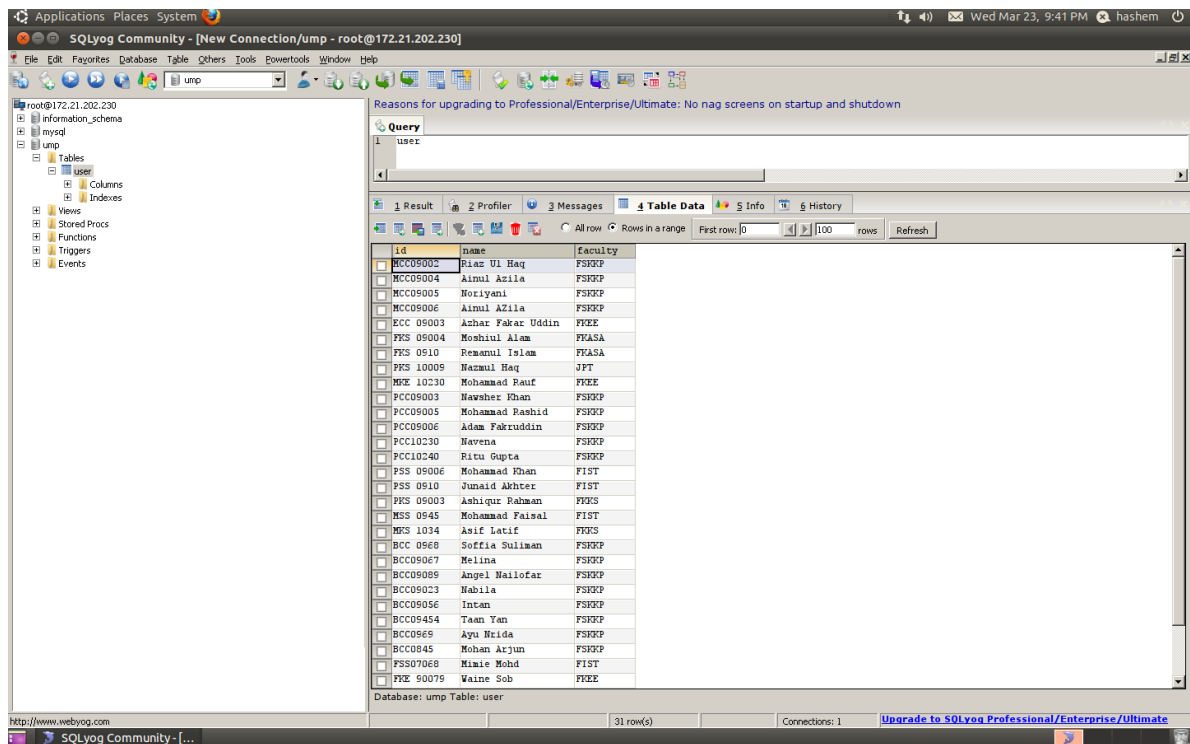


Figure 4.14: The inserted data on the table in a replication server (MySQL in Linux)

Figure 4.14 demonstrates the table exists on MySQL in Linux OS, which used as a replication server in the developed tool. The available column “Id”, “Name” and “Faculty” stores the data in SQL Server.

```

root@hashem-OptiPlex-960: ~
File Edit View Search Terminal Help
eth0  Link encap:Ethernet HWaddr 00:24:e8:3e:65:53
      inet addr:172.21.202.230 Bcast:172.21.202.255 Mask:255.255.255.0
      inet6 addr: fe80::224:e8ff:fe3e:6553/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:143 errors:0 dropped:0 overruns:0 frame:0
      TX packets:38 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:14577 (14.5 KB) TX bytes:9315 (9.3 KB)
      Interrupt:21 Memory:febe0000-fec00000

lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:20 errors:0 dropped:0 overruns:0 frame:0
      TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:1200 (1.2 KB) TX bytes:1200 (1.2 KB)

root@hashem-OptiPlex-960:~# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
172.21.202.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
0.0.0.0 172.21.202.254 0.0.0.0 UG 0 0 0 eth0

```

Figure 4.15: IP address information of MySQL Server (Linux)

Figure 4.15 shows the IP address information of MySQL Server (Linux). The IP address is 172.21.202.230, Subnet mask 255.255.255.0 and Default gateway 172.21.202.254.

UMP

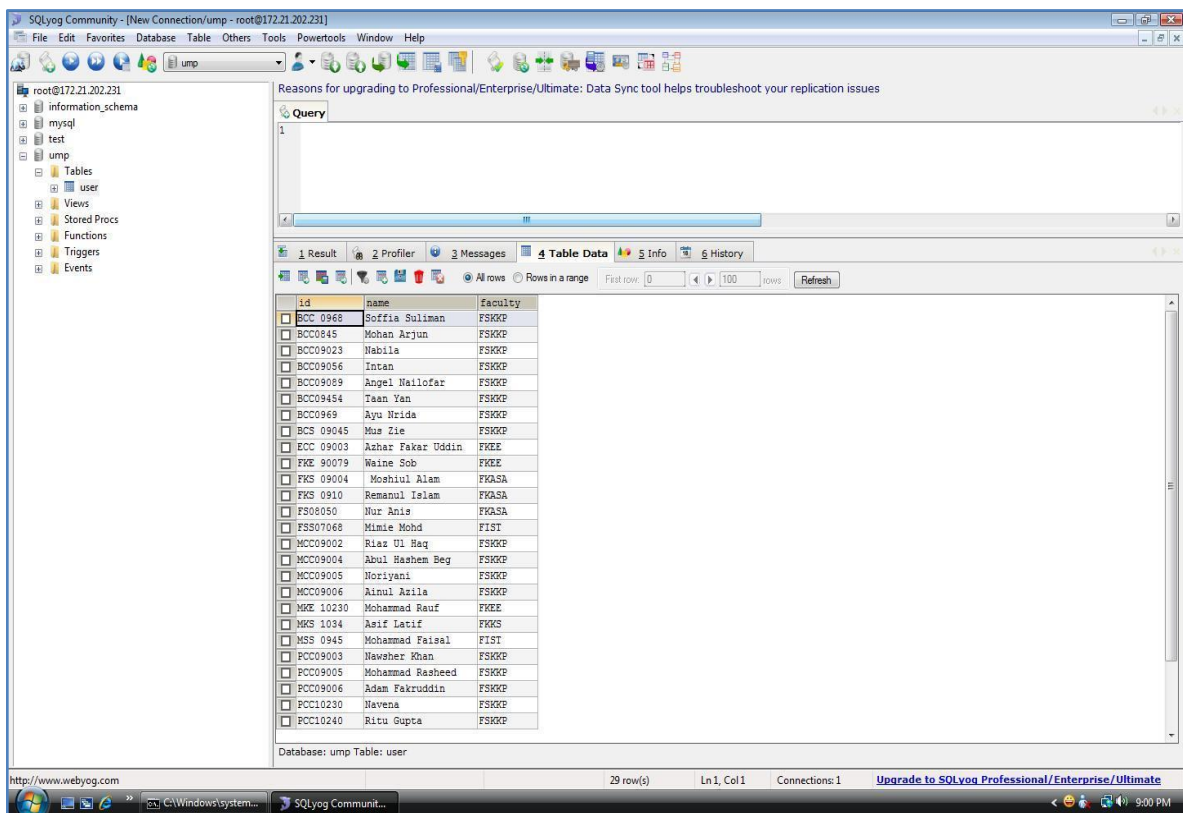
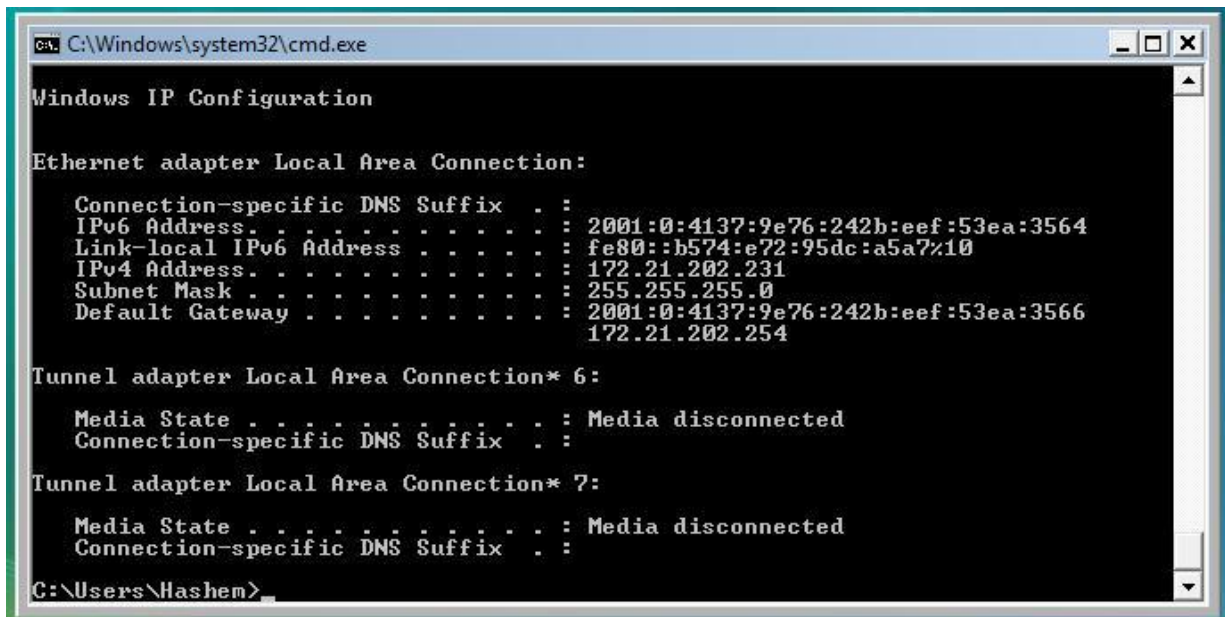


