Migration of SQL Database to MongoDB Database with a Multi-thread Approach

By: Fikru Kebede

A Thesis Submitted to the Department of Computing

School of Electrical Engineering and Computing

Presented in Partial Fulfillment for the Degree of Masters of Science in Software Engineering

Office of Graduate Studies

Adama Science and Technology University

Jun 2019

Adama, Ethiopia

Migration of SQL Database to MongoDB Database with a Multi-thread Approach

By: Fikru Kebede

Name of Advisor: Dr. Ravindra Babu.B

A Thesis Submitted to the Department of Computing

School of Electrical Engineering and Computing

Presented in Partial Fulfillment for the Degree of Masters of Science in Software Engineering

Office of Graduate Studies

Adama Science and Technology University

Jun 2019

Adama, Ethiopia

DECLARATION

I hereby declare that this MSc thesis is my original work and has not been presented as a partial degree requirement for a degree in any other university, and that all sources of materials used for the thesis have been dully acknowledged.

Name: Fikru Kebede

Signature: _____

This thesis has been submitted for examination with my approval as thesis advisor.

Name: Dr. Ravindra Babu.B

Signature: _____

Date of Submission: _____

Approval of Board of Examiners

We, the undersigned, members of the Board of Examiners of the final open defense by _____have read and evaluated his/her thesis entitled "_____" and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirement of the Degree of ……

_____	_____	_____
Supervisor/Advisor	Signature	Date
_____	_____	_____
Chairperson	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

Acknowledgement

First and foremost, I am thankful to **Almighty God** and **his Mother St. Merry**, who kindly helped me to complete my thesis. Without his blessings I wouldn't have been writing a single word. Then I would like to express my special appreciation and thanks to my advisor, Dr.Ravindra Babu.B for the continuous support of my research and for his motivation.

Then, a special thanks and appreciation to my family. Words can't express how glad I am to my mother and my father for all of the sacrifices that you made to my current status and your prayer for me was what sustained me this far. A special thanks to my wife Tiku, for her motivation patience and support. I would also like to thank my friend Ashenafi Alemu for his continuous comment, motivation and encouragement. And finally, I would like to thank Mr. Endris Mohammed for commenting and suggesting ideas on my thesis.

Table of Contents

List of Figures

List of Tables

List of Equations

List of Algorithms

Abbreviations and Acronyms

Acronym	Definition
1) API	Application Programming Interface
2) AJAX	Asynchronous JavaScript and XML
3) AWS	Amazon Webservice
4) BSON	Binary JSON
5) CPU	Central Processing Unit
6) CRM	Customer Relationship Management
7) DC	Data Center
8) DMM	Data Migration Manager
9) DMT	Database Migration Tool
10) ER	Entity Relationship
11) ETL	Extract, Transform and Load
12) ETSI	European Telecommunications Standards Institute
13) GB	Giga Byte
14) GCE	Google Compute Engine
15) GHz	Giga Hearth
16) HDFS	Hadoop Distributed File System
17) HDS	Hitachi Data Systems
18) HP	Hewlett-Packard
19) HTML5	Hyper Text Markup Language version 5
20) IDG	Internet Development Group
21) IP	Internet Protocol

22) IT	Information Technology
23) LPPL	LaTeX Project Public License
24) LUN	Logical Unit Numbers
25) OASIS	Organization for the Advancement of Structured Information Standards
26) OCCI	Open Cloud Computing Interface
27) SNIA	Storage Networking Industry Association
28) SQL	Structured Query Language
29) DMTF	Distributed Management Task Force
30) ISO	International Standard Organization
31) BLOB	Binary Large Object
32) Gen	Generation
33) HDD	Hard Disk Drive
34) I/O	Input Output
35) IaaS	Infrastructure as a Service
36) ICT	Information Communication Technology
37) IDE	Integrated Development Environments
38) IETF	International Engineering Task Force
39) IJIRSET	International Journal of Innovative Research in Science, Engineering and Technology
40) IoT	Internet of Things
41) ISO	International Organization for Standardization
42) JSON	JavaScript Object Notation
43) MHz	Mega Hearth
44) NIST	National Institute of Science and Technology

45) OSA	Open Security Architecture
46) PaaS	Platform as a Service
47) PC	Personal Computer
48) RAM	Random Access Memory
49) REST	REpresentational State Transfer
50) RQ	Research Question
51) SaaS	Software as a Service
52) SSL	Secure Socket Layer
53) TOG	The Open Group
54) USP	Universal Storage Platform
55) VPC	Virtual Private Cloud
56) XML	eXtensible Markup Language

Abstract

A cloud computing is an emerging and the current choice of computing model by its dynamically elastic and often virtualized resources delivery nature at the levels of infrastructures, platforms, and software. It is becoming a possible choice for most IT industries and other sectors to benefit out of its advantages. Most of the industries are migrating their legacy IT infrastructures, platforms, and software to the cloud computing environment and also from one cloud service provider to the other even if the migration process is difficult. One of the reasons that make the migration process more difficult is, the incompatibility of data format on the existing and destination platform.

The concern of this study is to design a database migration algorithm by using a multi-threaded approach to migrate data concurrently, JSON data interchange format for the transmission of data and an SSL protocol for securing the transmission of data. The database migration process is separated into pre-migration, during migration and post-migration stages. The proposed solution tried to migrate a database by dividing the total migration task into subtasks based on three conditions: a relational database with multiple schemas, a single schema with multiple relations and one database schema with a single relation but with a large number of records and the solution migrates the database based on thread per schema, thread per relation and thread per 100,000 records respectively.

Keywords: Database Migration, JSON data interchange format, Multi-threading in data migration, Programming language for data migration, NoSQL database, Relational Database, Bigdata migration

Chapter One

1. Introduction

1.1. Background of the study

The advancement of the three technologies (i.e. the microprocessor, storage device, and internet) provides a great contribution to the radical change of computing paradigm. Through the history of computing, lots of computing paradigms are implemented such as desktop computing, centralized (client/server computing), distributed computing, cluster and grid computing, Cloud Computing and Internet of Things (IoT). And that indicates, through time the computing demand of the individuals and industries is increasing at a high rate.

For such kind of computing demand, desktop computing and centralized computing can't be appropriate with different kinds of limitations such as scalability, cost, management, and reliability. By its nature, cloud computing has the ability to handle such kinds of computing requirements and due to that reason, most IT infrastructures and services are migrating to the cloud computing environment. Cloud computing is the current pay-as-you-go computing technology that is based on the internet and aims to provides everything as a service with the top three service delivery models (IaaS, PaaS and SaaS) and five deployment models (public, private, hybrid, community and virtual private) with the characteristics of rapid elasticity, broad internet access, on-demand self-service, resource pooling and measured service [1] [5].

Nowadays, different kinds of electronic equipment such as drones, vehicles, medical devices, etc. are generating a huge amount of data within seconds. Therefore, shifting to a cloud computing environment is the ultimate solution to handle such computing requirement. Because of the development in networking technology and the increasing demand of computing resources, many companies have been prompted to outsource their storage and computing needs, regarding the ever-cheaper hardware resources provided by cloud providers [2].

Due to different reasons, migrating from one cloud provider to the other is a well-known issue in a cloud environment. Migration in cloud computing is a process which involves moving a large amount of data or applications or the whole virtual machine to the target cloud [3]. And this research tried to maximize the interoperability among different clouds by providing a data migration facility as a service.

Nowadays, cloud computing becomes the best choice of computing interest in IT sectors and industries from different aspects such as cost, management, maintenance, simplicity, etc. But this technology has its own problems such as Privacy and Security, Performance, Latency and Reliability, vendor lock-in, Portability and Interoperability, Data-Breach through Fiber Optic Networks, Data Storage over IP Networks [4].

From these challenges of cloud computing, the aim of this research is to minimize the vendor lock-in, portability, and interoperability issue by implementing a data migration as a service web service that allows data migration between different cloud providers. The issue of migrating data from one cloud provider to the other or from the existing non-cloud database system to the cloud environment is the most difficult task [10]. Therefore, this research aims to minimize the migration process of data from source to target.

1.2. Motivation

Cloud computing is an interesting computing technology in different aspects of information technology such as cost, management, maintenance, simplicity, etc. With these aspects, users are subscribing/renting cloud computing service from cloud providers. But in the meantime, cloud users might have an interest in subscribing service from another cloud provider for different reasons. In this case, the key asset of the user is its data even if the virtual machines, applications, configurations, and other assets are important, we can recreate, reinstall and reconfigure the other assets respectively. So, contributing a solution to increase the interoperability of cloud computing ecosystem to enable users for migrating data easily is an interesting task.

1.3. Statement of the problem

Because of the advantages of cloud computing, organizations are shifting their existing IT systems into the cloud computing environment. In addition to that, cloud users might be migrated from one cloud provider to the other. The reasons for migrating from non-cloud IT system to the cloud environment is due to the advantages and characteristics of cloud computing. There are different kinds of reasons that a specific cloud user decides to migrate to a new cloud provider. Some of the reasons are dissatisfaction of service, better alternative, change business or technology strategy, low cost, and failure of provider [1, 2]. And also, as the demand for cloud computing services increases, the competition between cloud service providers also increases. For cloud clients to benefit from such competition, they should be able to freely and easily migrate their data from one cloud to another [2].

When migrating IT infrastructure from the existing cloud provider to a new cloud provider, from a legacy system to a cloud computing environment or from one database system to the other , one of the challenging issues is the incompatibility of database architecture. And to fit the new provider's standards and implementation, the required migration of an application or data from one cloud provider to another may require a significant effort and/or full-cycle of redevelopment [11]. For example, the source database could be relational and target database could be document-oriented and vice versa. In addition to that, the other challenge in cloud computing area is data portability between different clouds [1]. Especially, if the user subscribes SaaS, it is difficult to get the same platform on the other cloud provider. In this case, data integration utilities are important.

According to IDG Enterprises' 2014 Cloud Survey, 29% of IT leaders said switching from one cloud provider to another is more difficult [17]. And the main challenge is its portability issue. So, it is very important to have an easy, fast, secured and efficient mechanisms that enable users to transparently copy and move their data from one provider to another [2]. So, an appropriate data migration algorithm is required to migrate data form source platform to the target platform. Therefore, the following hypothesis and research questions are defined.

Hypothesis: It is possible to enhance the performance of database migration by applying a multi-threaded approach.

RQ1: How can be a multi-threaded application is implemented for a big data migration?

RQ2: Which programming language and data-interchange technology is more appropriate to implement the data migration algorithm?

RQ3: How does document-oriented database (NoSQL) is better than relational database for big data management?

1.4. Objectives

The thesis work has the following general and specific objectives.

1.4.1. General objective

The main objective of this study is to design relational database (RDB) to document-oriented (MongoDB) database migration algorithm in cloud computing environment by using a multi-threaded program and JSON data-interchange format.

1.4.2. Specific Objective

For the achievement of the general objective of the research, four specific objectives are identified:

- To study the current state-of-the-art cloud computing database technologies and data migration techniques through literature review
- To analyze the data portability and interoperability issue in cloud computing
- To evaluate the advantage of multi-threaded programs in the case of big data migration
- To evaluate the performance of JSON data-interchange format in the data migration process
- To increase interoperability among different cloud providers

- To minimize vendor lock-in problem from the cloud service subscribers' side

1.5. Scope and Limitations

The main aim of this study is to develop a database migration algorithm to migrate database from a relational database system to document-oriented (NoSQL) database system in a cloud computing environment. In particular, the experiment focuses on migrating data from MS SQL as a relational database to MongoDB as a document-oriented database (NoSQL). Even if dozens of database systems and technologies are available with different architectural style, in this study, the migration of data for the other database systems is not included.

1.6. Significance of results

On the ecosystem of cloud computing, migrating from one cloud service provider to the other is a common issue. But cloud computing service subscribers are suffering from a vendor lock-in problem [25] [26]. One of the reasons for the vendor lock-in problem is the issue of interoperability and portability. And these issues should not be a constraint to cloud users as well as cloud providers. The proposed solution has three kinds of beneficiaries; the cloud service subscribers, cloud service providers and researchers. The significance of the proposed solution is that:

- From the cloud service subscriber side, the freedom to change a service provider will be maximized. That means the vendor lock-in problem can be minimized by the implementation of the database migration tool.
- From the cloud provider side, the interoperability issue among different cloud providers can be minimized
- And researchers can migrate an experimental dataset from a relational database to NoSQL database easily with the help of the proposed database migration algorithm.

1.7. Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 will be a systematic literature review, literature will discuss in four categories which are cloud infrastructure, cloud monitoring, cloud monitoring properties, the cloud infrastructure monitoring and dynamic threshold. Chapter 3 is a research method, in this chapter the way how the research is organized will discuss. Chapter 4 is about the proposed solution. Chapter 5 is the experiment and evaluation of the result; the proposed solution is evaluated using evaluation tools and the result of the evaluation will discuss. Chapter 6 is all about the conclusion, recommendations and the future works, and finally a reference.

Chapter Two

2. Literature Review

2.1. Overview

The main objective of this chapter is to describe the research area on data migration, database technologies running on the cloud, architecture, and characteristics of cloud computing, the leading cloud service providers, cloud service delivery models, cloud deployment models, data migration tools and techniques, vendor lock-in problem in cloud computing, portability and interoperability issue in cloud computing and data interchange technologies.

Interoperability issue among different cloud providers is the main reason for the vendor-lock-in problem and that forbids cloud users from navigating between cloud provides easily in the cloud computing business. One of the main reasons for the interoperability issue between cloud systems is that the data and database standard of cloud providers is different.

Descriptions about interoperability issue among cloud providers and state-of-the-art solutions, data migration techniques and existing works on data migration in a cloud computing environment are described in a way to explore the research area and understand the need and importance of such a study.

2.2. Cloud computing

Cloud computing was defined by different organizations (e.g. NIST, ISO and Gartner) and scholars in different ways. NIST defined as, "*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"* and also it offers clients applications, data, computing resources, and IT management functions as a service through the Internet or a dedicated network [1][5].

The growth of Cloud Computing is rapidly changing the landscape of information technology and ultimately turning the long-held promise of utility computing into a reality because cloud computing comes with a pay as you go model like other household utilities (like electricity, water, and telephone) [7].

2.2.1. Essential characteristics of cloud computing

NIST mentioned five essential characteristics of cloud computing [1]. These characteristics are:

- **Broad network access:** Provisioned capabilities of the provider are available over the network and can be accessed through a standard mechanism by using heterogeneous thin and thick clients.
- **On-demand self-service:** Which means the cloud service subscriber can get computing capabilities as needed automatically without any human interaction with each service provider.
- **Rapid elasticity:** Provider capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward corresponding with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
- **Resource pooling:** The provider's computing resources are pooled to serve multiple tenants using a multi-tenant model, with different dynamically allocated and reallocated physical and virtual resources based on consumer demand. There is a big sense of location independence and the customer generally has no knowledge or control over the exact location of the provided resources but the location can be specified at a higher level of abstraction. Storage, processing, memory, and network bandwidth are examples of resources.
- **Measured service:** The cloud systems control and optimize automatically the usage of resource by leveraging a measuring capability at some level of abstraction appropriate to the type of services such as storage, processing, active user accounts, and network

bandwidth. The consumption of resource can be monitored, controlled, and reported to provide transparency for both the consumer and provider of the used service.

