MIGRATION FROM A RELATIONAL DATABASE TO A DOCUMENT-ORIENTED DATABASE BASED ON DOCUMENT-ORIENTED DATA SCHEMA

HAMOUDA SHADY M S

UNIVERSITI SAINS MALAYSIA

2020

MIGRATION FROM A RELATIONAL DATABASE TO A DOCUMENT-ORIENTED DATABASE BASED ON DOCUMENT-ORIENTED DATA SCHEMA

by

HAMOUDA SHADY M S

**Thesis submitted in fulfilment of the requirements
for the degree of
Doctor of Philosophy**

July 2020

ACKNOWLEDGEMENT

بسم الله الرحمن الرحيم

{نَرْفَعُ دَرَجَاتٍ مِّن نَّشَاء وَفَوْقَ كُلِّ ذِي عِلْمٍ عَلِيم} .(سورة يوسف 76)

First of all, I would like to thank “Allah”, our creator, for giving me his blessings and the opportunity to complete my PhD.

Now, I would like to share my happiness by acknowledging some of the numerous people who have helped me directly or indirectly in shaping my academic career over the past years. I would like to express my deepest gratitude to my learned supervisors Assoc. Prof. Dr. Zurinahni Zainol (main supervisor) for her encouragement, innovative suggestions, and invaluable help during the entire period of my PhD in Universiti Sains Malaysia.

Most importantly, I would like to thank my family. Their support, encouragement, patience, and unwavering love were undeniably the foundation. I thank my parents, Mohammed and Najah, for their faith in me and for allowing me to be as ambitious as I wished. It was under their watchful eye that I gained the drive and ability to tackle challenges head-on. I would also like to thank my friends Dr. Mohammed Anbar and Dr. Omar Elejla unconditional support and guidance as needed. Last but not the least, I wish to dedicate this work to my wife Amal and my kids Yamen, and Mohammed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACID	Atomicity, Consistency, Isolation, and Durability
ACK	Acknowledgment
BASE	Basically Available, Soft State, Eventual consistency
BSON	Binary encoding of JavaScript Object Notation
CAP	Consistency, availability, and partition tolerance
CRUD	Create, Retrieve, Update, and Delete
CSV	Comma-separated values
DODS	Document-oriented data schema
ER	Entity relational
EReX	Entity relational extended to XML
ERX	Entity relational for XML
GN-DTD	Graphical notations-data type documentation
GOOSSDM	Graph object-oriented semi-structured data model
JSON	JavaScript Object Notation
NoSQL	Not only structured query language
ORA-SS	Object relationship attribute model for semi-structured data
QODM	Query-oriented data modelling
RDBMS	Relational database management system
SQL	Structured query language
TR	Transformation rules
UML	Unified Modeling Language
XER	Extensible ER
XML	EXtensible markup language
XSEM	Conceptual model for XML
XUML	Executable of Unified Modeling Language

LIST OF APPENDICES

MIGRASI DARI PANGKALAN DATA HUBUNGAN KE PANGKALAN DATA BERORIENTASI DOKUMEN BERDASARKAN SKEMA DATA BERORIENTASI DOKUMEN

ABSTRAK

Data raya adalah isu penting yang muncul sebagai salah satu teknologi paling penting di dunia moden. Kebanyakan kajian menonjolkan ketidakupayaan pangkalan data hubugan untuk mengendalikan data raya. Cabaran ini telah membawa kepada penyampaian "pangkalan data pertanyaan bahasa berstruktur (NoSQL) bukan sahaja" sebagai konsep teknologi pangkalan data baru. Salah satu jenis pangkalan data NoSQL yang paling berkuasa adalah pangkalan data berorientasikan dokumen yang menyokong data skema dan penyimpanan yang fleksibel dalam format separa berstruktur. Baru-baru ini, ramai penyelidik terpaksa berhijrah dari pangkalan data hubungan ke pangkalan data berorientasikan dokumen kerana kebolehan skalabiliti, ketersediaan, dan prestasi. Walau bagaimanapun, kaedah penghijrahan mereka menghadapi tiga isu; isu pertama ialah tidak ada spesifikasi yang dapat mengenali untuk menentukan skema untuk pangkalan data berorientasikan dokumen, dan kedua, tidak ada cara untuk normalisasi dan de-normalisasi data untuk melaksanakan dokumen tertanam dan rujukan. Yang ketiga adalah penghijrahan dari pangkalan data hubungan ke pangkalan data yang berorientasikan dokumen tidak menganggap semua sifat yang terdahulu, terutama tentang cara mengendalikan berbagai jenis hubungan. Penyelidikan ini mencadangkan metodologi untuk menangani isu penghijrahan melalui tiga fasa: pertama, reka bentuk skema data yang berorientasikan dokumen (DODS) berdasarkan model entiti-hubungan (ER); kedua, mencadangkan peraturan transformasi untuk memetakan skema hubungan entiti kepada skema data berorientasikan dokumen berdasarkan data normalisasi dan de-normalisasi; ketiga, memindahkan pangkalan data hubungan ke pangkalan data berorientasikan dokumen. Kajian ini menyediakan enam penilaian; penilaian ini memberi tumpuan kepada pengesahan kebolehpercayaan penghijrahan dari pangkalan data hubungan ke pangkalan data berorientasikan dokumen berdasarkan skema yang dicadangkan. Akhir sekali, skema yang dicadangkan menyediakan ciri-ciri baru untuk perwakilan konseptual dari pangkalan data berorientasikan dokumen. Kajian ini menilai skema fleksibiliti, membandingkan ciri DODS dengan ciri separa struktur, dan menilai prestasi data dinormalisasi dan dinormalisasi untuk pangkalan data berorientasikan dokumen berdasarkan DODS. Penilaian seterusnya adalah fokus pada pengesahan kebolehpercayaan migrasi dari pangkalan data relasional ke pangkalan data berorientasikan dokumen berdasarkan skema yang dicadangkan. Akhirnya, hasil proses penghijrahan menunjukkan semua sifat pangkalan data hubungan dan data yang telah dipindahkan tanpa kehilangan data atau pertindihan berbanding kaedah penghijrahan sebelumnya. Hasilnya juga menunjukkan bahawa penghijrahan data mempunyai prestasi yang lebih baik daripada pangkalan data hubungan dan kaedah penghijrahan sebelumnya.