Figure 4.16: The inserted data on the table in a replication server (MySQL in Windows)

Figure 4.16 demonstrates the table exists on MySQL in Windows operating system, which used as a replication server in the developed tool. The column “Id”, “Name” and “Faculty” store the data in MySQL.



```
C:\Windows\system32\cmd.exe
Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : 
    IPv6 Address. . . . .             : 2001:0:4137:9e76:242b:eef:53ea:3564
    Link-local IPv6 Address . . . . . : fe80::b574:e72:95dc:a5a7%10
    IPv4 Address. . . . .              : 172.21.202.231
    Subnet Mask . . . . .             : 255.255.255.0
    Default Gateway . . . . .          : 2001:0:4137:9e76:242b:eef:53ea:3566
                                         172.21.202.254

Tunnel adapter Local Area Connection* 6:

    Media State . . . . .             : Media disconnected
    Connection-specific DNS Suffix  . : 

Tunnel adapter Local Area Connection* 7:

    Media State . . . . .             : Media disconnected
    Connection-specific DNS Suffix  . : 

C:\Users\Hashem>
```

Figure 4.17: IP address information of MySQL (Windows server)

Figure 4.17 shows the IP address information of MySQL Server (Windows). The IP address is 172.21.202.231, Subnet mask 255.255.255.0 and Default gateway 172.21.202.254.

UMP

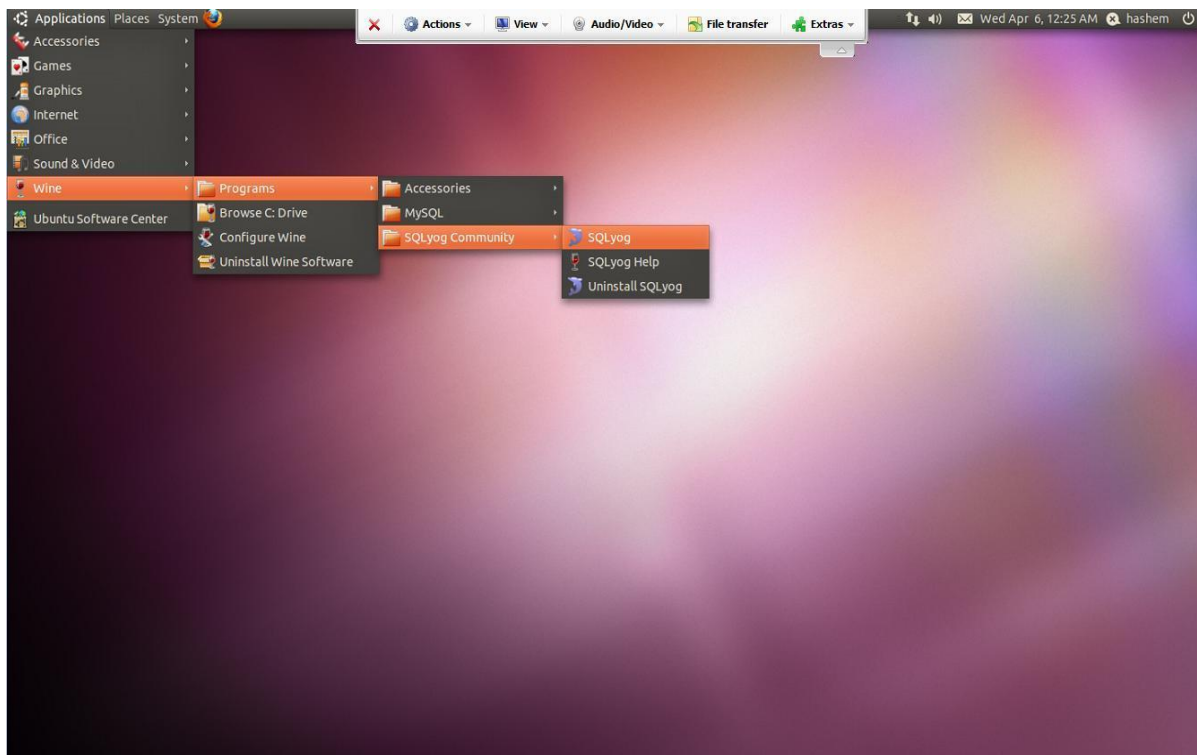


Figure 4.18: Showing permalink SQLyog link under wine in Ubuntu

Figure 4.18 shows the permalink to run SQLyog which is installed under wine. In Ubuntu has been installed wine utility software, which enables to run window *exe* or *msi* files. Thus, after installing SQLyog, the permalink mark as wine software.