2.2.2. Cloud computing reference frameworks and architectures

When comparing cloud technical architectures, informing business stakeholders to evaluate cloud services, and providing for a common cloud taxonomy across organizational boundaries cloud reference frameworks are important tools and which is composed of a variety of reference architectures and reference models, that collectively describe all of the relevant aspects in a context that can then be applied to particular stakeholder viewpoints and interests [8]. Lots of standard organizations and technology companies proposed their own cloud reference models. From these, NIST, IETF, TOG, ETSI, OASIS, SNIA, OSA, OCCI, ISO, DMTF, and CSA contributed different cloud reference frameworks.

Each reference model has its own unique concerns. For example, the NIST reference architecture makes the security concern of cloud providers, consumers, and other parties. In addition to that, it enables the use case and scenario planning methods by providing major actor and service constructs for cloud [1][8] [10].

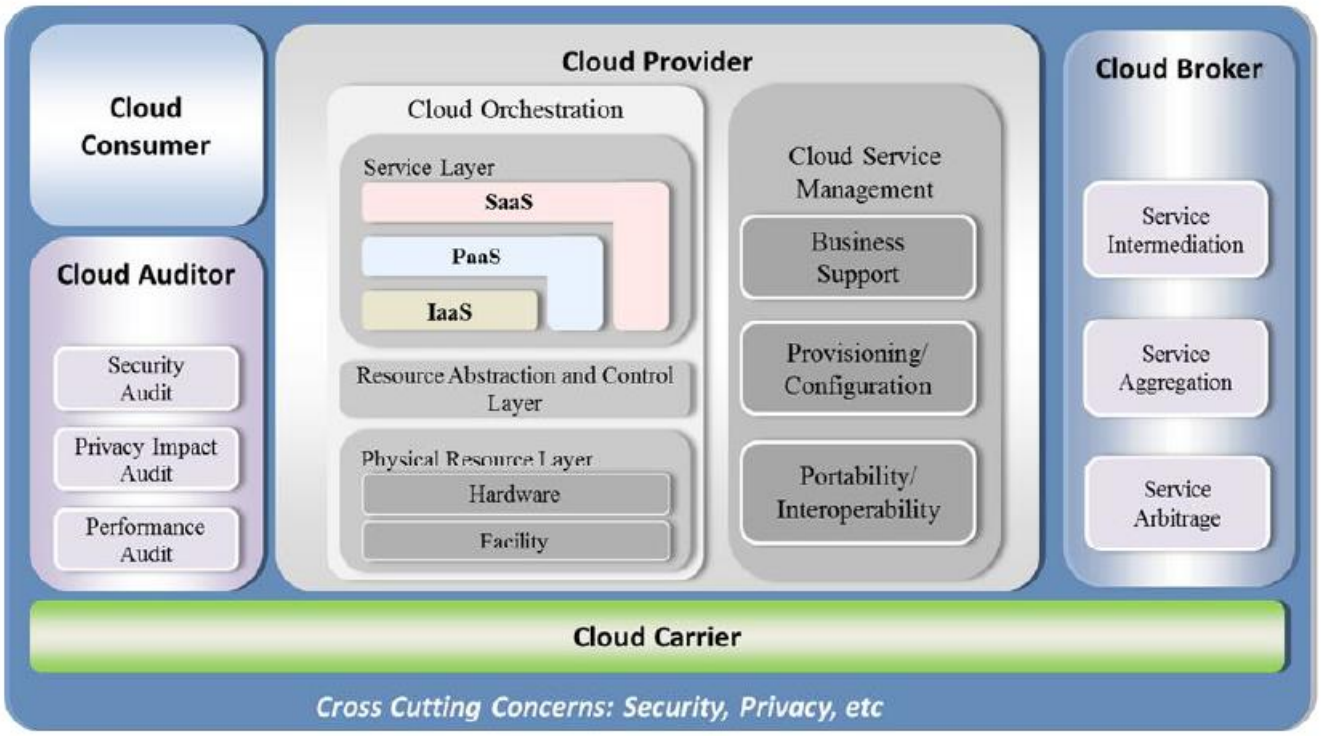

Figure 2.1 Cloud reference architecture by NIST [10]

The other cloud reference framework is developed by the Cloud Security Alliance (CSA). CSA is a not-for-profit organization that encourages research for the best practices of securing cloud computing and to secure the other computing forms of cloud technologies [8][9]. The main aim of this reference model is to provide the essential architectural building blocks for engaging security and compliance for cloud enterprises.

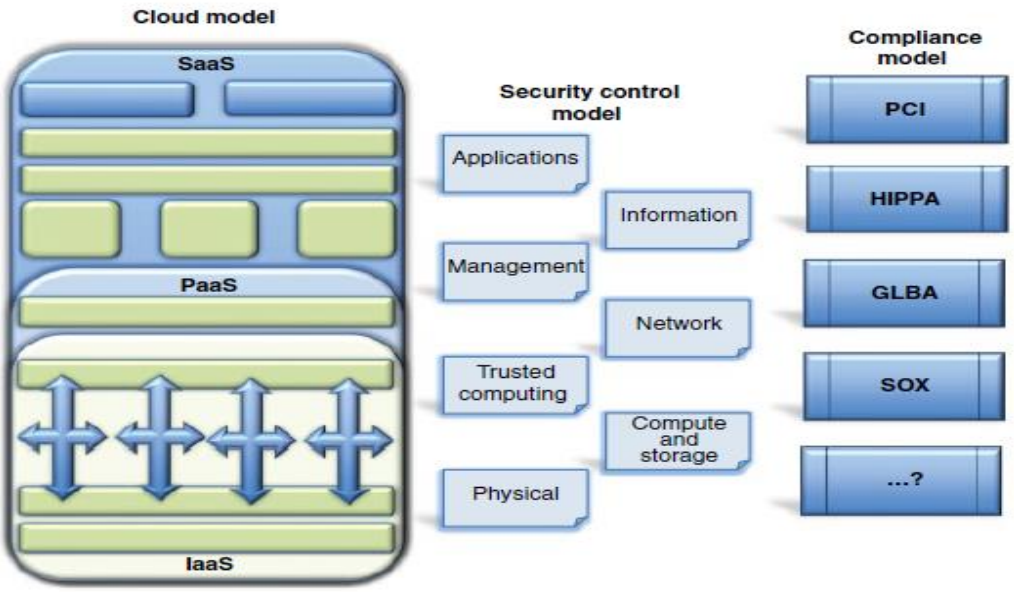

Figure 2.2 The CSA cloud, security, and compliance reference model [8]

2.2.3. Service delivery models in cloud computing

Provision of everything as a service is the main principle of cloud computing. That means, each and every kind of computing resource such as, software, platform and infrastructure are available as a service based on the demand of the user from the resource pool of cloud service providers. The best nature of cloud computing services is its scalability and availability. The service can be scaled out and scaled in as per the demand of the subscriber of the service plus the service is available every-time and everywhere.

Even if, several kinds of cloud services are emerging, such as security as a service (SeaaS), knowledge as a service (KaaS) and analytics as a service (AaaS), the services provided by cloud service providers can be broadly classified into three major types: Software as a service (SaaS), Platform as a service (PaaS) and Infrastructure as a service (IaaS) [4] [5].

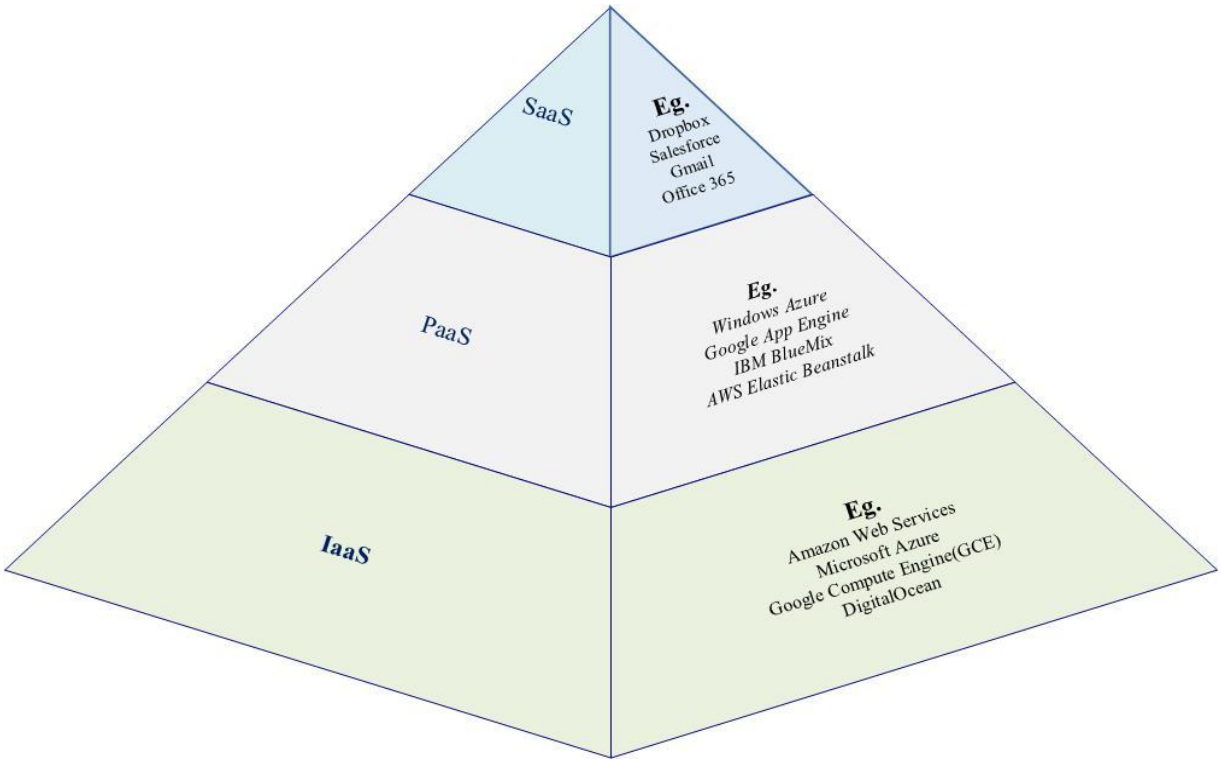

Figure 2.3 Service delivery models in cloud computing

a. Software as a service

SaaS is a licensing and distribution model used to deliver software application as a service over the Internet. It is the most utilized cloud service on a cloud computing ecosystem. Software as a service cloud computing service model is also called a software cloud. In the SaaS model, the cloud provider hosts an application and delivered as a service to users, mainly through the Internet or a dedicated network [5]. There are different service providers from commercial up to free services such as Google (Gmail, google docs, google map, etc.), Microsoft (Office 365) and Salesforce.com (CRM) [37][38][39]. On the cloud computing environment, SaaS is the most available and the cheapest cloud computing service delivery model.

b. Platform as a service

Platform as a Service (PaaS) or the cloud platform service, provide a cloud-based framework and other components for developers that they can build upon and use to develop software

applications. Therefore, the capability provided to the service subscriber enables to deploy the consumer-created or developed applications using programming languages, libraries, tools, and services supported by the provider onto the cloud infrastructure [1][5] [10]. The PaaS service delivery model is similar to SaaS, except instead of delivering the software over the internet, PaaS delivers a platform for software development. For example, Microsoft provides Windows Azure, Google provides App Engine, and Amazon provides AWS Elastic Beanstalk.

c. Infrastructure as a Service

In an IaaS cloud service delivery model, raw computer infrastructure, such as processing, network, data center facilities, and other fundamental computing resources are provided to the subscriber as a service on demand [5] [10]. Mainly, IaaS provides virtualized computing resources over the internet. The virtualized resources such as servers, storages, processing, and network resources are available in an on-premises data center.

2.2.4. Cloud computing service deployment models

For delivering each and every kind of computing resource as a service and to satisfy different requirements of the customers such as security, cost, and reliability, the computing resources should be deployed with different deployment strategies. Therefore, in cloud computing the are five types of service deployment models namely: Private cloud, Public cloud, Virtual Private cloud, Community cloud and Hybrid cloud [5].

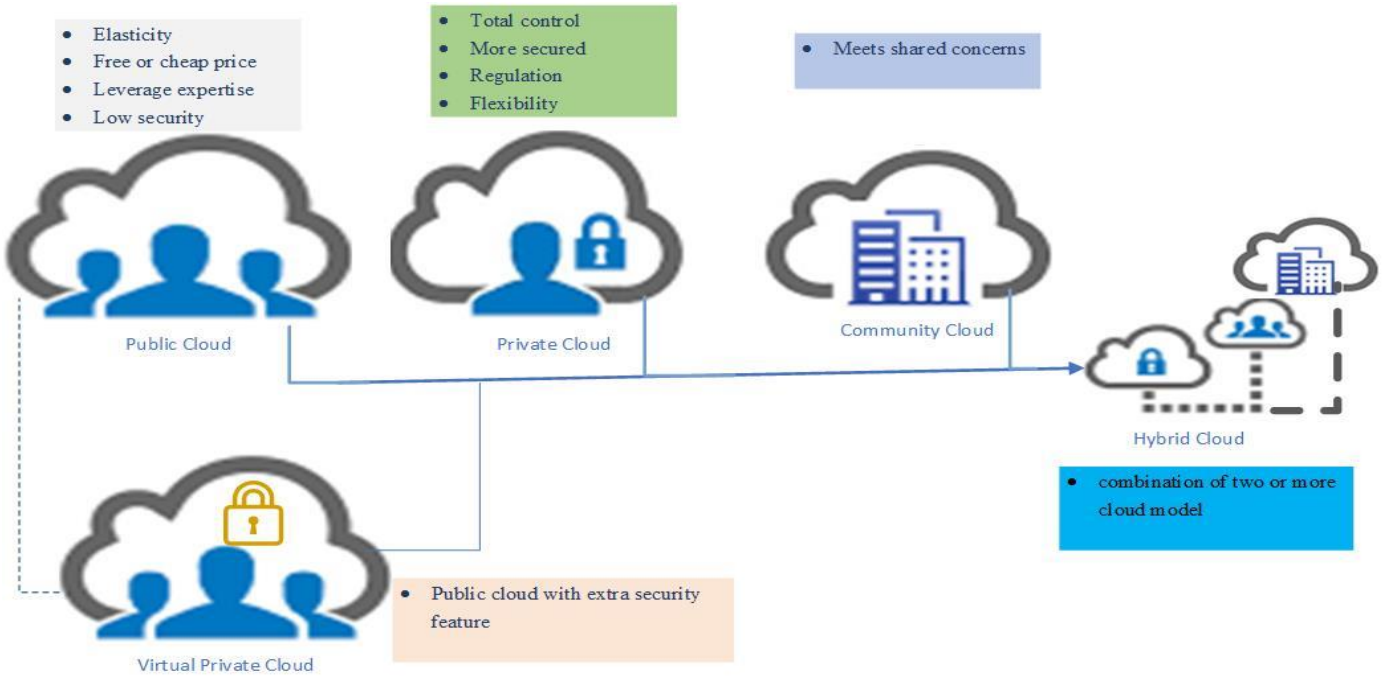

Figure 2.4 Cloud computing service deployment models

a. Private cloud

The cloud service is provided for exclusive use by a single organization or enterprise behind its firewall for its own use and it could include multiple consumers and it can be owned, operated and managed by the organization, a third party, or some combination of them, and it may exist on or off premises [1][5].

b. Public cloud

The public cloud is the most common and widely known form of a cloud, and the cloud infrastructure is provisioned for open use by the general public [1].

c. Community Cloud

A community cloud is an industry cloud provisioned to the community of industries having the same security and business interest. It is used by a particular industry sector or a group of users and especially deployed to meets specific requirements to address issues that are crucial to them [5].

d. Hybrid Clouds

The result of the combination of two or more of the above cloud deployment models is a hybrid cloud and it has the advantages of its components such as security of private cloud and cost benefit from the public cloud.

e. Virtual Private Cloud (VPC)

VPC is a segment of a public cloud, designated for a user with extra provisions for meeting that user's specific security and compliance requirements. And also, VPC provides users with more control over the resources they use than a pure public cloud does such as Amazon's VPC [5].