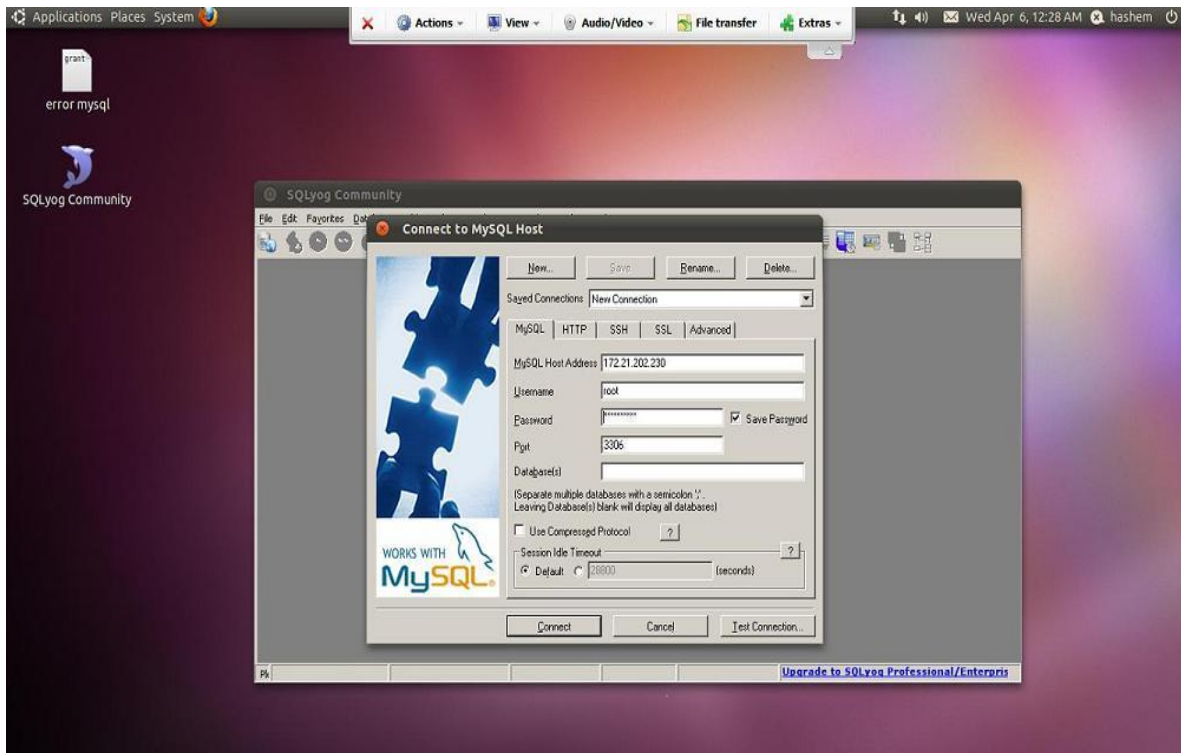


Figure 4.19: Running SQLyog in Ubuntu

Figure 4.19 shows the SQLyog running in Ubuntu OS under wine software. In the SQLyog there are some input parameters. The parameters are MySQL host name, user name, user password and finally the port number. After filling the correct information of the parameters users can be able to connect on the MySQL Server. “Test Connection” is the button to test the input parameter is correct or not.

4.6 RESULT AND DISCUSSION

The proposed heterogeneous synchronous replication (PLSR) has been compared with other replication processes in terms of replication time for transactional insert and synchronization. Table 4.5 shows the comparative time between SQL server and PLSR replication for insertion. In the PLSR, SQL server is the main server and MS Access is the replication server 1. MySQL in Linux OS is the replication server 2 and MySQL in Windows is the replication server 3. The result shows that for 1000 rows of data

insertion, SQL server taken 2 seconds, where PLSR main server, replication server 1, 2 and 3 taken 0.798, 0.364, 5.416 and 6.295 seconds respectively. On the other hand, for 5000 rows of data insertion, SQL server taken 7 seconds, where PLSR main server, replication server 1, 2 and 3 taken 2.411, 1.462, 6.356 and 13.762 seconds respectively. Conversely, for 10000 rows of data insertion, SQL server taken 26 seconds, where, PLSR main server, replication server1, 2 and 3 taken 4.786, 2.933, 7.494 and 22.183 seconds respectively. From the table data, it can be found that the main server (SQL Server) and replication of PLSR take less time than SQL Server for transactional insertion.

Table 4.5: Transactional insert time between SQL Server and PLSR (SQL Server)

No. of Rows	SQL Server	PLSR Main Server (SQL Server)	PLSR Replication Server 1 (MS Access)	PLSR Replication Server 2 (MySQL, Linux OS)	PLSR Replication Server 3 (MySQL, Windows)
100	0	0.381	0.099	5.184	2.388
500	1	0.537	0.216	5.287	3.424
1000	2	0.798	0.364	5.416	6.295
5000	7	2.411	1.462	6.356	13.762
10000	26	4.786	2.933	7.494	22.183

Table 4.6 shows the comparative time between SQL Server and PLSR replication for synchronization. In the PLSR system, MySQL in Linux OS is the main server. MS Access is the replication server 1. SQL Server is the replication server 2 and MySQL in windows is the replication server 3. The result demonstrates that, for 1000 rows of data insertion, SQL server taken 2 second, where PLSR main server, replication server 1, 2 and 3 taken 5.487, 0.372, 0.790 and 6.140 seconds respectively. On the other

hand, for 5000 rows of data insertion, SQL server taken 7 seconds, where PLSR main server, replication server 1, 2 and 3 taken 6.422, 1.507, 2.314 and 12.789 seconds respectively. Conversely, for 10000 rows of data insertion, SQL server taken 26 seconds, where, PLSR main server, replication server 1, 2 and 3 taken 7.526, 2.900, 4.351 and 21.179 seconds respectively. From the table data, it can be found that the main server (MySQL Server) and replication of PLSR take less time than SQL Server for transactional insertion.

Table 4.6: Transactional insert time between PLSR (MySQL Server)

No. of Rows	SQL Server	PLSR Main Server (MySQL Server, Linux)	PLSR Replication Server 1(MS Access)	PLSR Replication Server 2 (SQL Server)	PLSR Replication Server 3 (MySQL, Windows)
100	0	5.252	0.097	0.342	2.260
500	1	5.392	0.220	0.501	3.351
1000	2	5.487	0.372	0.790	6.140
5000	7	6.422	1.507	2.314	12.789
10000	26	7.526	2.900	4.351	21.179

The total replication time for SQL server and PLSR replication are evaluated and shown in Table 4.7 and 4.8. Table 4.7 shows that, the total replication time between SQL server and PLSR replication. In the PLSR system, SQL Server is the main server. MS Access is the replication server1. MySQL in Linux is the replication server 2 and MySQL in windows is the replication server 3. The result shows that, for 1000 rows of data replication, SQL server taken 3 seconds, where PLSR main server, replication server 1, 2 and 3 taken 0.798, 0.364, 5.416 and 6.295 seconds respectively. On the other hand, for 5000 rows of data replication, SQL server taken 9 seconds, where PLSR main

server, replication server 1, 2 and 3 taken 2.411, 1.462, 6.356 and 13.762 seconds respectively. Conversely, for 10000 rows of data insertion, SQL server taken 31 seconds, where, PLSR main server, replication server 1, 2 and 3 taken 4.786, 2.933, 7.494 and 22.183 seconds respectively. From the table data, it can be found that the main server (SQL Server) and replication of PLSR take less time than SQL Server on the replication process.

Table 4.7: Total replication process time between SQL Server and PLSR (SQL Server)

No. of Rows	SQL Server	PLSR Main Server (SQL Server)	PLSR Replication Server 1(MS Access)	PLSR Replication Server 2(MySQL, Linux OS)	PLSR Replication Server 3 (MySQL, Windows)
100	0	0.381	0.099	5.184	2.388
500	2	0.537	0.216	5.287	3.424
1000	3	0.798	0.364	5.416	6.295
5000	9	2.411	1.462	6.356	13.762
10000	31	4.786	2.933	7.494	22.183

Table 4.8 shows that the total replication time between SQL server and PLSR replication. In the PLSR system, MySQL in Linux is the main server. MS Access is the replication server1. SQL Server is the replication server 2 and MySQL in windows is the replication server 3. The result shows that, for 1000 rows of data replication, SQL server taken 3 seconds, where PLSR main server, replication server 1, 2 and 3 taken 5.487, 0.372, 0.790 and 6.140 seconds respectively. On the other hand, for 5000 rows of data replication, SQL server taken 9 seconds, where PLSR main server, replication server 1, 2 and 3 taken 26.422, 1.507, 2.314 and 12.789 seconds respectively. Conversely, for 10000 rows of data insertion, SQL server taken 31 seconds, where, PLSR main server,

replication server 1, 2 and 3 taken 7.526, 2.900, 4.351 and 21.179 seconds respectively. From the table data, it can be found that the main server (MySQL Server) and replication of PLSR take less time than SQL Server on the total replication process.

Table 4.8: Total replication process time between SQL Server and PLSR (MySQL Server)

No. of Rows	SQL Server	PLSR Main Server (MySQL Server, Linux)	PLSR Replication Server 1(MS Access)	PLSR Replication Server 2(SQL Server)	PLSR Replication Server 3 (MySQL, Windows)
100	0	5.252	0.097	0.342	2.260
500	2	5.392	0.220	0.501	3.351
1000	3	5.487	0.372	0.790	6.140
5000	9	6.422	1.507	2.314	12.789
10000	31	7.526	2.900	4.351	21.179

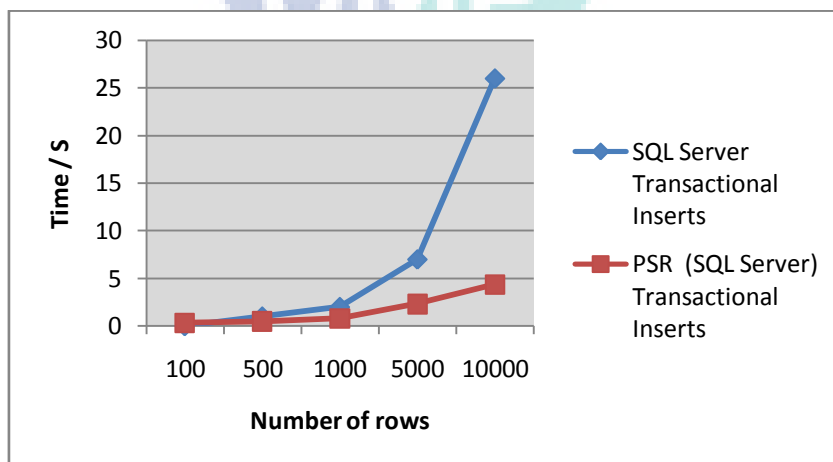


Figure 4.20: Comparative time for transactional insert

Figure 4.20 represent the comparative time between SQL Server and PLSR (SQL Server) transactional time. The graph shows the transactional insertion's time for PLSR (SQL Server) replication is significantly lower than SQL server replication, as the number of data goes higher; SQL Server transactional inserts time getting much higher in comparing to the PLSR replication.

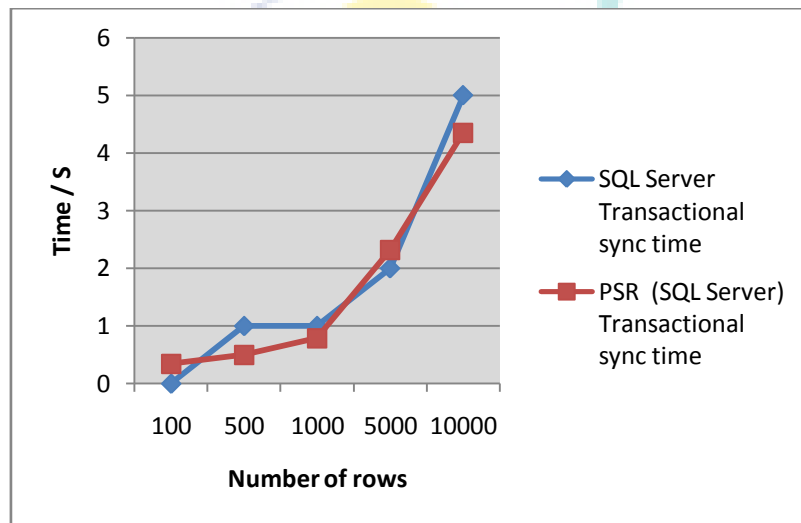


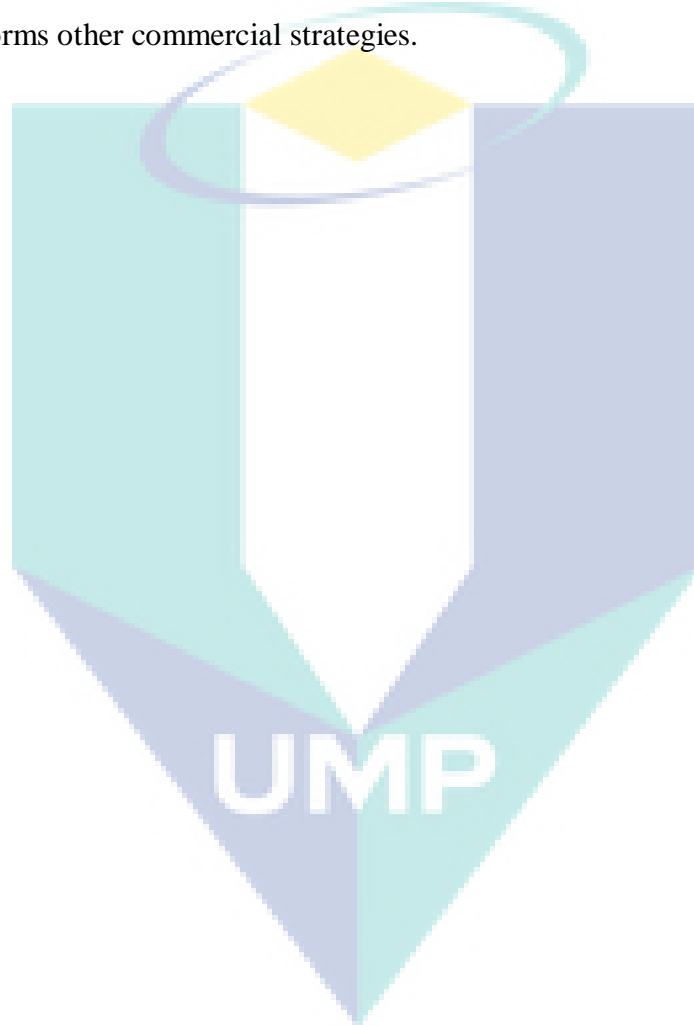
Figure 4.21: Comparative time for synchronization

Figure 4.21 demonstrate the comparative time between SQL Server and PLSR (SQL Server) Synchronization. From the figure, it can be seen that, PLSR (SQL Server) takes less time for replication synchronization compare to SQL Server replication.

The motivation to compare the result of PLSR replication with SQL Server replication is the transactional replications can alter using several trigger, which is similar with the proposed strategy, as the algorithm can perform rollback command from the persistence layer which can alter the result. The result shows that PLSR outstanding performs 83.2% and 2.49% better than SQL server for transactional insert and synchronization in compare to time per seconds. From this above result and the execution point of view, it can be finding that PLSR is highly acceptable.

4.7 CONCLUSION

This chapter represents the detail description and the snap shot of the prototype tool. A detailed performance evaluation has been shown as well. The comparative result shows that the PLSR is highly accepted. A mathematical equation has also been shown. From the result section, it is clearly identified that even in the heterogeneous system, PLSR outperforms other commercial strategies.



CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 INTRODUCTION

This work has been addressed using multi-threading technique to develop a persistence layer for heterogeneous synchronous replication known as PLSR. This chapter summarizes the important findings from the work carried out this research. It also includes some suggestions for future work in each of the areas covered during this research.

5.2 CONCLUSION

At the present time, in the data grid community and clustering system, a lot of work has been focused on providing efficient and safe replication management services through designing of algorithms and systems. A lot of organizations used replication for many purposes.

In this work, the techniques based on the work by many researchers have been discussed (refer Chapter 2). In particular, a new technique PLSR is proposed for managing data replication in the heterogeneous system (refer Chapter 3 and 4). It can provide several advantages like, enterprise application more secure and reliable data transmission. In addition, one of the main goals is to make the database replication easier to handle. Thus, make it vastly configurable and also the whole architecture is a service oriented. It is used latest technology trends and the replication will be from the

persistence layer. Persistence layer is a part of a software engine and it is used the latest customizable fourth generation language like Java. Therefore, a new era can move forward related to networking as well as database programming. Adaptive persistence layer is suitable to be used scalable in the large scale database system. This is because it introduces some packages such as "Replication Sync Service," "Data Queue" and "Intelligent Thread handler." This package provides a structural model to perform the persistence layer more adaptive.

In Chapter 5, discussion and results show the comparative data between the SQL server replication and PLSR replication. It can be seen that PLSR replication is acceptable as the PLSR replication takes less time than SQL Server replication for transactional insert and synchronization. It was observed that PLSR showed outstanding performance and it was 83.2 % and 2.49% than SQL server for transactional insert and synchronization in compare to time (seconds).

This research does provide the following novel contributions: The first major contribution of the research is the framework and algorithm called Persistence Layer Synchronous Replication (PLSR). The second contribution is the tools that can be used in the heterogeneous environment and performs better than SQL Server replication.