2.2.5. Advantages of cloud computing

Cloud computing provides lots of advantages over the other former computing paradigms. One of the well-known cloud computing service providers "SalesForce.com" identified 12 advantages of cloud computing. Those advantages are Cost saving, Better security, Flexibility, Mobility, Insight, maximized collaboration, Quality control, Disaster recovery, Loss prevention, Automatic software update, Competitive edge and Sustainability [12] [13].

2.2.6. Limitations of cloud computing

Even if, cloud computing is more advantageous than the former computing platforms, it has its own limitations. Some of the limitations of cloud computing are, vendor lock-in, privacy and security, connectivity and open access, service availability, absence of proper service level agreement, interoperability and changes in IT organization [13][25]. Regarding the solution what this study tries to solve, vendor lock-in, data portability, and compatibility issues are concerned.

i. Vendor Lock-in

In cloud computing, the vendor lock-in problem is characterized by a time-consuming and expensive migration of application and data to an alternative cloud service provider [15].

According to J. Opara-Martins et al. inability to move data from one cloud service vendor to the other service provider or back to our IT infrastructure is the 5th cause of vendor lock-in problem for cloud computing environment [16].

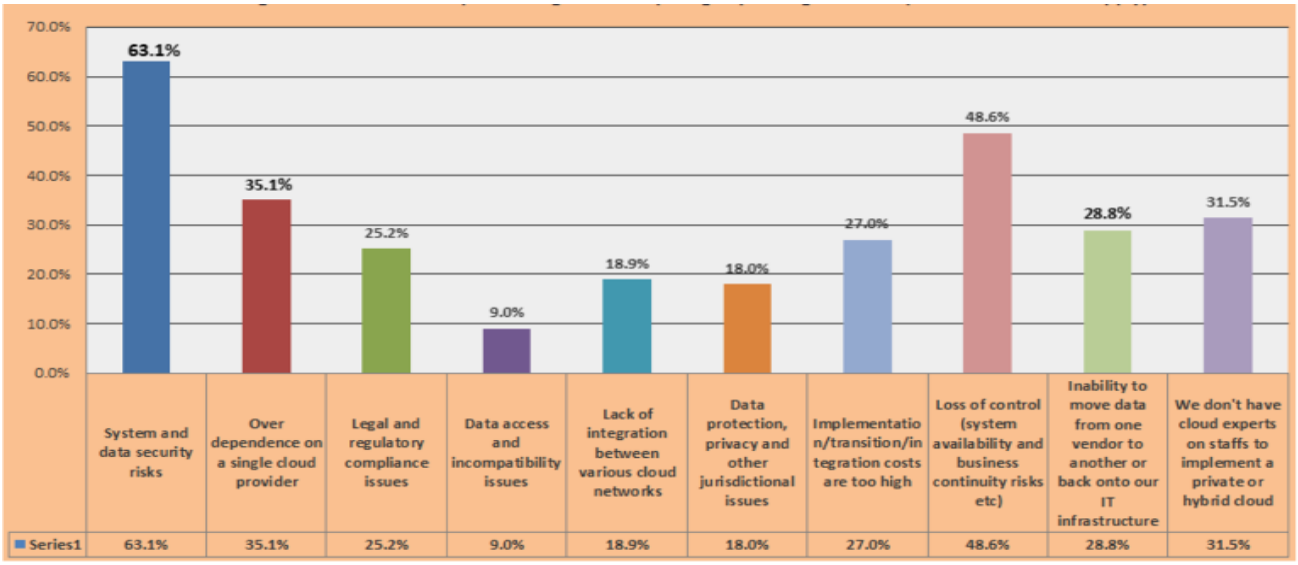

Figure 2.5 Barriers to cloud implementation in the UK [16]

Therefore, the design and implementation of a database migration solution minimize the vendor lock-in problem. Most of the database management systems have an import/export or backup/restore mechanism. But this mechanism works well if the user subscribes the IaaS service delivery model. Because the subscriber of the service can install a compatible database infrastructure/platform on IaaS. So, database migration service is mandatory for the remaining service delivery models.

ii. Data portability and compatibility

Even if there are different database migration solutions provided by the largest IT companies such as Microsoft, Amazon, and Oracle, the solutions are targeted to migrate a database into their own cloud service. For example, Amazon has a database migration service to migrate a database into the AWS cloud by targeting its own database platforms and the same is true for Microsoft and Oracle. The detailed database migration tools and techniques provided by the greatest IT service providers are discussed in the following section.

Different cloud providers have different kinds of database technologies and platforms. For example, some of the cloud service providers have an SQL database strategy and the others may follow the NoSQL strategy.

2.3. Database migration tools and techniques

2.3.1. Database Migration tools

Database migration is the process of moving a database from one platform to the other within a heterogeneous or homogeneous environment. The migration process could be from a legacy database system to a cloud-based database system or from a cloud-based database system to a cloud-based database system or from a cloud-based database system to a non-cloud database system. To migrate the database from source to the target a migration tool is required. So, a database migration tool is a software tool that lets companies transfer data from one type of database to another, or from a database to another type of data repositories such as a data warehouse or data lake, without having to rely on manual coding or overly complicated ETL tools [14].

Database migration is a complex, multistep process, which typically includes assessment, database schema conversion, conversion of a script, data migration, functional testing, performance tuning, and many other steps [17]. According to Amazon's database migration checklist size of the database, a number of schemas and tables, availability of engine-specific data types and availability of enough bandwidth to migrate the database are important to determine the possibility of database migration. There are three fundamental types of database migration tools to migrate database [18]. These are,

- **On-premise tools**: Used to migrate data within an Enterprise network installation.
- **Open Source tools**: A free or very low-cost database migration tools that can be supported and developed by the community.
- **Cloud-based tools**: Designed to migrate the database to the cloud from various sources and streams, such as on-premise and cloud-based data stores, applications, services, etc.

S.No	Name	Company	Source	From	To	Operating System
1	AWS Database Migration service	Amazon	Closed	DB2, MS SQL Oracle, SyBase, Informix, Aurora, PostgreSQL, Teradata, Netezza, Greenplum, MySQL, Vertica & MariyaDB	Aurora, MySQL, PostgreSQL, MariaDB and Redshift	Windows, Linux, Unix & Mac OS
2	Microsoft Azure Database Migration Service	Microsoft	Closed	MySQL, Oracle, Microsoft SQL	Microsoft SQL Azure	Windows
3	OSDM Toolkit	Apptility	Open	DB2, MS Access, MS SQL Oracle, SyBase, Informix,	PostgreSQL& MySQL	Windows, Linux, Unix & Mac OS
4	DB Migration	Akcess	Closed	Oracle & MS SQL	PostgreSQL& MySQL	Windows
5	Mssql2 Pgsql	OS Project	Open	MS SQL	PostgreSQL	Windows
6	MySQL Migration Toolkit	MySQL AB	Open	MS Access & Oracle	MySQL	Windows

7	MySQL Migration Toolkit	Intelligent Convertors	Closed	MS Access, MS SQL, Dbase & Oracle	MySQL	Windows
8	Open DBcopy	Puzzle ITC	Open	Any RDB*	Any RDB*	OS Independent
9	Progression DB	Versora	Open	MS SQL	PostgreSQL, MySQL & Ingres	Linux & Windows
10	Shift2Ingres	OS Project	Open	Oracle & DB2	Ingres	OS Independent
11	SQLPorter	Real Soft Studio	Closed	Oracle, MS SQL, DB2 & Sybase	MySQL	Linux, Mac OS & Windows
12	SQLWays	Ispirer	Closed	All Relational Databases	PostgreSQL & MySQL	Windows
13	SwisSQL Data Migration Tool	AdventNet	Closed	Oracle, DB2, MS SQL, Sybase & MaxDB	MySQL	Windows
14	SwisSQL SQLOne Console	AdventNet	Closed	Oracle, MSSQL, DB2, Informix & Sybase	PostgreSQL & MySQL	Windows

15	MapForce	Altova	Closed	SQL Server, DB2, MS Access, MySQL & PostgreSQL	SQL Server, DB2, MS Access & Oracle	Windows, Linux & Mac OS
16	Centerprise Data Integrator	Astera	Closed	SQL Server, DB2, MS Access, MySQL & PostgreSQL	SQL Server, DB2, MS Access, MySQL & PostgreSQL	Windows
17	DBConvert	DB Convert	Closed	Oracle, DB2, SQLite, MySQL, PostgreSQL, MS Access & Foxpro	Oracle, DB2, SQLite, MySQL, PostgreSQL, MS Access & Foxpro	Windows
18	Centerprise Data Integrator	Astera Software	Closed	Oracle, IBM DB2, MySQL, PostgreSQL, Sybase, Teradata, Netezza, Microsoft SQL Server, Redshift, MS Access	Oracle, IBM DB2, MySQL, PostgreSQL, Sybase, Teradata, Netezza, Microsoft SQL Server, Redshift, MS Access	Windows

Table 2.1 Database migration tools

Based on the source of the software, the data migration tools can be grouped into closed source and open source. Closed source tools are a commercial database migration tools whereas open sources tools are freely available data migration tools.

Even if there are different database migration tools, the leading database migration service is provided by Microsoft and Amazon. But AWS database migration service has more source and target database platforms.

a. AWS Database Migration

Amazon web service provides a good database migration service. The AWS Database Migration Service provided by Amazon helps you migrate your databases to AWS with virtually no downtime [19]. Amazon provides migration service for homogeneous database platforms such as Oracle-to-Oracle as well as heterogeneous database platforms such as Oracle-to-MySQL migration service.

But, there are some limitations of amazon database migration service, such as:

- The database is migrated to the Amazon cloud system only and,
- There is no schema migration service for MongoDB.

The table below describes the source to target mapping relationships for AWS database migration service.

Source Database	Target Database on Amazon RDS
Oracle Database	Amazon Aurora, MySQL, PostgreSQL, MariaDB
Oracle Data Warehouse	Amazon Redshift
Microsoft SQL Server	Amazon Aurora, Amazon Redshift, MySQL, PostgreSQL, MariaDB
Teradata	Amazon Redshift
IBM Netezza	Amazon Redshift
Greenplum	Amazon Redshift
HPE Vertica	Amazon Redshift
MySQL and MariaDB	PostgreSQL
PostgreSQL	Amazon Aurora, MySQL, MariaDB
Amazon Aurora	PostgreSQL
IBM DB2 LUW	Amazon Aurora, MySQL, PostgreSQL

Table 2.2 AWS Database Migration Service source and target database platforms

b. Microsoft Azure database migration service

The other popular cloud computing service provider is Azure database migration service. Microsoft provides an Azure cloud computing service to its clients and the clients must have one of the following legacy database systems to migrate into the Azure cloud environment by using the Microsoft Azure database migration service.

On-Premises Database	Target Database on Azure
SQL Server	Azure SQL Database Azure SQL Database Managed Instance SQL Server on Azure virtual machines
Oracle Database	Azure SQL Database Azure SQL Database Managed Instance SQL Server on Azure virtual machines
MySQL	Azure SQL Database Azure SQL Database Managed Instance SQL Server on Azure virtual machines

Table 2.3 Microsoft Azure Database Migration Service source and target database platforms [21]

The above table describes, what source and target database platform are possible to do migration with Microsoft Azure database migration service. The limitations in Microsoft Azure database migration service is that,

- The source database systems are SQL Server, Oracle, and MySQL only.
- The target database platform is Microsoft SQL on Azure and Azure SQL database and
- Not possible to migrate a database into other cloud vendors

Generally, the tools provided by different solution providers can't migrate SQL database to MongoDB database, requires the installation of the database migration tool at the client machine, users can't migrate database using thin clients and some of the tools provide a migration service to migrate database to the service provider's platform only. So, the proposed solution can fill such kind of gaps.

2.3.2. Data Migration techniques

The database migration technique describes how the source database is accessed, how the schema is converted, how is the data type converted and how the actual data is copied/moved.

All database migration tools have their own migration technique. According to a report on TechTarget, there are five fundamental data migration techniques [20].

1. **Host-based file-level migration,** to migrate largely static and unstructured content, host-based file-level migration is the best. The best example of this technique is rsync.
2. **Host-based block-level migration**, this kind of technique is used to migrate large structured files such as databases, this technique is the best choice and which is already installed with most operating systems.
3. **Network-based file-level migration**, this type of data migration technique is preferred and used when data migration can't be done on the host.
4. **Network-based block-level migration**. This technique is the way of moving the data online and doesn't have to take control of the logical unit numbers (LUNs). The best example of this kind of technique is Brocade's Data Migration Manager (DMM).
5. **Array-based block-level migration**. This kind of data migration technique works great if the customer already has or is moving toward an HDS USP array and the hosts cannot support the workload required to move the data. Example, HDS's Universal

2.4. Relational Database Vs NoSQL Database

Relational databases are widely used in most of the applications with a good performance when they handle a limited amount of data. In case of huge volume of data (Big data) like internet, multimedia and social media the use of traditional relational databases is inefficient. To overcome such kind of problem the **Not only SQL** or **NoSQL** concept was introduced.

Handling unstructured data such as documents, email, multimedia and social media efficiently is the main advantage of a NoSQL database. And NoSQL is not a tool, instead it is a methodology that is composed of several complementary and competing tools [27]. The NoSQL database concept follows four different kinds of strategies for storing data in a non-relational database. The table below shows the details of each NoSQL strategies.

NoSQL Database type	Strategy	Example
Key-Value pair	Conceptual distributed dictionaries of key and value	Riak
Document-Oriented	A Key-Value pair concept with a functionality of finding documents based on their content and a write/read by a key	MongoDB
Column-Family	The data is stored in cells grouped in columns, and the columns are logically grouped into families of the columns.	HBase, Hypertable
Graph-Oriented	It models the data in the form of graph and the nodes of a graph depict the entities while the relationships depict the association of the nodes.	Neo4J, GraphBase

Table 2.4 Different kinds of NoSQL strategies

2.5. Related Work

Concerning Database migration, in a cloud computing environment- it could be from a data center or another cloud environment source varies researches have been conducted in academic and IT companies [23] [33]. The researchers' main emphasis raised on migrating SQL database to NoSQL database and from no SQL to no SQL Databases. On the other hand, some scholars were concerned with its security mechanism that should be exerted during the migration process. However, non-have considered a back to back migration, meaning migrating a data from SQL based source to no SQL based target and vice versa. And, from the performance and speed of the migration process, less emphasis is given. Besides, some researchers have devoted extensively on database migration and its migration process performance where we have keen to build upon.

Likewise, [34] has proposed an approach of model transformation and data migration from the relational database to MongoDB and thereby developed a tool for demonstration. The research considers the query characteristics and data characteristics of the relational database to design

the model transformation algorithm based on description and action tags. Moreover, the research has pioneered with its kickoff contribution, an automatic model of data migration based on the model information. The research has necessitate steps like mapping the description and action tags from relational database to overcome the limitation of relational database and coined out Frequent Joint tag, Big Size, Frequent Modify tag and Frequent insert tag from relational database log for the description tag whereas, Embed Child entity, Embed Parent entity and reference pertaining action tags. And then, a model transformation algorithm is performed on the ER model by mapping the concept of ER model to MongoDB. From the aforementioned information, authors developed transformational algorithm 1 in which ER model transformed into MongoDB physical model with time complexity of O(n+m) where n and m represent the number of entities and relationships, respectively. Beside Algorithm 1, authors also developed the algorithm of action tags from description tag and the relationship of ER model with time complexity of O(m) where m represents relationships. And finally, they proposed the migration algorithm based on transformed MongoDB model. The proposed work is set to experiment with TPC-H dataset and proved its effectiveness on reading query. However, the model lack to consider other dimensions like migrating data from other no SQL database source to MongoDB as well as from MongoDB to SQL database. In addition to this, they did not cover the speed of the migration process. Thus, this research work aims to extend by working on top of the aforementioned features.

On the other hand, some researchers have devoted to contributing a data migration design pattern from which one database family in no SQL database is transformed into another database family. For instance, [35] took initiative in providing a design pattern that enables portability between column family databases as Neo4j with graph database as H-base. The design pattern has provided 11 recipes for mapping the Neo4j database to the HBase database. Whereas, they also provided a reverse migration i.e. from HBase to Neo4j design pattern is provided with 8 recipes. And then, the proposed design pattern is evaluated with a simple suitable scenario on Health Infoscape dataset. From this, they have coined out that migrating small size data with a complex domain to a graph database resulted in a facilitated analyzed data whereas, using the column database provided a smooth way for maintaining and analyzing data in a large size. Nevertheless, the research did not focus on SQL based database sources

and other no-SQL database families. Moreover, they did not provide a prototype and not raised a performance issue of the migration process. And this is where it leaves a room for this research on which the limitations are built upon.

On the other way, researchers such as in [11] have introduced a broker called CIB at the SaaS layer of cloud computing. The broker aimed to mitigate the vendor-lock-in problem that would cause a problem for migrating data from one cloud vendor to the other and vice versa. To realize the specified broker the following consecutive steps incurred; collecting and analyzing of a meta-data, developing the mapping model, designing, implementing and finally testing the solution were introduced as a methodology for the solution. Then, the proposed model evaluated with a real enterprise dataset with a case study by compromising all the specified methodologies. And then, authors have demonstrated as the proposed work fit to fill the interoperability gap between two different vendors by enabling a data migration easy and straightforward. However, they did not evaluate the broker in relation to different aspects, such as performance and time. In addition to this, they did not provide an orthogonal migration process and did not cover the migration process speed, in general. Thus, we have based to build on the aforementioned limitations.

Similarly, [3] has adopted an SSL protocol to secure data migration between clouds using third party audition (TPA). Likewise, they also keen to improve the time constraint parameter for data migration. The proposed work has considered the data storage and sharing service with three entities viz. USER, Third Party Auditor, and Cloud. Having the specified entities, they suggested 11 migration process steps ranging from USER'S migration request with SSL protocol supervision to the point where the source data node sends the blocks to the target node. And then the proposed work performance is analyzed on behalf of metrics like block encryption and decryption phase on a network simulator tool NS-2 and compared the result with MDM. The result has demonstrated as TPA embedded with SSL protocol resulted in a better time constraint. However, the proposed word did not consider the parallel processing of data migration and other aspects like processing speed as well as the research lack to demonstrate beyond the simulation tool. Although authors have gone far to excel the security and performance of data migration between two cloud vendors, they lack to consider data migration from other data storage services

On top of aforementioned research works, this research work extends the process of data migration in a cloud computing environment by giving due attention on varies dimensions sources and target of migration, and boost data migration speed by introducing multithreading concept with varies techniques.

2.6. Summary

This chapter discussed the current state-of-the-art database migration tools and techniques, cloud computing architectures and frameworks, service delivery and deployment models, characteristics, advantages and limitation. And related works are identified. This will help the research to have inclusive view about cloud computing environment and database migration process. Then, how the research work is done and the steps followed in this research is discussed on the next chapter.

Chapter Three

3. Research Methodology

3.1. Overview

A research methodology refers to how and what steps are followed to answer the fundamental research questions. On this step of the study, what type of research methods are going to be used, method of searching and evaluating literature, the type of data to be used and the way of conducting an experiment, and the mathematical model to evaluate the performance the proposed algorithm is included. In academic research, the methodology is an important tool to achieve the desired goal of a particular study.

Establishing and answering the research questions, and thereby achieving the research objectives requires a sound methodology, which is a process and roadmap that ranges from problem formulation to evaluation. So, for the achievement of the general and specific objectives, an Experimental Research Design Method is used in the lifespan of the research work. This is because, the underlined work needs an experiment for the evaluation of the expected result and it also provides space for acceptance criterion, that we take as a reason to consider Experimental Research Design Method fit and realize the proposed work. According to the selected research method, the following seven steps are considered and the diagram below describes what and how the steps are followed for the research work.

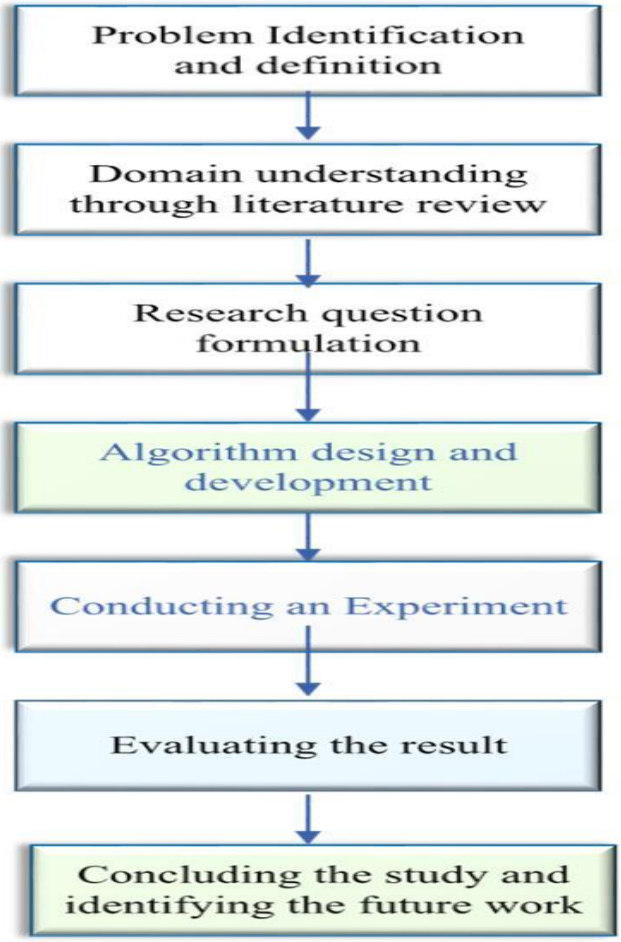

Figure 3.1 Flow of the research method

According to the diagram above, the research passes through seven fundamental steps: problem identification and definition, domain understanding through literature review, research question formulation, algorithm design and development, conducting an experiment, result evaluation, conclusion and future work identification. Because of the nature of the study, an experimental research design method is selected. That means, the main idea of this research is to design a data migration algorithm for migrating data from the relational database to document-oriented database. So, the proposed algorithm should be evaluated with the result of the experiment according to the identified performance measures.

3.2. Problem identification and definition

This stage is about identification of a specific problem and defining the identified problem with a well-organized way. In the very beginning, the problem to be solved is identified and defined. The issue of interoperability and portability among different cloud providers leads to vendor lock-in problem on cloud service subscribers and the main cause of this problem is the lack of sufficient data portability and migration mechanisms. So, the identified issue is the main problem what this research tries to solve.

3.3. Domain understanding through literature review

In the second stage of the study, literature review is used to understand research domain and to identify a gap in previous works. So, related literatures from different kinds of sources such as journal articles, books, topic related blogs and conference papers had reviewed to understand the problem domain deeply. The selected journals are

- IEEE Explore
- Google Scholar
- ScienceDirect
- Elsevier,
- IJIRSET
- Online blogs

3.3.1. Literature Searching method

The mechanism used to search the literature follows the research question to be answered at the end of the thesis. In addition to the research questions, general queries such as overview of cloud computing and overview of database migration are used. The query strings used to search literatures are:

- Cloud computing deployment models, cloud computing service delivery models,

- Document-oriented database vs Relational database,
- Data migration algorithm performance factors,
- Comparation of programming languages based on accessing database,
- Reasons to migrate database,
- Database migration service providers,
- Data transformation technologies, XML vs JSON
- Algorithm performance criteria

Based on the searching criteria listed above, lots of documents are downloaded. But some filtering mechanism is used for identifying important data. The filtering criteria includes year of publication, type of document, number of citation and the journal where the paper is published. According to these criteria: year before of 2010, document type of tutorial and presentation slide, number of citations below 100 and articles from unfamiliar journal sites are discarded.

3.4. Formulation of research question

The questions to be answered by the study and inferring the consequence are the tasks in this phase. The final objective of the study is answering the questions formulated in this phase. The research expected to answer:

- The reason how a document-oriented database (NoSQL) is better than relational database for big data management,
- Better programming language of implementation and data interchange format for database migration algorithms and
- The way of implementing a multi-threaded application for database migration

3.5. Algorithm design and development

Designing and developing an efficient Relational to Document-Oriented database migration algorithm is the main task of this step. In addition to that, identification of performance factors and evaluation criteria for the proposed algorithm are the other important tasks. To implement the proposed algorithm, an appropriate programming language will be selected based on different evaluation attributes.

In algorithm design and development phase, programming language selection with different criteria, data interchange technology criteria, and space and time complexity analysis will be conducted. Then, the identified space and complexity analysis result will be used as an input for comparing previous works.

3.6. Conducting an experiment

In the previous steps, different inputs such as space and time complexity analysis result, programming language to be used, data interchange technology to be used and performance factors to consider are identified for the experiment. Based on the identified algorithm performance factors, there is an experiment on the data migration algorithm developed in the previous stage. The result of the experiment is the final output to evaluate the finding of the thesis. Therefore, according to the experimental result the whole study will be evaluated.

3.7. Evaluating the findings

On this stage, the result of the experiment is evaluated according to the evaluation criteria of the algorithm. The performance factors, measuring attributes and the result of the previous works are used to evaluate the finding.

3.8. Concluding the result and identifying future works

Finally, the thesis will be concluded based on the evaluation of the identified result. Then, the future works will be identified and a recommendation is proposed for other researchers.

3.9. Summary

In this chapter, the methodology to be used through the research work is defined. And Experimental Research Design Method is used due to the nature of the selected research problem. The next chapter discussed the proposed database migration algorithm in detail.

Chapter Four

4. Proposed Database Migration Algorithm

4.1. Overview

The objective of this chapter is to have a clear picture of the existing scenario in data migration solutions focusing on the migration of database from relational database systems to a document-oriented database system and the proposed data migration algorithm. The availability of effective data migration solution and the interoperability feature on the business of cloud computing ecosystem provides great freedom of using cloud services for the service subscribers.

The primary source of information for this research is obtained through structured experimentation and evaluation on the available database migration solutions. And also, the secondary data source like thesis, journal, books, reports, conference articles and white paper from the websites of reliable authors and organizations have been used to get information about database migration in the cloud computing environment.

The general structure of the proposed solution is described in figure 4.1. The solution has two main parts, the frontend and the backend (REST API). As discussed in the data interchange technology section, the data interchange technology that provides a data integration service between the source platform and target platform is JSON.