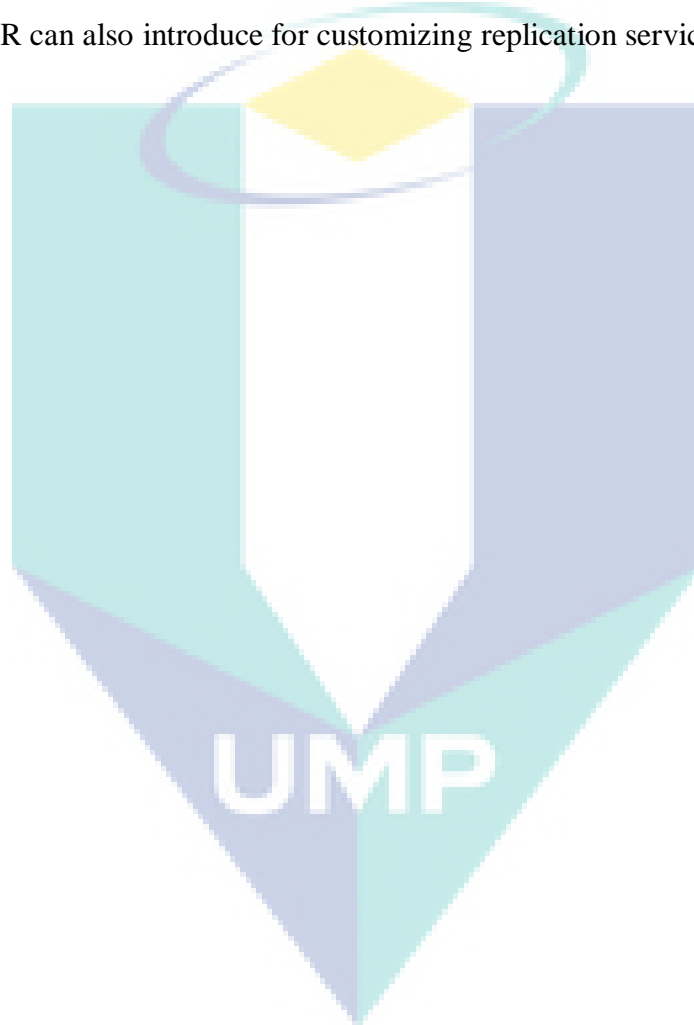
5.3 FUTURE WORK

PLSR can be improved in many different ways. Currently, PLSR does not support complex SQL queries, which can be a significant improvement in commercial application.

Currently, PLSR is support only 3 types of databases. Supporting all other popular databases can be another important improvement of PLSR.

The prototype tool shows only few fields to insert data into different master and replication table. To make this more user friendly, various form of input can be introduce. Therefore, many input type fields can be added in future.

PLSR does not show any charts or reports of data in different variations. In future it can make a significant improvement for commercial usage. A web based version of PLSR can also introduce for customizing replication service.



REFERENCES

- Ahmad, N., Sidek, R.M., Klaib, M.F.J. and Jayan, T.L. 2010. A Novel Algorithm of Managing Replication and Transaction through Read-One- Write-All Monitoring Synchronization Transaction System (ROWA-MSTS). *Second International Conference on Network Applications, Protocols and Services*, pp. 20- 25.
- Ahmad, N., Zin , N.M., Sidek, R.M., Klaib, M.F and Wahab, M. H. 2010. Neighbour Replica Transaction Failure Framework in Data Grid. NDT 2010, Part II, CCIS, *Springer-Verlag Berlin Heidelberg*, **88**: 488-495.
- Alom, B.M. M., Henskens, F. and Hannaford, M. 2009. Querying Semistructured Data with Compression in Distributed Environments. *IEEE confrence on Information Technology, New Generations, ITNG '09*, pp.1546 - 1553.
- Beg, A.H., Noraziah, A., Abdalla A.N., Mohd Zin, N., Sultan, E.I. 2011. Synchronous Replication: Novel Strategy of Software Persistence Layer Supporting Heterogeneous System. *2nd International Conference on Software Engineering and Computer Systems, Communication in Computer and Information Science (CCIS) Series of Springer LNCS*, Pahang, Malaysia (Accepted)
- Bell, D. and Grimson, J. 1992. Distributed Database Systems. *Addison-Wesley*.
- Bellavista, P., Corradi, A. and Magistretti, E. 2005. REDMAN: An optimistic replication middleware for read-only resources in dense MANETs. *Pervasive and Mobile Computing*. 1: 279- 310.
- Bolton, D. 2011. About.com Guide (online)
<http://cplus.about.com/od/glossar1/g/multithreading.htm> (February 12, 2011).
- Bost, Charron, B., Pedone, F. and Schiper, A. 2009. Replication Theory and Practice. *Berlin Heidelberg NewYork, Springer*, ISBN-10 3-642-11293-5 Springer, ch. 2.
- Bsoul, M., Khasawneh, A., Abdallah, E. and Kilani,Y. 2010. Enhanced Fast Spread Replication strategy for Data Grid. *Journal of Network and Computer Applications*. j.jnca.12 (6).
- Boyera, W.F and Hura G.S. 2005. Non-evolutionary algorithm for scheduling dependent tasks in distributed heterogeneous computing environments. *Journal of Parallel Distributed Computing*. **65**: 1035-1046.

- Buertta, M. 1997. Data Replication: Tools and Techniques for Managing Distributed Information. *John Wiley, New York*.
- Caviglione, L. and Cervellera, C. 2007. Design of a peer-to-peer system for optimized content replication. *Computer Communications*. **30** : 3107–3116.
- Cervellera C. and Caviglione, L. 2009. Optimization of a peer-to-peer system for efficient content replication. *European Journal of Operational Research*. **196**: 423–433.
- Chang, R.S., Chang, J.S. and Lin, S.Y. 2007. Job scheduling and data replication on data grids. *Future Generation Computer Systems*. **23**: 846- 860.
- Chen, Y., Sun, X. and Wu, M. 2008. Algorithm-system scalability of heterogeneous computing. *Journal of Parallel Distributed Computing*. **68**: 1403-1412.
- Chidambaram, J., Rao, P. A.N., Aneesh, C. S., Prabhu, C. S. R., Wankar, R. and Agarwal, A. 2008. A Methodology for High Availability of Data for Business Continuity Planning / Disaster Recovery in a Grid using Replication in a Distributed Database. *TENCON, IEEE Region 10 Conference*, pp.1-6.
- Chi, C.H., Su, M., Liu, L. and Wang, H.G. 2006. An Active Peer-to-Peer System for Heterogeneous Service Provisioning. *IEEE International Conference on Information Reuse and Integration*, pp. 17- 22.
- Ciglan, M. and Hluchy, L. 2007. Content synchronization in replicated grid database resources. *Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*, pp. 379-386
- Connolly, T.M. and Begg, C.E. 2005. Database systems: A Practical Approach to Design, Implementation and Management. *4th Edition, Addison- Wesley*.
- Deris, M.M., Mamat, A., Seng, P.C. and Saman, M.Y. 2001. Three Dimensional Grid Structure of Efficient Access of Replicated Data. *International Journal of Interconnection Network, World Scientific*. **2**(3): 317-329.
- Deris, M.M., Nathrah, B., Suzuri, M. H. and Osman A.M.T., 2004. Improving Data Availability Using Hybrid Replication Technique in Peer-to-Peer Environments. *Journal of Interconnection Networks*. **5**(3): 299-312.
- Dewald, B. and Kline, K. InformIT. 2002. SQL Server: Transaction and Locking Architecture (online)
<http://www.informit.com/articles/article.aspx?p=26657> (January 30, 2011).