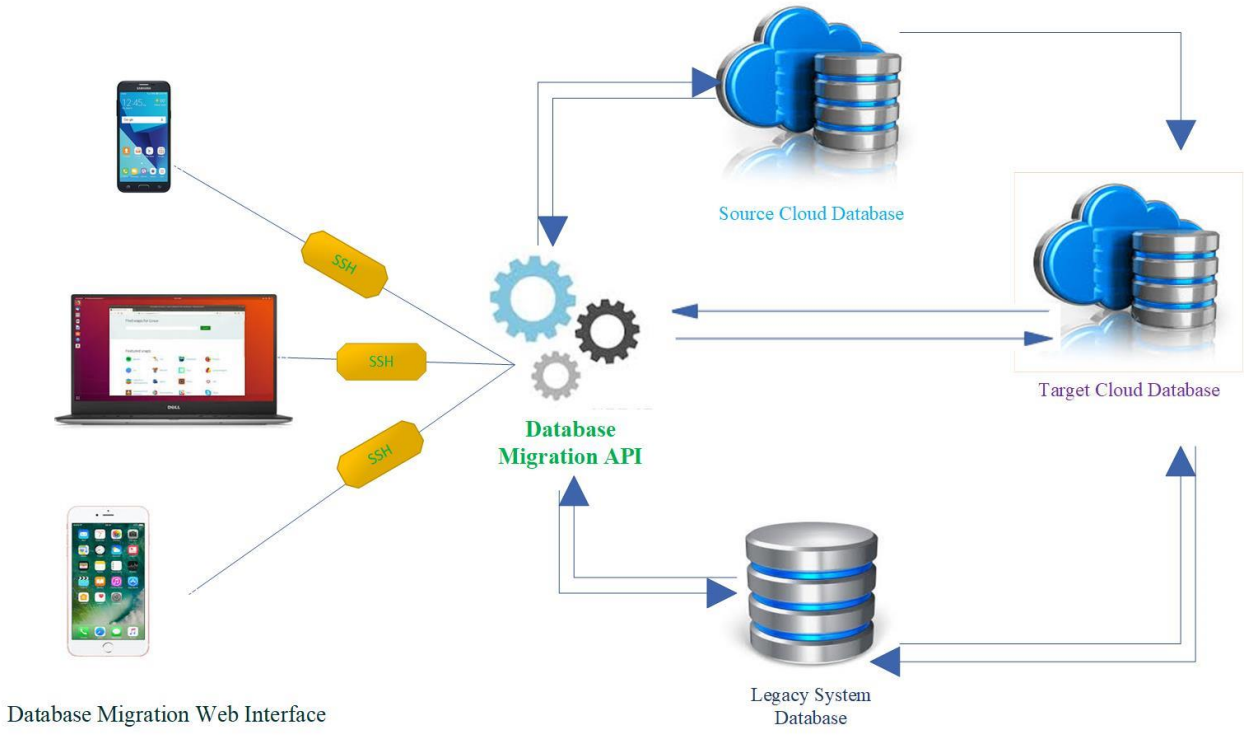

Figure 4.1 Architecture of the proposed database migration solution

The proposed solution allows migrating database from two kinds of data sources. The first source is from a legacy database system. In this case, the proposed solution can migrate the database from the obsolete database technology used by the institution. And the second one is from the cloud database; in this case, the user has already hosted the database to the cloud and having a plan to migrate into the other cloud service provider due to different reasons. The best feature of the proposed solution is, the users can use the thick client as well as a thin client to migrate a database from source to target.

4.2. Structure of the proposed solution

The proposed database migration solution has two main parts: the frontend and the backend (REST API). The responsibility of the frontend is creating an interface for the user and forwarding command and parameters to the backend as well as showing the status of the migration process for the user. The main component of the system is the backend (REST API)

module, which is responsible for connecting source and target, metadata extractions and mapping, target data formator, data compressor/decompressor and data importer.

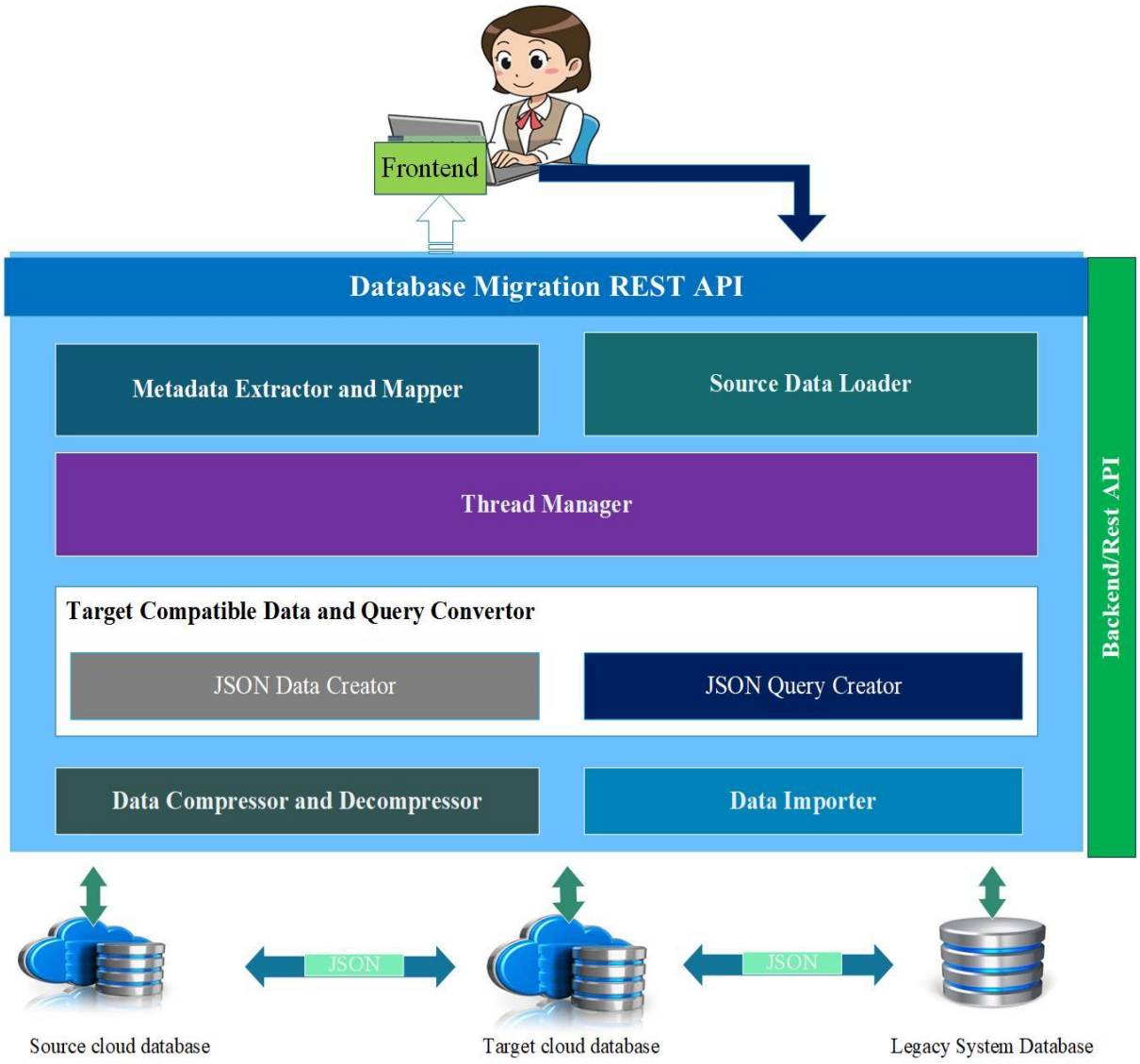

Figure 4.2 The detailed structure of the proposed database migration solution

According to figure 4.2, there are five main modules and two submodules.

i. Metadata extractor and mapper: the task of this module is to extract metadata such as schema definition from the source database and mapping it to the target database based on the destination database platform.

ii. Source data loader: is responsible for loading data from the source database and passing it to the target compatible data and query converter module.

iii. Target compatible data and query converter: is one of the main modules intended to convert source data and query into target compatible format. This module has two main modules namely, the JSON data creator for converting relational data into a JSON format and JSON query creator for creating target compatible query.

iv. Data compressor and decompressor: for compressing JSON data on the source environment and decompressing at the target environment. This module maximizes the performance and speed of the database migration process.

v. Data Importer: is responsible for importing the decompressed data into the target database.

vi. Thread manager: manages all thread related tasks all over the data migration process such as creating a thread, assigning a task to a thread and destroying a thread.

4.3. Database migration tool selection criteria

In the database migration process, performance is the main concern. According to a report by Oracle, up to 75 percent of new systems fail to meet expectations because of errors in the migration process result in data that is not adequately validated for the intended task [23]. In this study, five basic criteria are identified that should be considered when planning to migrate a database with a database migration tool:

- **Security**: the movement of data from a source platform to the target platform should be secured.
- **Performance**: indicates, how much the migration tool is good
- **Model transformation**: schema of the source database should be transformed into the schema of the target platform completely and correctly.
- **Migration cost**: indicates that, how much it cost to migrate a database
- **Data quality**: there should no loss of data and metadata in pre-migration, during migration and post-migration of the database.

- **Types of databases the DMT supports**: the tool should support both the source and target database platforms.

4.4. Fundamental stages of the database migration algorithm

Based on the nature of the tasks through the database migration process, the activities should be grouped according to its purpose. So, in a database migration process, there are three important stages: pre-migration, during migration and post-migration.

4.4.1. Pre-migration

During this stage, the data that will be migrated are selected based on the business, project, and technical requirements and dependencies [24]. For the migration of the database, identifying the source and target platform, connecting source and target database are the tasks in a pre-migration stage. And finally, in a pre-migration stage the DMT block any modification of the database to keep the source and target database is synced.

4.4.2. During-migration

After the completion of the pre-migration stage, the DMT will start the actual database migration process. This stage has two sub-tasks: the schema migration and data migration. In schema migration stage, loading source schema information and mapping the result to the target database will be done.

The main responsibility the actual database migration process is the migration of actual records into the target platform.

4.4.3. Post-migration

In this step, the main task of the DMT is to check the success of database migration process. The number of tables and records in a source and target database is the fundamental measure of successful data migration. So, counting the number of tables and records on both the source and target database can ensure the quantitative success of the migration. And finally, the database on the source cloud can be deleted based on the need of the user.

4.5. Programming language of implementation

One of the characteristics of a good algorithm is the feature of language independence. That means an algorithm can be implemented by any programming language. But, the performance of the solution is different from one programming language to the other. For the implementation of the proposed database migration algorithm, three programming languages (Java, C#, and Python) are identified. From these languages, the best is selected based on the identified criteria. And the criteria to select the best language of implementation are:

- Platform independence
- Support for different kinds of database driver
- Reusability of query
- Webservice support and
- Support for multi-threading
- Support for SSL protocol

Language Selection Parameter	**Language**		
	Java	***C#***	***Python***
Platform independence	Yes	No	Yes
Support for multi-threading	Yes	Yes	Yes
Support for multiple database drivers	Yes	Yes	Yes
Reusability of query and syntax	Yes	No	No
Web service support	Yes	Yes	Yes
SSL protocol support	Yes	Yes	Yes

Table 4.1 Programming language of implementation selection criteria

Platform dependency: The ability of a program to run on any platform is the platform independence feature. Java and Python support this feature and C# is a platform dependent

language. As a requirement, the database migration tool should be hosted on any kinds of platform.

Support for multiple database drivers: To migrate data from one database to the other, the programming language of implementation should have a driver for both the source and the target database. In this case, all three languages have a database driver for MS SQL and MongoDB.

Reusability of query and syntax: query reusability is the ability of a programming language that allows running a common query and syntax for different kinds of database. Java has a good feature of using common query and syntax for different kinds of databases. For example, Java uses the same query and syntax for MS SQL, MySQL, PostgreSQL, MS Access and SQLite. But, the other programming languages have not such kind of feature.

Table 4.2 shows a simple assessment that evaluates the reusability of query and syntax in Java, C# and Python for MySQL, PostgreSQL and MS SQL relational database technologies. The assessment takes a database Connection object for all the languages.

Language	**Database**		
	MS SQL	***PostgrSQL***	***MySQL***
Java	Connection con;	Connection con;	Connection con;
C#	SqlConnection con;	NpgsqlConnection con;	MySqlConnection con;
Python	con = pymssql.connect()	con = psycopg2.connect()	con = MySQLdb.connect()

Table 4.2 Query and syntax reusability comparison of three programming languages

As the assessment shows that, C# and Python have a different syntax for initializing a database connection object. And the same is true for all database objects. For example, C# uses SQLCommand object for a MS SQL and MySQLCommand for MySQL to run a query on a database. But Java uses the same syntax for most of the database technologies.

Support for web service: To build a cloud computing based applications, the programming language should support the development of a web service. So, all three programming languages support web service development.

Multithreading support: As discussed on the objective of this study, the proposed algorithm uses a multithreading concept to maximize the speed of database migration by dividing the data migration task into multiple threads.

SSL protocol support: to establish an encrypted link between the frontend and the database migration API, an SSL security protocol is proposed. So, the standard security technology for establishing an encrypted link should be supported by the language.

So based on the above criteria, Java is the best programming language of implementation for the proposed database migration solution.

4.6. Data interchange format

The two widely used data interchange formats, XML and JSON are compared to apply the best on the proposed solution. Based on the selected comparison parameters below, JSON is the choice format. According to a study by B. Lin and Y. Chen, figure 4.3 and 4.4 indicates the efficiency and transmission time between XML and JSON.