- Di, R.H., Wang, T., Liang, Y. and Su, L. 2010. The Analysis and Implementation of Partition Replication-Based Distributed Cache System. *IEEE International Conference on High Performance Computing and Communications (HPCC)*, pp. 719 – 724.
- Dogan, A. 2009. A study on performance of dynamic file replication algorithms for real-time file access in Data Grids. *Future Generation Computer Systems*. **25**: 829-839.
- Elghirani, A., Zomaya, A.Y. and Subrata, R. 2007. An Intelligent Replication Framework for Data Grids. *Computer Systems and Applications, AICCSA '07, IEEE/ACS International Conference*, pp. 351- 358.
- Filip, I., Vasar, C. and Robu, R. 2009. Considerations about an Oracle Database Multi-Master Replication. *IEEE/5th International Symposium on Applied Computational Intelligence and Informatics. SACI '09*, pp. 147 – 152.
- Essmann, B., Hampel, T. and Keil-Slawik, R. 2007. Challenges towards a Distributed Persistence Layer for Next Generation CSCW Applications. *IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops '07*, pp. 199-203.
- Freeman, R.G. 2005. *Portable DBA Oracle*, McGraw-Hill/Osborne, California, USA.
- Gao, T. and Liu, F. 2007. A Dynamic Data Replication Technology in Educational Resource Grid. *Information Technologies and Applications in Education, ISITAE '07. First IEEE International Symposium*, pp. 287- 291.
- Gutzait, M. Simplify SQL Server replication. 2007 (online)
<http://searchsqlserver.techtarget.com/tip/Simplify-SQL-Server-replication> (June 23, 2010)
- Gu, L., Budd, L., Caycl, A., Hendricks, C., Purnell, M. and Rigdon, C. 2002. Practical Guide to DB2 UDB Data Replication V8. *Durham, NC, USA, IBM*, ch.1.
- Gu, X., Lin, W. and Veeravalli, B. 2006. Practically Realizable Efficient Data Allocation and Replication Strategies for Distributed Databases with Buffer Constraints. *IEEE transactions on parallel and distributed systems*, **17**(9): 1001-1013.
- Hao, W., Fu, J., Yen, I.L. and Xia, Z. 2008. A Novel PSO-MP Approach for Database Replications at Edge Servers. *Tools with Artificial Intelligence, ICTAI '08. 20th IEEE International Conference*, pp. 291- 298.

- Hao, Y., Xing-chun, D. and Guo-quan, J. 2008. Research on Data Synchronization in Oracle Distributed System. *International Seminar on Future Information Technology and Management Engineering, FITME '08*. pp. 540 -542.
- Hitachi data system. 2007. Synchronous Data Replication (online). <http://www.hds.co.uk/assets/pdf/sb-synchronous-data-replication.pdf> (18 February, 2010).
- Ho, K.M., Poon, W.F., and Lo, K.T. 2007. Performance Study of Large-Scale Video Streaming Services in Highly Heterogeneous Environment. *IEEE Transactions on Broadcasting*. **53** (4).
- Horri, A., Sepahv, R. and Dastghaibyard, Gh. 2008. A hierarchical scheduling and replication strategy. *International Journal of Computer Science and Network Security*. **8**(8).
- Ibej, U.C., Slivnik B. and Robic, B. 2005. The complexity of static data replication in data grids. *Parallel Computing*. **31**: 900- 912.
- Ibison, P. 2010. Basic Comparison of Replication Times between Merge and Transactional Replication. (online) <http://www.replicationanswers.com/ReplicationTimesArticle.asp> (November 28, 2010).
- Inigo A. J.E., Rodriguez, J.J.R., Mendivil, G.J.R., Garitagoitia, J.R., Briz L.I., F.D. and Escoi, M. 2011. A formal characterization of SI-based ROWA replication protocols. *Data & Knowledge Engineering*. **70**: 21- 34.
- Jianfeng, Z., Leihua, Q., Dong, Z., Jinli, Z. 2008. A Duplicate-Aware Data Replication. *Frontier of Computer Science and Technology, FCST '08. Japan-China Joint Workshop*, pp. 112- 117.
- Khan, S.U. and Ahmad I. 2008. Comparison and analysis of ten static heuristics-based Internet data replication techniques. *Journal of Parallel Distributed Computing*. **68**:113-136.
- Khanli, L.M., Isazadeh, A. and Shishavan, T.N. 2011. PHFS: A dynamic replication method, to decrease access latency in the multi-tier data grid. *Future Generation Computer Systems*. **27**: 233- 244.
- Kim, J., Sim, S., and Park, S. 2007. A Cluster File System for High Data Availability using Locality-Aware Partial Replication. *Seventh IEEE International Conference on Computer and Information Technology*, pp. 345 – 350.

- Knezevic, P., Wombacher, A. and Risse, T. 2006. DHT-based self-adapting replication protocol for achieving high data availability. In: *Proceedings of The International Conference on Signal-image Technology and Internet based Systems, SITIS*.
- Kosar, T. and Livny, M. 2005. A framework for reliable and efficient data placement in distributed computing systems. *Journal of Parallel Distributed Computing*. 65:1146 – 1157
- Lei, M., Vrbsky, S.V. and Hong, X. 2008. An on-line replication strategy to increase availability in Data Grids. *Future Generation Computer Systems*. **24**: 85- 98.
- Li, Z. and Shen, H. 2009. A mobility and congestion resilient data management system for distributed mobile networks. *Mobile Adhoc and Sensor Systems, MASS '09. IEEE 6th International Conference on Macau*, pp. 60-69.
- Lin, Y. 2007. Practical and consistent database replication. *Mc Gill University Montreal, Quebec*, ch. 1-2.
- Litke, A., Skoutas, D., Tserpes, K. and Varvarigou, T. 2007. Efficient task replication and management for adaptive fault tolerance in Mobile Grid environments. *Future Generation Computer Systems*. **23**:163–178.
- Lou, Y.S., Wang, Z.J., Huang, L. and Yue, L. 2009. The Study of a Reflected Persistence Data Layer Framework. *WRI World Congress on Software Engineering, WCSE'09*, pp. 291-294.
- Ma, D., Zhang, W. and Li, Q. 2004. Dynamic Scheduling Algorithm for Parallel Real-time Jobs in Heterogeneous System. *The Fourth International Conference on Computer and Information Technology, CIT '04*, pp. 462- 466.
- Mavromoustakis, C.X. and Karatza H.D. 2008. Under storage constraints of epidemic backup node selection using HyMIS architecture for data replication in mobile peer-to-peer networks. *The Journal of Systems and Software*. **81**: 100–112.
- Nam, C.D., Youn, C., Jeong, S., Shim, E., Lee, E. and Park, E. 2004. An efficient replication scheme for data grids. In: *Proceedings 12th IEEE International Conference on Networks, ICON*, pp. 392- 396.
- Noraziah, A., Deris M. M., Saman, M.Y. M., Norhayati, R., Rabiei, M. and Shuhadah, W.N.W. 2009. Managing Transaction on Grid-Neighbour Replication in Distributed System. *International Journal of Computer Mathematics, Taylor and Francis*. **86**(9):1624-1633

<http://www.openehr.org/208-OE.html?branch=1&language=1> (February 18, 2010)

- Palomar, E., Tapiador J.M.E., Castro, J.C. H. and Ribagorda, A. 2008. Secure content access and replication in pure P2P networks. *Computer Communications*. **31**:266-279
- Perez, J.M., Carballeira, F.G., Carretero, J., Calderon, A. and Fernandez, J. 2010. Branch replication scheme: A new model for data replication in large scale data grids. *Future Generation Computer Systems*. **26**: 12- 20.
- Poddar, S. 2003. SQL Server Transactions and Error Handling (Online)
<http://www.codeproject.com/KB/database/sqlservertransactions.aspx> (January 30, 2011).
- Post, G.V. 2006. Database Management system: Design and Building Business Applications. *McGraw-Hill/Irwin, New York*.
- Pucciani, G., Domenici, A., Donno, F. and Stockinger, H. 2010. A performance study on the synchronisation of heterogeneous Grid databases using CONStanza. *Future Generation Computer Systems*. **26**: 820- 834.
- Qiao, H., Ju, R., Li, G., and Huang, K. 2006. A New Persistence Framework for Parallel and Distributed Simulation. *International on Multi-Symposiums on Computer and Computational Sciences, IMSCCS '06*, pp. 344- 348.
- Qin X. and Jiang, H. 2006. A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems. *Parallel Computing*. **32**:331-356.
- Rahman, R., Barker, K. and Alhaji, R. 2006. Replica placement design with static optimality and dynamic maintainability. *In: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid, CCGRID*, pp. 16-19.
- Rahman R.M., Alhaji, R. and Barker K. 2008. Replica selection strategies in data grid. *Journal of Parallel Distributed Computing*. **68**: 1561- 1574.
- Sashi, K. and Thanamani, A. S. 2011. Dynamic replication in a data grid using a Modified BHR Region Based Algorithm. *Future Generation Computer Systems*. **27**: 202- 210.
- Sato, H., Matsuoka, S., and Endo, T. 2009. File Clustering Based Replication Algorithm in a Grid Environment. *9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 204 – 211.

- SearchSQLserver.com. Definition SQL Server. 2006 (online)
<http://searchsqlserver.techtarget.com/definition/SQL-Server> (February 25, 2011)
- Shena H. and Zhu, Y. 2009. A proactive low-overhead file replication scheme for structured P2P content delivery networks. *Journal of Parallel and Distributed Computing*. **69**: 429- 440.
- Stockinger, H. 2001. Distributed Database Management Systems and the Data Grid. *IEEE-NASA Symposium*, pp.1-12.
- Sriram, I. and Cliff D. 2010. Effects of Component-Subscription Network Topology on Large-Scale Data Centre Performance Scaling. *Engineering of Complex Computer Systems (ICECCS), 15th IEEE International Conference*, pp. 72- 81.
- Tang, M., Leel, B.S., Yeo, C.K. and Tang, X. 2005. Dynamic replication algorithms for the multi-tier Data Grid. *Future Generation Computer Systems*. **21**: 775- 790.
- Tang, M., Lee, B.S., Tang, X. and Yeo, C.K. 2006. The impact of data replication on job scheduling performance in the Data Grid. *Future Generation Computer Systems*. **22**: 254- 268.
- Tanga, X., Kenli L., Renfa, L., Veeravalli, B. 2010. Reliability-aware scheduling strategy for heterogeneous distributed computing systems. *Journal of Parallel Distributed Computing*. **70**:941-952.
- Tim Ford. 2010. MSSQL Tips (online)
<http://www.mssqltips.com/tip.asp?tip=2183> (January 25, 2011).
- Tong, X. and Shu, W. 2009. An Efficient Dynamic Load Balancing Scheme for Heterogenous Processing System. *IEEE Conference on Computational Intelligence and Natural Computing*, pp. 319- 322.
- Urbano, R. 2003. Oracle Database Advanced Replication. *Part No. B10732-01, Oracle Corporation* ,Ch. 1, Retrieved February 8, 2010, from http://www.databasebooks.us/oracle_0003.php
- Wang, C., Yang, C. and Chiang, M. 2007. A fair replica placement for parallel download on cluster grid, in: Network-Based Information Systems. *In: Lecture Notes in Computer Science*, **4658**: 268-277.
- Wang, C.H., Kim, H., Wu, Y. and Ying, V. 2007. Compiler-Managed Software-based Redundant Multi-Threading for Transient Fault Detection. *International Symposium on Code Generation and Optimization, CGO '07*. pp. 244- 258.

- Wang, Y. and Li, S. 2006. Research and performance evaluation of data replication technology in distributed storage systems. *Computers & Mathematics with Applications*. **51**:1625-1632.
- Wang, C.M., Chen, H.M., Lee, G.C., Wang, S.T., and Hong, S.F. 2005. A Tree-Structured Persistence Server for Data Management of Collaborative Applications. *IEEE International Conference on Advanced Information Networking and Applications, AINA*, pp. 503 – 506
- Wei, L., Ping, W.X., Qi, Z. and Nong, Z. 2009. Improving Throughout of Continuous k-Nearest Neighbor Queries with Multi-threaded Techniques. *IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS*. pp. 438- 442.
- Wu, H., Tong, H., Yu, C., Zhao, X. and Liu, Y. 2007. Real-time Compilation System for Input-Output Table Based on Distributed Database and Remote Data Replication Technology. *IEEE conference on Convergence Information Technology*, pp. 1988- 1992.
- Wujuan, L. and Veeravalli, B. Design and analysis of an adaptive object replication algorithm in distributed network systems. *Computer Communications*. **31**: 2005-2015.
- Wu, Q., Hu Y. and Wang, Y. 2010. Research on Data Persistence Layer Based on Hibernate Framework. *2nd International Workshop on Intelligent Systems and Applications (ISA)*, pp. 1 – 4.
- Youn, H.Y., Krishnamsetty, B., Lee, D., Lee, B.K., Choi, J.S., Kim, H.G., Park, C.W., and Su, L.H. 2002. An efficient hybrid replication protocol for highly available distributed system. In: *Proceedings of the Communications and Computer Networks, CCN, Cambridge, USA*. 381:078.
- Zhang, Z., Wang, X., Qi, G. and Yao, W. 2010. Study on the Method of Building Data Persistence Layer Based on the Data Dictionary. *IEEE International Conference on Computer and Communication Technologies in Agriculture Engineering (CCTAE)*, pp. 320 - 323
- Zhoua, X. and Xu, C.Z. 2007. Efficient algorithms of video replication and placement on a cluster of streaming servers. *Computer Applications*. **30**:515–540.
- Zhou, Z. and Chen, Z. 2010. Performance Evaluation of Transparent Persistence Layer in Java Applications. *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 21 - 26

APPENDIX

```
public class PersistentEngine implements Runnable {

    private ArrayList<String> valLst;
    private char actionType;

    private String mainServer = null;
    private ArrayList<String> replicationServers = null;
    JTextArea jT;

    public PersistentEngine()
    {
        replicationServers = new ArrayList<String>();
    }

    public void generateAction(JTextArea jT)
    {
        populateServerInfo();
        this.jT = jT;
        Thread thisThread = new Thread(this);
        thisThread.setPriority(Thread.MAX_PRIORITY);
        thisThread.start();
    }

    private void populateServerInfo()
    {
        String path = "\\Config\\serverConfig.xml";

        XPathReader xPath = new XPathReader(path);
        String expression = "/root/server/main/one/type";
        this.mainServer = xPath.read(expression, XPathConstants.STRING).toString();

        expression = "/root/server/replications/one/type";
        this.replicationServers.add(xPath.read(expression,
        XPathConstants.STRING).toString());

        expression = "/root/server/replications/two/type";
        this.replicationServers.add(xPath.read(expression,
        XPathConstants.STRING).toString());
    }
}
```

```

}

private void entryToServer(String serverName)
{
    if(serverName.equals("mysql"))
    {
        new MySQL(valLst.get(0), valLst.get(1), valLst.get(2));
    }
    else if(serverName.equals("msaccess"))
    {
        new MSAccess(valLst.get(0), valLst.get(1), valLst.get(2));
    }
    else if(serverName.equals("mssql"))
    {
        new MSSQL(valLst.get(0), valLst.get(1), valLst.get(2));
    }
}

public PersistentThread(ArrayList<String> lst, ArrayList<String> repServerLst,
JTextArea jT)
{
    thisThread = new Thread(this);
    this.valLst = lst;
    this.repServer = repServerLst;
    this.jT = jT;

    thisThread.setPriority(Thread.NORM_PRIORITY);
    thisThread.start();
}

public MSAccess(String id, String name, String faculty)
{
    String path = "\\Connection\\connectionString.xml";

    XPathReader xPath = new XPathReader(path);
    String expression = "/root/connectionString/msaccess/driver";
    this.DRIVER = xPath.read(expression, XPathConstants.STRING).toString();

    expression = "/root/connectionString/msaccess/url";
    this.URL = xPath.read(expression, XPathConstants.STRING).toString();

    InsertData(id, name, faculty);
}

```

```

public void InsertData(String id, String name, String faculty) {
try {
    Class.forName(DRIVER);
    Connection connection = null;
    connection = DriverManager.getConnection(URL);

    Statement stm = connection.createStatement();
    String query = "Insert into user values (" + id + ", " + name + ", " + faculty + ")";
    stm.executeUpdate(query);

}
catch (Exception e) {
    e.printStackTrace();
}
}

public class MSSQL {

private String connectionUrl = null; //"jdbc:sqlserver://localhost:1433;" +
// "databaseName=ump;integratedSecurity=true;";
private String DRIVER = null; //"com.microsoft.sqlserver.jdbc.SQLServerDriver";

public MSSQL(String id, String name, String faculty)
{
    String path = "\\Connection\\connectionString.xml";

    XPathReader xPath = new XPathReader(path);
    String expression = "/root/connectionString/mssql/connection";
    this.connectionUrl = xPath.read(expression, XPathConstants.STRING).toString();

    expression = "/root/connectionString/mssql/driver";
    this.DRIVER = xPath.read(expression, XPathConstants.STRING).toString();

    InsertData(id, name, faculty);
}

public void InsertData(String id, String name, String faculty)
{
    Connection con = null;
    Statement stmt = null;
    try
    {
        // Establish the connection.