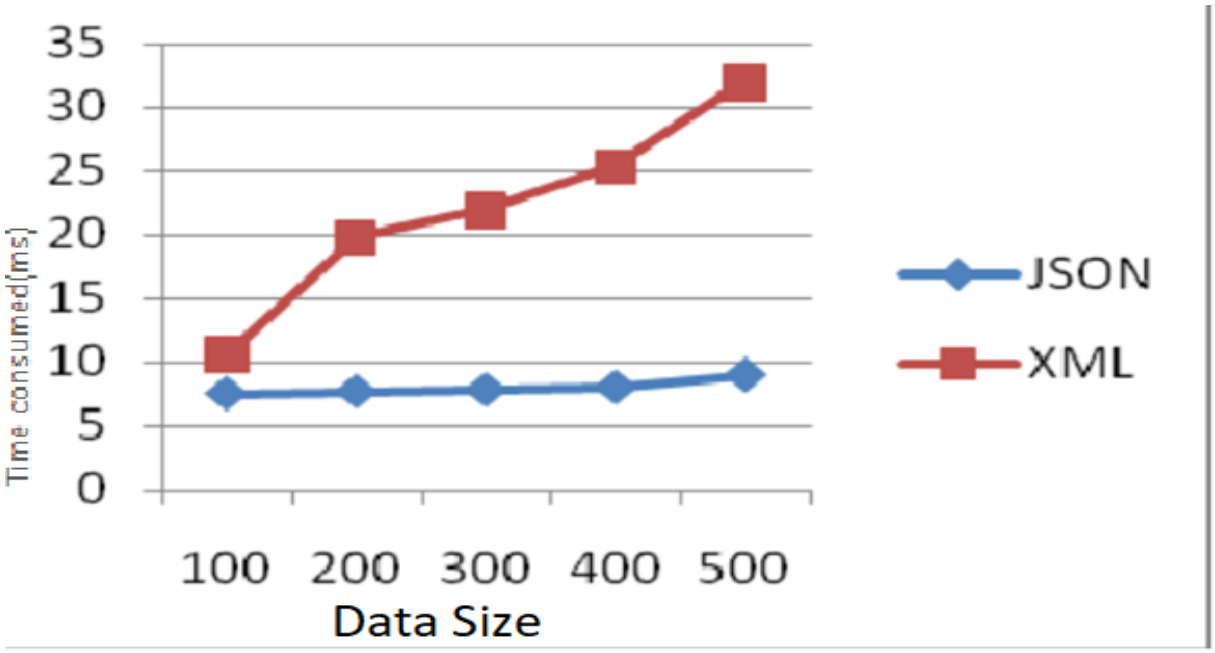

Figure 4.3 Comparison of efficiency between JSON and XML

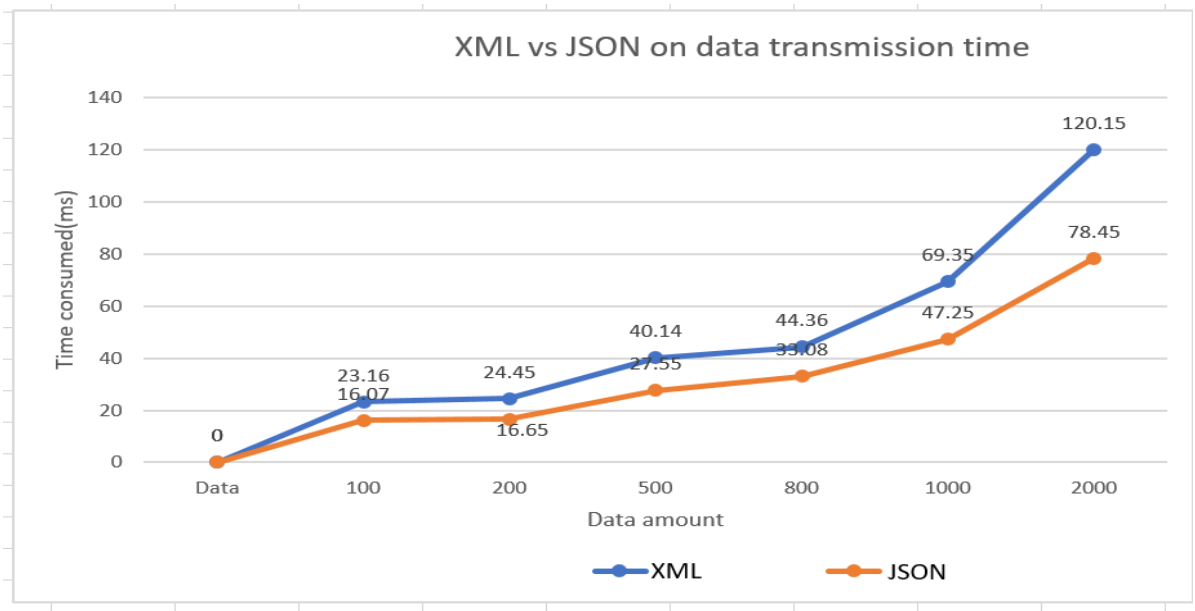

Figure 4.4 Comparison of data transmission time between JSON and XML

The encoding below shows how XML describe the data elements.

```xml
<?xml version="1.0" encoding="utf-8"?>
<countries>
  <country>
    <ID>1</ID>
    <Name>Ethiopia</Name>
  </country>
  <country>
    <ID>2</ID>
    <Name>Eritria</Name>
  </country>
  <country>
    <ID>3</ID>
    <Name>Djibuti</Name>
  </country>
</countries>
```

And JSON follows the following data element description format for the same document.

```json
[
  {
    "ID": 1,
    "Name": "Ethiopia"
  },
  {
    "ID": 2,
    "Name": "Ertirea"
  },
  {
    "ID": 3,
    "Name": "Djibuti"
  }
]
```

The other important parameter to compare XML and JSON is the document format of MongoDB. In MongoDB data is stored and queried in BSON document format [28]. And

BSON is the advanced form of JSON. The query below is a simple insert query of MongoDB to insert two mobile products into products collection.

```
>db.products.insert({"_id": "12345",
  "product_name": "Apple iPhone",
  "category": "mobiles"
})
>db.products.insert({"_id": "12346",
  "product_name": "Samsung S3",
  "category": "mobiles"
})
```

So, based on the comparison result, JSON is selected as a data interchange format of the proposed algorithm.

4.7. Schema construction and migration

In a relational database, schema refers to the organization of data as a blueprint of how the database is constructed [31]. Schema migration could be the first task if the data migration process is for the first time. One of the modules on the proposed database migration algorithm is a schema construction and migration module. The following algorithm is designed to construct a schema based on the structure of the source database through a multi-threading approach.

Algorithm 1 *Schema construction algorthsm*

Input: *source database connection string, srcConnString*

Output: *List of MongoDB collections*

```
1: procedure buildSchema(srcConnString)
2:     schemaList ← srcConnString.getAllSchemas( )
3:     collectionList ←new JSONObject
4:     tempObj ←new JSONObject
5:     foreach schema in schemaList
6:         listOfTables ← schema.getAllTables
7:         foreach table in listOfTables
8:             tempObj. constraint ← table.constraint
9:             tempObj columns ← table.columns
10            collectionList.put(table.name, tempObj)
11:        end foreach
12:        clear tempObj
13:    end foreach
14:    return collectionList
15:    end procedure
```

4.8. Implementation of multi-threading in a database migration

The importance of implementing a thread is to increase the performance of a process by separating a task into multiple threads. Running several threads is similar to running several different programs concurrently, but with the following benefits

Multiple threads within a process can share the same data space and can, therefore:

- Threads can share information, one thread can read, write or change another thread's data
- Communication with each other more easily than if they were a separate process and
- Threads do not require many resources

The proposed solution uses a multi-threaded program to minimize the time required to migrate a database from one platform to the other. Based on the predefined rules, a thread generated and a task is assigned for the thread. To generate a thread and assign a task, a relational database is grouped into three main groups based on the number of schemas, relation, and record-count in single relation.

Case 1: A database with multiple schema and multiple tables

Case 2: A database with a single schema and multiple tables

Case 3: A database with single schema and single relations but with a big size of records

For all the three cases, multiple threads can be generated and a database migration task is assigned to them.

Case	**Action to be taken**
Case 1	Thread per schema is generated
Case 2	Thread per relation is generated
Case 3	Thread per 100,000 rows is generated

Table 4.3 Number of threads per schema, table, and rows

Case 1:

By assuming that there are S numbers of schemas, the time t required to migrate an average of R numbers of rows from T numbers of relations, in a single threaded application the time taken to migrate the data can be calculated as:

$$t = S \times R \times T$$

Equation 1 Time requirement in a single-threaded application to migrate R number of records from T number of tables with S number of schemas

By applying a thread per schema, the time required to migrate the same size of records become:

$$t = \frac{R \times T}{S}$$

Equation 2 Time requirement in a multi-threaded application to migrate R number of records from T number of tables with S number of schemas

Case 2:

In a single threaded application, the time required to migrate R numbers of rows from T numbers of relations can be calculated as:

$$t = R \times T$$

Equation 3 Time requirement in a single threaded application to migrate R number of records from T number of tables

In a multi-threaded application, the time required to migrate R numbers of rows from T numbers of relations can be calculated as:

$$t = \frac{R}{T}$$

Equation 4 Time requirement in a multi-threaded application to migrate R number of records from T number of tables

Case 3:

In a single threaded application, the time required to migrate R numbers of rows from a single relation can be calculated as:

$$t = R$$

Equation 5 Time requirement in a single-threaded application to migrate R number of records from a single table

But, in a multi-threaded application the time required to migrate R numbers of 10,000×N rows from a relation can be calculated as:

$$t = \frac{R}{N}$$

Equation 6 Time requirement in a multi-threaded application to migrate R number of records from a single table

The following pseudo code creates a dynamic thread based on the above three cases.

Algorithm 2: Database migration task assignment per thread

Input: *SCHEMA_COUNT and TABLE_COUNT*

Output: *List of threads with assigned task*

1: **procedure** migrateData()

2: *IF (SCHEMA_COUNT > 1)*

3: *Foreach Schema sh*

4: *Generate new Thread* tr_i

5: tr_i ←*assignSchema(sh)*

6: tr_i*.doMigration()*

7: *End Foreach*

8: *END IF*

9: *ELSE IF (TABLE_COUNT > 1)*

10: *FOREACH Table* t_i

11: *Generate new Thread* tr_i

12: tr_i←t_i

13: tr_i*.doMigration()*

14: *END FOREACH*

15: *END ELSE IF*

16: *ELSE*

17: *TOTAL_THREAD ←total_row/100,000*

18: *IF (TOTAL_THREAD <1)*

19: *main_thread ←total_row*

20: *main_thread.doMigration()*

21: *END IF*

22: *ELSE*

23: *Current_row ← 1*

24: *FOR (i=1 to TOTAL_THREAD)*

25: *Generate new Thread tri*

26: *FOR (j= Current_row to total_row)*

27: *IF (Current_row+100,000<=total_row)*

28: *tr_i ← Current_row to Current_row+100,000*

29: *END IF*

30: *ELSE*

31: *tri ←Current_row to total_row*

32: *break from loop*

33: *END ELSE*

34: *Current_row ←Current_row+1*

35: *tr_i.doMigration()*

36: *END FOR*

37: *END FOR*

38: *END ELSE*

39: *END procedure*

4.8.1. Responsibility of threads

As defined by the database migration task per thread algorithm, each of the randomly generated thread has eight tasks, four on the source platform and four on the target platform. From the source platform side: creating a JSON file, conversion of record to JSON data format and writing it to a JSON file, compressing the file and sending the compressed file via the SSL tunnel. And on the target platform side: it receives the compressed file, decompressing the data, reading data from JSON file and inserting it to MongoDB and finally it deletes the received JSON file. The deletion of the received file is to make the data secure and to minimize the storage space requirement on the target platform.

After the completion of the migration process, the user is requested for the deletion of a relational database. If the user has a plan to stay more with the previous cloud provider or with the existing system, then the DMT leaves the database as it was. But, if the user confirmed the deletion of the previous relational database, then the migration tool can delete the relational database fully.

Figure 4.3 shows how each thread works in the proposed database migration solution. After the completion of the migration task assigned to randomly generated thread, the thread destroys itself.

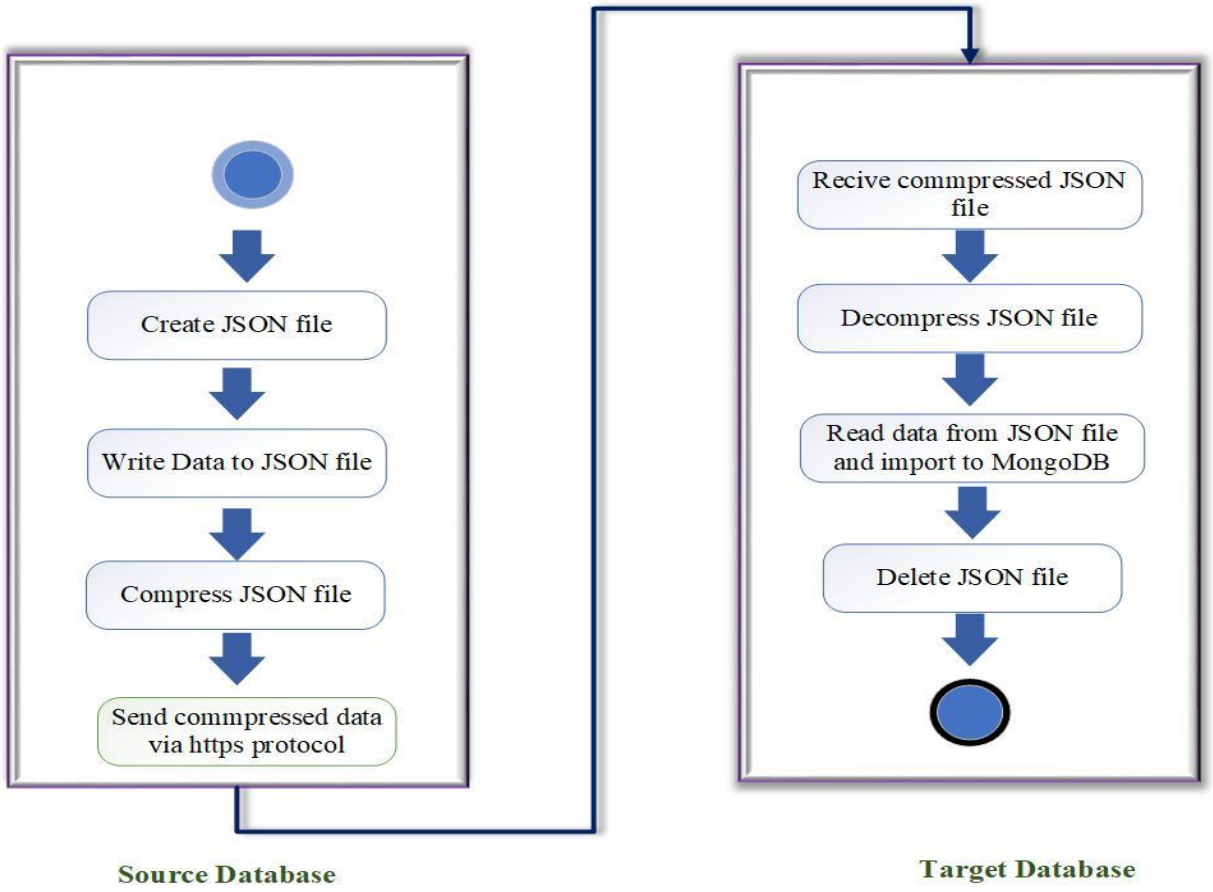

Figure 4.5 Responsibility of thread in database migration

4.9. Construction of MongoDB document from row of relational database

After a successful construction of schema on MongoDB, the next step is moving the actual data record from relational database to the target database. To do the migration, the data in a structured/tabular format should be converted to a document format. So, the following algorithm is responsible to convert the tabular data into a MongoDB document.

Algorithm 3 *MongoDB document construction*

Input: *Relational database record ResultSet*

Output: *List of MongoDB Documents*

```
1: procedure doMigration(recordResultSet)
2:     columCount ← recordResultSet.MetaData.columnCount( )
3:     temDocument ←new JSONObject
4:     documentList ←new Array of JSONObject
5:     while recordResultSet.next() ≠ false do
6:         for i← 0 to  columCount
               temDocument.put(recordResultSet.MetaData.getColumnName,
7:             recordResultSet.MetaData.getObject )
8:         end for
9:         documenList.put(temDocument)
10         clear temDocument
11:    end while
12:        clear tempObj
13:    end foreach
```

14: *return documenList*

15: **end procedure**

Each dynamically generated thread executes the above procedure according to the task assigned to it in section 4.8.

4.10. Join operation in MongoDB

In a relational database, join is the process of combining columns from one or more tables. But, in MongoDB, the join feature is not available. In MongoDB, there are two mechanisms to combine columns: embedding and referencing. Embedding is a way of inserting one document into the other document. Whereas referencing is the method of linking a document by using the **_id** field of the other document. The drawback of embedding a document is that it has a high risk of duplicated documents.

In a relational database, the normalization strategy minimizes the problem of data duplication and increases the data integrity and consistency. But, due to the absence of normalization in MongoDB, the chance of duplicate data is high. Because, one of the mechanisms to relate documents in MongoDB is through embedding child document/referenced document into a parent document.