```

```

Class.forName(DRIVER);
con = DriverManager.getConnection(connectionUrl);

// Create and execute an SQL statement that returns some data.
stmt = con.createStatement();
String query = "INSERT INTO [ump].[dbo].[user] ([id],[name],[faculty])
VALUES ( '"+id+"','"+name+"','"+faculty+"')";
stmt.executeUpdate(query);

stmt.close();
con.close();
}
catch (Exception e)
{
    e.printStackTrace();
}
}

public MySQL(String id, String name, String faculty)
{
    String path = "\\Connection\\connectionString.xml";
    XPathReader xPath = new XPathReader(path);
    String expression = "/root/connectionString/mysql/connection";
    this.connectionString = xPath.read(expression,
XPathConstants.STRING).toString();

    expression = "/root/connectionString/mysql/username";
    this.userName = xPath.read(expression, XPathConstants.STRING).toString();

    expression = "/root/connectionString/mysql/password";
    this.password = xPath.read(expression, XPathConstants.STRING).toString();

    InsertData(id, name, faculty);
}

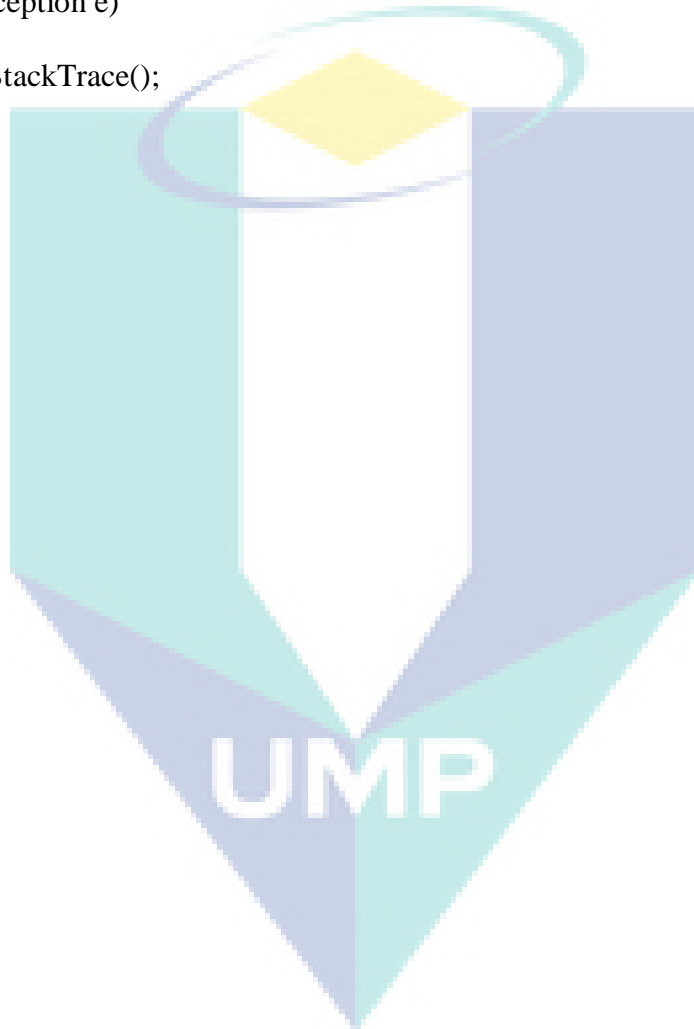
public void InsertData(String id, String name, String faculty) {
try
{
    Class.forName("com.mysql.jdbc.Driver").newInstance();
    Connection conn = DriverManager.getConnection(connectionString, userName,
password);

    System.out.println("connected");

    Statement st = conn.createStatement();

```

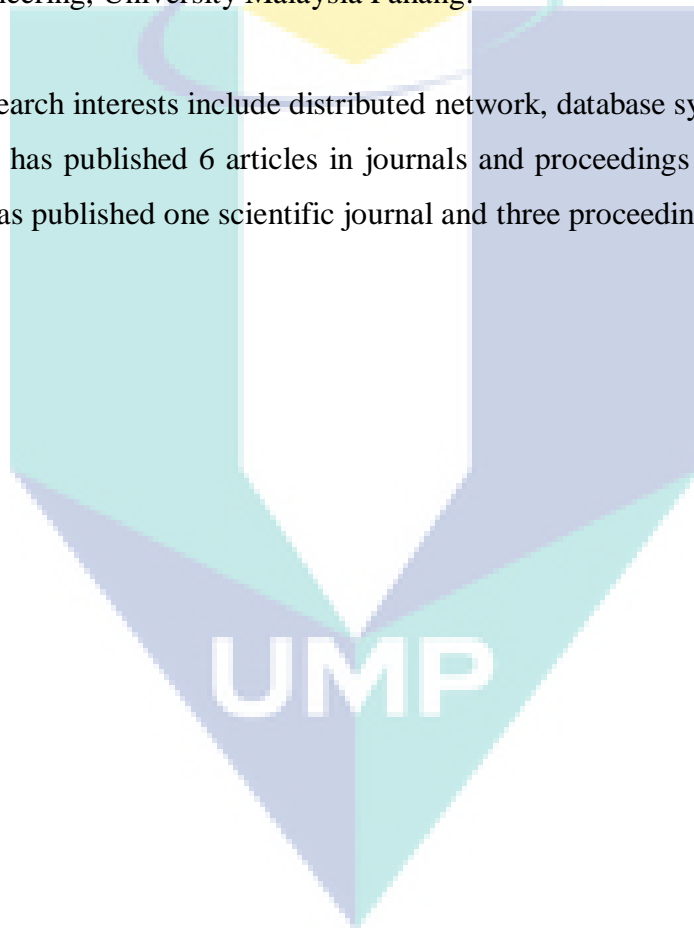
```
String query = "Insert into user values ('+id+', '"+name+', '"+faculty+')";  
  
st.executeUpdate(query);  
  
st.close();  
conn.close();  
System.out.println("Done");  
}  
catch (Exception e)  
{  
    e.printStackTrace();  
}  
}
```



BIODATA OF THE AUTHOR

The author was born in 1981 in Tangail, Bangladesh. He obtained bachelor degree in Computer Science and Engineering in 2005 from Darul Ihsan University, Bangladesh. Currently he is undergoing his M.Sc program at the Faculty of Computer Systems and Software Engineering, University Malaysia Pahang.

His current research interests include distributed network, database systems and database replication. He has published 6 articles in journals and proceedings (international). For this work, he has published one scientific journal and three proceedings.



LIST OF PUBLICATIONS

1. **A.H.Beg**, Noraziah Ahmad, Ahmed N Abd Alla, Nawsher Khan, “Framework of Persistence Layer for Synchronous Data Replication (PSR)”, *Australian Journal of Basic and Applied Sciences*, 4(10): 5394-5400, 2010. **Index: ISI.**
2. **A.H.Beg**, Noraziah Ahmad, Ahmed N Abd Alla, K.F. Rabbi, Nawsher Khan, “ Structure and Framework of Synchronous Replication Based On Data Persistency to Improve Data Availability into a Heterogeneous System”, *International Conference on Software and Computing Technology, IC SCT 2010*, Kunming, China.vol.1, pp. 127-130. **Index: IEEE.**
3. **A.H.Beg**, Noraziah Ahmad, Ahmed N Abd Alla, K.F.Rabbi, Soffia Suliman, “Architecture and Algorithm of Heterogeneous Persistence Layer for Synchronous Replication Based on Multi-threading”, *Annual International Conference on Advances in Distributed and Parallel Computing, ADPC 2010*, Mandarin Orchard, Singapore, pp. 64-69. **Index: EBSCO, Scirus, EI Compendex, ISTP.**
4. **A.H.Beg**, A.Noraziah, Ahmed N Abdalla, Noriyani Mohd Zin, E.I.Sultan, “Synchronous Replication: Novel Strategy of Software Persistence Layer Supporting Heterogeneous System”, *2nd International Conference on Software Engineering and Computer Systems, ICSECS 2011*, Pahang, Malaysia, Part II, CCIS 180, pp. 232–243, 2011.Springer-Verlag Berlin Heidelberg 2011. **Index: ISI Proceedings and Scopus.**