To minimize such kind problem, the proposed algorithm has its own constraint extractor module and that allows to extract the primary key and foreign key of a relational database. Based on the extracted keys, the JSON data builder of the algorithm creates a link between the two documents. And the link is built based on their previous relation on the source database. Therefore, the referencing strategy is used on the proposed solution to minimize document duplication.

For example, the snapshot below shows two different documents: patron and address with a one-to-many relationship, by referencing or linking [29].

```
{
    _id: "joe",
    name: "Joe Bookreader"
}

{
    patron_id: "joe",
    street: "123 Fake Street",
    city: "Faketon",
    state: "MA",
    zip: "12345"
}
```

Then, when embedding the address document into the patron document, the combination of the two documents can be created as follows.

```
{
    _id: "joe",
    name: "Joe Bookreader",
    address: {
                street: "123 Fake Street",
                city: "Faketon",
                state: "MA",
                zip: "12345"
              }
}
```

In the case of embedding document, one document can be embedded multiple times based on the relationship between the two documents and that leads to replication of data in a database. So, to minimize data duplication and storage requirement, the referencing mechanism is a good approach.

The proposed solution constructs a link between two documents based on their previous relation on a relational database. Therefore, referencing method is used to join different MongoDB documents.

Chapter Five

5. Experiment and Result Analysis

5.1. Overview

The main objective of the research work was designing an algorithm to migrate a relational database from one cloud provider to the other with a high migration speed. A simple case study was conducted to realize and clarify the importance of the proposed algorithm from a real-world perspective.

The proposed solution is designed based on four different principles: multi-threading, JSON data format, data compression and SSL protocol. That ensures performance, data confidentiality and integrity for the database migration algorithm. The development environment, programming language, dataset, data interchange format, and database management system used for the experiment of the proposed algorithm is described in this chapter. The prototype for the thread management and schema migration component of the algorithm is discussed with a support of screenshots based on the corresponding results.

The result of the proposed solution is equated with the database migration tools provided by the top three cloud providers (Amazon, Microsoft and Google) and also the strength and limitation are identified.

5.2. Description of the dataset

In this study, the well know Microsoft's experimental dataset, AdventureWorks2017 is used. In addition to the collection of the datasets, an experiment has been conducted, with an experimental setup as shown in Table 5.1. The experimental dataset has a total of 6 schemas, 70 relations and 567,561 records. The database migration is tested in a single threaded method and in a multi-threaded method to evaluate the performance of the proposed solution. For a data migration task an ETL function is required to see the extracted data from various sources being fixed and then loaded into the target database platform [32].

5.3. Experimental setup and tools

An experiment has been conducted with AngularJS framework as a frontend, Java REST API as a backend, Microsoft SQL as a source database, MongoDB as a target database, JSON as a data interchange format, IntelliJ IDEA 2018.3 as an IDE for both backend and frontend on the experimental environment is configured as illustrated below in Table 5.1. The reason to select Java as a language of implementation and JSON as a data interchange language is defined in section 4.5 and section 4.5 respectively. To use the Microsoft's AdventureWorks2017 open source database, Microsoft SQL Server is used as a relational database.

Tool	**Specification**	
Computer	Brand	HP
	Processor	Core i5, 7th Gen
	HDD	250GB
	RAM	8GB
	Processor speed	2.5GHz, 2712MHz
Operating System	Windows 10, 64bits	
Backend	Java RESTful API with Jersey	
Frontend	HTML5, Angularjs and bootstrap	
Source database	MS SQL Server	
Target database	MongoDB	
IDEs	IntelliJ IDEA 2018.3 for frontend and Eclipse IDE for backend	
Dependency management	Maven for backend and bower for frontend	

Table 5.1 Experimental setup and tools

5.4. Prototype Implementation

As a discussion below, the prototype is divided into two main parts: the frontend and the backend.

5.4.1. Frontend

The frontend of the prototype is implemented in HTML5, Angularjs and bootstrap. With the responsive feature of bootstrap, the frontend can be accessed by any client device and that makes the proposed solution platform independent. The Angularjs framework allows the frontend to easily communicate with the backend by its AJAX capability.

The frontend of the prototype has the source and target database information page. These forms provide an input functionality to build the connection string for both source and target databases. The figure below shows the connection details of the source database. So that, all relations and database records are migrated from **AdventureWorks2017** of SQL Server (DESKTOP-IE77J90H\LOCALHOST) with a default SQL server port 1433 are migrated to **AdventureWorks** of MongoDB (127.0.0.1) on port number 27017.

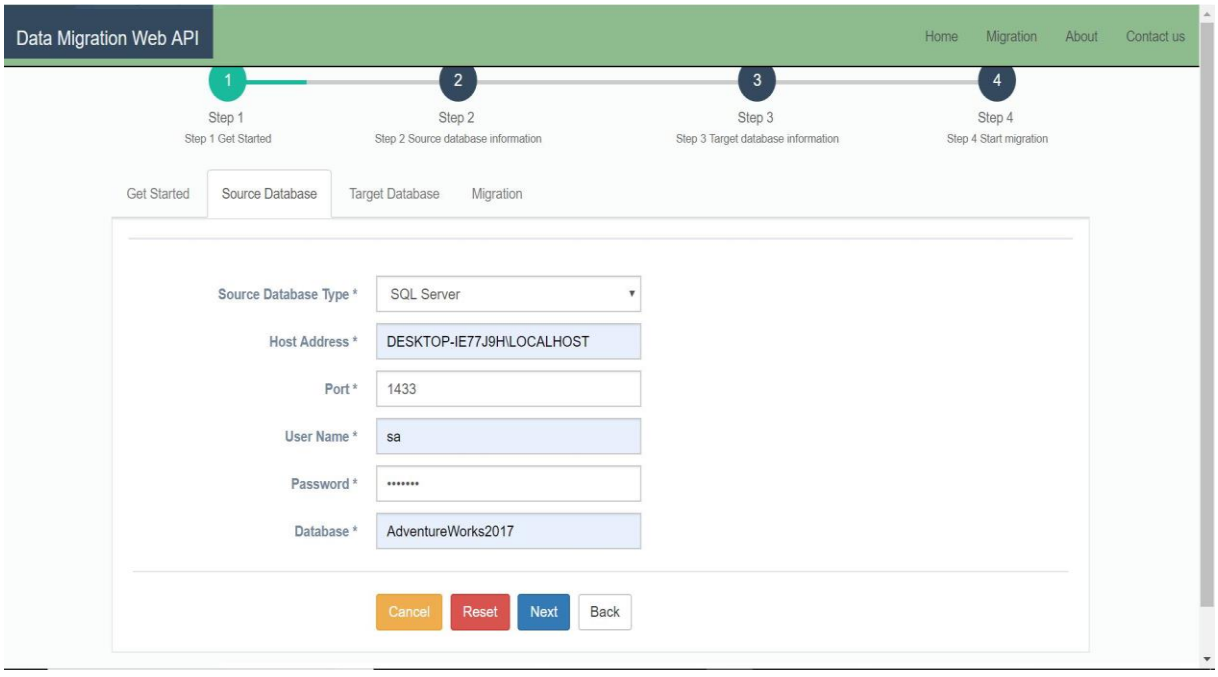

Figure 5.1 Source database information page

Both the source and target database information form allow the user to select database type, host address, port, user name, password and name of database to be migrated. Figure 5.2 shows that, the target database connection information form for a target database.

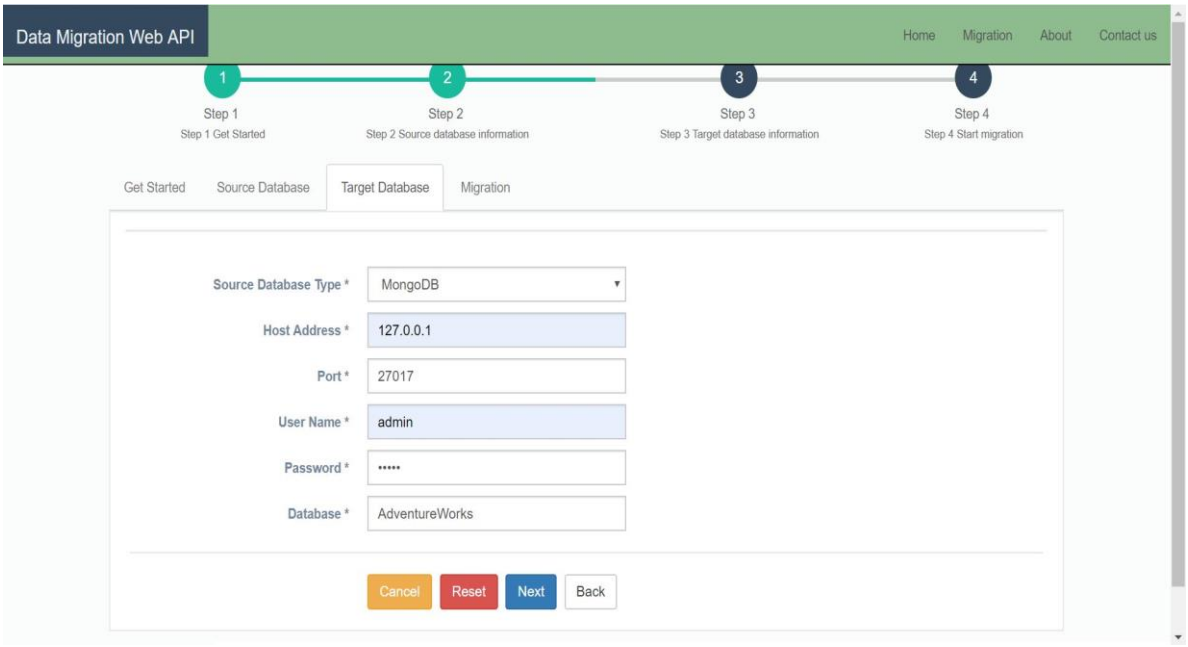

Figure 5.2 Target database information page

After the source and destination database information is selected, the user needs to decide to delete the source database after a successful completion of database migration. If the user agreed to delete the source database, then the relational database can be permanently deleted from the source platform. In this step, as shown in figure 5.3 the user is confirmed about the deletion of the source database as well as the beginning of the migration process.

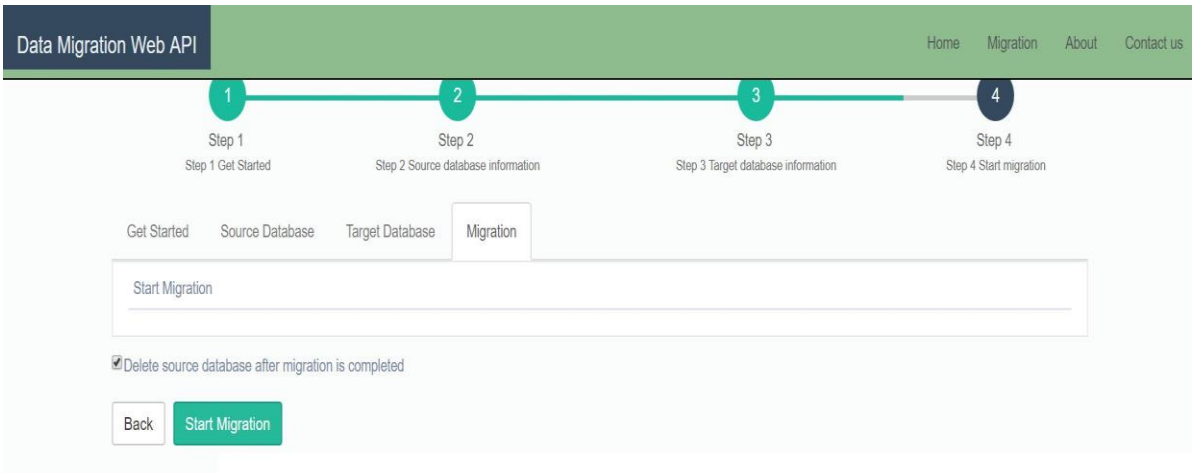

Figure 5.3 Migration starter with source database deletion page

And finally, after the database migration process is started, the frontend provides a page that shows the migration progress and an option to cancel the database migration.

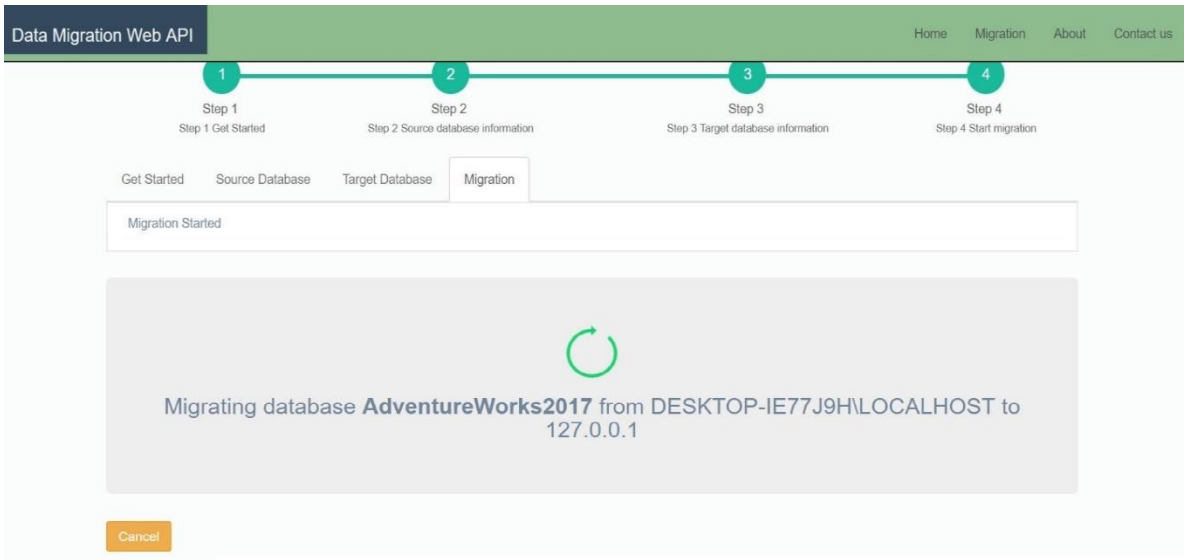

Figure 5.4 Database migration progress and cancelation page

5.4.2. Backend

As discussed in Table 5.1, a Java RESTful API is used to implement the backend. And Jersey webservice framework is applied to implement the java RESTful API. The main objective of implementing the data migration logic as a RESTful service is that, the frontend of the DMT

can be any kind of application such as windows based, web based, console and mobile applications. That means, the backend can receive a request from all kinds of applications. As discussed above, the frontend of the prototype is a web-based application. So, the user who needs to migrate a database may use any type of devices that have a capacity to access the web. As figure 4.2 shows that, the backend (database migration API) has different modules such as metadata extractor and mapper, source data loader, thread manager, JSON data creator, data compressor and decompressor etc.

To build a JSON file, JSONObject class is used as shown below in the collectionBuilder method. The method takes list of schemas as an input and builds list MongoDB collection in JSON format.

```
public JSONObject buildTableStructure(List<String> allSchemas) throws SQLException {
    JSONObject tabelsObject = new JSONObject();
    JSONObject schemawithTabelsObject = new JSONObject();
    String tblName;// Build table name without schema name to get key of json object
    JSONObject foreignKey; // Foreign key constraint object
    String primaryKey;// Primary key constraint
    JSONObject constarints = new JSONObject();// JSONObject to store primary-key and foreign-key constraints
    for (String sch : allSchemas) {
        JSONObject jObject = new JSONObject();
        ResultSet tables = builder.getTables(sch);
        while (tables.next()) {
            constarints.clear();// Clearing constraints of previous table
            tblName = tables.getString( columnIndex: 3);// Getting table name from ResultSet
            primaryKey = extractor.getPrimaryKeyColumn(tblName);// Getting primary-key of a table
            foreignKey = extractor.getForegnKeys(tblName);//extract foreign key constraint of a table
            constarints.put("primary-key", primaryKey);
            constarints.put("foreign-key", foreignKey);
            jObject.put("Constraints", constarints);
            jObject.put("ColumnNames", builder.getColumns(tblName));
            tabelsObject.put(tblName, jObject);
        }
    return schemawithTabelsObject;
}
```

After building the structure of the MongoDB database, each record/row should be copied from relational database table to the corresponding MongoDB collection. The method below shows how a document is built from a row.

```
@SuppressWarnings("unchecked")
public JSONArray buildCollectionData( String  schemaName) throws SQLException {
    // Json object having all tables within the database
    JSONArray allTabelsObject = new JSONArray();
    String tblNameWithSchema;// Build table name with schema name for querying records
    JSONObject tabelObject = new JSONObject();
    ResultSet tables = builder.getTables(schemaName);
    while (tables.next()) {
            tblNameWithSchema = schemaName + "." + tables.getString( columnIndex: 3);
            JSONObject foreginKeyObject = extractor.getForegnKeys(tables.getString( columnIndex: 3));
            records = runner.executeQuery(tblNameWithSchema, foreginKeyObject);
            if (records.size() > 0) {
                tabelObject.clear();
                tabelObject.put("tableName", tables.getString( columnIndex: 3));
                tabelObject.put("records", records);
            }
       allTabelsObject.add(tabelObject);
    }
    return allTabelsObject;
}
```

The responsibility of the buildCollectionData method is to build MongoDB document and add the created document into its collection. Each dynamically generated thread executes this method to build MongoDB collection documents. After building the collection object, the result is compressed in zip format.

5.5. Evaluation criteria and result analysis

To perform an evaluation on the proposed solution, a DMT prototype is designed and implemented. Based on that, the following six evaluation criteria are identified to guide the evaluation of the experimental result. The identified database migration result evaluation criteria are:

- Usage of network bandwidth
- Performance
- Platform dependability

- Data transmission speed
- Data quality and
- Security

On the evaluation of the experimental result, these criteria are considered and each criterion are discussed as follows:

i. **Usage of network bandwidth:** the proposed DMT uses a special strategy to transfer data from source to destination. This strategy is transferring lots of data items together as a single and compressed file. i.e. lots of database records/rows can be combined into a single JSON file and also the file is in a compressed way. So, there is no a continuous transmission of database records from source to target database. Instead, there is a discrete way of data transmission.

ii. **Performance:** to implement a high-performance solution in the proposed solution, a multi-threaded approach is applied. Therefore, more than one thread executed parallelly in data migration task. That means the proposed solution works in a multi-tasking fashion. The other advantage of applying a multi-threading approach is that, multiple threads can share common resource. In addition to that, the dynamic garbage collection feature of java provides its own contribution to increase the performance of the proposed solution.

iii. **Platform independence:** platform independence is a crucial issue for cloud-based applications. Because, the designed service should be run on any kind of platforms. In this case, the proposed solution is platform independent in two ways: first, the backend: as discussed in section 4.5, the selected language of implementation for the actual DMT (backend) is java. And, by its nature an application implemented in java programming language is platform independent. So, the data migration webservice can be hosted on any cloud platform. Secondly, the frontend: the frontend of the proposed solution is implemented in a simple web technology (HTML5, bootstrap and angularjs). The responsive nature of bootstrap allows the frontend to be accessed by any internet

enabled devices. So, the users are not required to have any advanced such as PC to migrate their data. i.e. they can use their smartphone to interact with the DMT.

iv. **Data transmission speed:** in the case of data transmission speed, the contribution of a multi-threaded approach is high. As a discussion in section 4.8, multiple threads are responsible to migrate a single database. In addition to that, the data to be transmitted is in a compressed way and the size of each compressed file is too small.

v. **Data quality:** as a result of the experiment indicates that, all of the 6 schemas, 70 relations and 567,561 records are migrated successfully with no loss of data. The proposed solution has a feature of tracking each record, relation and schemas by an iterative way and in a post migration phase, the total relation and records are counted on the source and target databases. So, the proposed solution provides a good data quality during the migration.

vi. **Security:** for the proposed solution an SSL security protocol is integrated. So that, each and every data migration communication between two nodes is in an encrypted way. i.e. this solution is secured.

Chapter Six

6. Conclusion and Recommendations

6.1. Conclusion

The general objective of this thesis is to design a database migration algorithm to migrate database from one cloud provider to the other and also from a legacy database system to the cloud. In order to achieve this objective, the design, implementation and evaluation of the proposed solution has been discussed on previous chapters. Lots of different research papers, books, white papers, and websites has referred to realize state-of-the-art database migration tools and techniques and to find a research gaps which is not covered by other studies yet.

This work is intended to answer the hypothesis: “It is possible to enhance the performance of database migration by applying a multi-threaded approach”. The main point of the hypothesis is performance enhancement of database migration. The proposed solution helps to migrate database from relational database system to NoSQL database system by applying a multi-threaded approach and JSON data interchange format.

Thus, the implementation of a multi-threaded approach for the database migration increased the performance of the proposed solution by minimizing the time required to migrate the data by 1/N times, where N is the number of schemas, relations and 100,000 rows/records. By taking different criteria into consideration such as, efficiency, compatibility to target platform and data transmission time, JSON is the best data interchange format for data transmission.

For a database migration algorithm, Java is a preferred language of implementation based on its platform independence, reusability of syntax and query, support for multiple database drivers and support for web service development features. To make a secure data transmission between source and target, SSL security protocol is used. Due to the absence of schema constraint, the lack horizontal scalability and the capability to manage unstructured data, NoSQL database is the best approach to store and manage big data.

6.2. Recommendation

As discussed on the previous sections, the proposed solution is intended to migrate relational database (MS SQL) into NoSQL (MongoDB) database, any company or individual who needs to migrate data from a legacy relational database to NoSQL based cloud system may use this database migration solution. And also, researchers who needs to convert a relational database into NoSQL (MongoDB) for any experimental purpose can use this database migration solution.

6.3. Future Work

There are still several issues regarding the database migration on the cloud computing environment that permit further research work. Migration security and integrating a feature to support different kinds of databases will be seen in the future.

Therefore, the following are the future works:

- Integrating an advanced security mechanism to secure database migration over cloud computing environment
- The proposed solution is limited to migrate relational database (SQL) to a document-oriented database (MongoDB). In the future, the proposed DMT will support almost all kinds database systems.
- In this research, the proposed algorithm is implemented as a prototype. So, the solution will be implemented to provide a real database migration service.

References

[1] P Mell T. Grance, “The NIST Definition of Cloud Computing”, NIST Special Publication 800-145, September 2011

[2] Q. Shen, L. Zhang, X. Yang, Y. Yang, Z. Wu, and Y. Zhang, “SecDM: Securing Data Migration Between Cloud Storage Systems”, Ninth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2011

[3] S. Dewan, D. Kumar, S. Gonnade, “Secure Data Migration across Cloud System Using Third Party Auditor (TPA)”, International Journal of Innovative Research in Science, Engineering and Technology, June 2015

[4] E.Aruna, A.A. Shri and A. Lakkshmanan, “Security Concerns and risk at different levels in Cloud Computing”, International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), 2013

[5] S. Murugesan and I. Bojanova, "Cloud Computing: An Overview", Encyclopedia of Cloud Computing, John Wiley & Sons, Ltd, May 2016.

[6] K. Chandrasekaran and A. Ananth, "Cloud Services and Service Providers", Encyclopedia of Cloud Computing, John Wiley & Sons, Ltd, May 2016.

[7] Shawish and M. Salama, “Cloud Computing: Paradigms and Technologies”, Inter-cooperative Collective Intelligence: Techniques and Applications, Studies in Computational Intelligence 495, Springer-Verlag Berlin Heidelberg, 2014

[8] K. Bakshi and L. Beser, "Cloud Reference Frameworks", Encyclopedia of Cloud Computing, John Wiley & Sons, Ltd, May 2016.

[9] F. Fowley, C. Pahl, P. Jamshidi and D. Fang, “A Classification and Comparison Framework for Cloud Service Brokerage Architectures”, IEEE Transactions on Cloud Computing, January 2016

[10] D. Gallagher, “NIST Cloud Computing Standards Roadmap” Natl. Inst. Stand. Technol. Spec. Publ. 500-291, May 24, 2013

[11] H. Ali, R. Moawad, A.A.F. Hosni, “A Cloud Interoperability Broker (CIB) for data migration in SaaS”, in Future Computing and Informatics Journal 1, 2016

[12] Salesforce.com, “BENEFITS OF CLOUD COMPUTING”, 2019, [Online]. Available: https://www.salesforce.com/hub/technology/benefits-of-cloud/, [Accessed: 05-Jan-2019]

[13] M.G. Avram, “Advantages and challenges of adopting cloud computing from an enterprise perspective”, The 7th International Conference Interdisciplinarity in Engineering, 2013

[14] Attunity, “DATABASE MIGRATION TOOL”, [Online]. Available: https://www.attunity.com/database-migration-tool/ [Accessed: Dec 12, 2018]

[15] J. Opara-Martins, R. Sahandi and F. Tian, "Critical review of vendor lock-in and its impact on adoption of cloud computing," *International Conference on Information Society (i-Society 2014),* London, 2014

[16] J. Opara-Martins, R. Sahandi and F. Tian, "Critical review of vendor lock-in and its impact on adoption of cloud computing: a business perspective," *Journal of Cloud Computing: Advances, Systems and Applications*, 2016

[17] *Gilderman,* “*Database Migration—What Do You Need to Know Before You Start?*”, *21-NOV-2016. [Online]. Available:* httechttps://aws.amazon.com/blogs/database/database-migration-what-do-you-need-to-know-before-you-start/ [Accessed: 10-Sep-2018]

[18] G.Alley, “Data Migration tools”, 16-Oct-2018. [Online]. Available: https://www.alooma.com/blog/data-migration-tools. [Accessed: 05-Jan-2019]

[19] A. W. Service, "AWS," Amazon Web Service, Inc, 02-Jun-2006. [Online]. Available: https://aws.amazon.com/dms. [Accessed: 09-Jul-2018]

[20] TechTarget, “Top five data migration tools”, Jun, 2017, [Online]. Available: https://searchitchannel.techtarget.com/feature/Top-five-data-migration-tools, [Accessed: 12-Jul-2018]

[21] Microsoft Azure, “Azure database migration service-announcement”, 01-Jan-2012. [Online]. Available: https://azure.microsoft.com/en-us/blog/azure-database-migration-service-announcement-at-build/, [Accessed: 09-May-2018].

[22] M. Elamparithi & V. Anuratha. “A Review on Database Migration Strategies, Techniques and Tools”, World Journal of Computer Application and Technology, 2015

[23] Oracle, “Successful Data Migration”, Oct 2011, An Oracle White Paper

[24] Data Migration, [Online], Available: “https://en.wikipedia.org/wiki/Data_migration”, [Accessed: 05-Jan-2019]

[25] M. Islam, M. Manzurul, S. Morshed and P. Goswami, “Cloud Computing: A Survey on its limitations and Potential Solutions”, International Journal of Computer Science Issues, 2013

[26] B. Lin and Y. Chen, “Comparison between JSON and XML in application on AJAX”, International Conference on Computer Science and Service System, 2012

[27] One-to-One Relationships with Embedded Documents, [Online], Available: https://docs.mongodb.com/manual/tutorial/model-embedded-one-to-one-relationships-between-documents/”, [Accessed: 05-Jan-2019]

[28] What is an SSL certificate? [Online], Available: https://www.digicert.com/ssl/ , [Accessed: 10-May-2019]

[29] wikipedia.org, “Relational Schema”, 18-Dec-2018, [Online], Available: https://en.wikipedia.org/wiki/Relational_schema, [Accessed:15-Jun-2019]

[30] Considerations When Planning To Test Data Migration, [Online], Available: https://www.qualitestgroup.com/white-papers/considerations-planning-test-data-migration/, [Accessed: 10-May-2019]

[31] S.Sarmah "Data Migration", Scientific & Academic Publishing, 2018, [Online], Available: http://journal.sapub.org/scit

[32] G. Zhao, Q. Lin, L. Li et. al., "Schema Conversion Model of SQL Database to NoSQL", Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, IEEE, 2014

[33] E. Andirson et. al., "An Experimental Study of Data Migration Algorithms", University of Washington

[34] T. Jia, X. Zhao, Z. Wang, D. Gong and G. Ding, "Model Transformation and Data Migration from Relational Database to MongoDB", IEEE International Congress on Big Data, 2016

[35] M.N. Shirazi, H.C. Kuan and H. Dolatabadi, "Design Patterns to Enable Data Portability between Clouds' Databases", 12th International Conference on Computational Science and Its Applications, 2012

[36] S. J´anos, "The algorithmicx package∗", LaTeX project, April 27, 2005

[37] Salesforce.com, Inc., "CRM 101: What is CRM?", 2019. [Online]. Available: https://www.salesforce.com/crm/what-is-crm/. [Accessed: 20-Apr-2019]

[38] Microsoft Inc., "What is Office 365", 2019, [Online]. Available: https://www.office.com/. [Accessed: 26-Apr-2019]

[39] Wikipidea, "List of Google products", June 2019, [Online], Available: https://en.wikipedia.org/wiki/List_of_Google_products. [Accessed: 10-Jun-2019]

[40] Amazon Inc., "AWS Database Migration Service", 15-Dec-2018, [Online], Available: https://aws.amazon.com/dms/, [Accessed: 10-Jun-2019]

[41] A. Alemu, “Quantification of Quality Expressions for Cloud Migration Decision Support”, May 2018, ASTU