MIGRATION FROM A RELATIONAL DATABASE TO A DOCUMENT-ORIENTED DATABASE BASED ON DOCUMENT-ORIENTED DATA SCHEMA

ABSTRACT

Big data is a crucial issue that has emerged as one of the most important technologies in the modern world. Most studies have highlighted the inability of a relational database to handle big data. This challenge has led to the presentation of the "not only structured query language (NoSQL) database" as a new concept of database technology. One of the most powerful types of NoSQL databases is the document-oriented database that supports a flexible schema and store data in a semi-structured format. Recently, many researchers have migrated from relational databases to document-oriented databases because of scalability, availability, and performance. However, their migration methods are facing three issues; the first issue is that there are no specifications that can be recognized to define a schema for a document-oriented database, and second, there is no method to normalize or de-normalize data in order to implement the embedded and reference document. The third is the migration from the relational database to a document-oriented database does not consider all the properties of the former, especially on how to handle various types of relationships. This study proposed a methodology to handle the migration issues through three phases: first, design a document-oriented data schema (DODS) based on the entity-relational (ER) model; second, enhance transformation rules to map the entity relational schema to the document-oriented data schema based on normalization and de-normalization data; third, migrate a relational database to a document-oriented database. The study evaluates the flexibility schema, compare the DODS features with the semi-structure features, and evaluate the performance of normalized and de-normalized data for a document-oriented database based on DODS. The next evaluations will be focusing on the verification of the reliability of the migration from a relational database to a document-oriented database based on the proposed schema. Finally, the proposed schema provides new features for the conceptual representation of a document-oriented database. In addition, transformation rules have been applied to three case studies. The result showed that all the properties of the entity-relationship schema have been migrated to the document-oriented data schema in terms of strategy to apply the embedded and reference documents in the migration to avoid data redundancy and improve the database performance. By the end, the result of the migration process showed all the relational database properties and data to have been migrated without data loss or duplication compared to the previous migration methods. The result also showed that data migration has better performance than a relational database and the previous migration methods.

CHAPTER 1

INTRODUCTION

Nowadays, business applications need databases that can support extreme scales, deal with all kinds of data formats, respond in quickly delivery level with high performance. The new business applications requirements that encompass flexibility in data model structure support the next generation of web applications, and handles complex data types, presents a challenge for organizational systems.

According to Katal *et al*. (2013), data size is expected to reach 35 zettabytes by 2020 and grow at a rate of 40 % per year (Manyika *et al*., 2011). This huge volume of data has become a big problem for big companies. For instance, Walmart performs around 267 million transactions per day while Facebook generates around 3 billion pieces of content per day (Chen & Zhang, 2014). This flood of data has caused many issues and challenges pertaining to big data. Stanescu *et al*. (2016) revealed the challenges of big data, such as the method of dealing with the increasing data volume. In addition, Assunção *et al*. (2015) presented one of the important issues related to big data and that is the need for the semi-structured data type for storage, and handling of large amounts of data with flexibility schema (Yaish & Goyal, 2013). Information technology in the big organization is trying to shift from structure data to semi-structured data (Wang *et al*., 2018).

1.1 Relational database

Various studies confirmed that the relational database cannot cope with the large volumes of data as it has restrictions towards meeting scalability, flexibility, and performance challenges (Anagnostopoulos *et al.*, 2016; Assunção *et al.*, 2015; Atzeni *et al.*, 2014; Bhogal *et al.*, 2015; Chickerur *et al.*, 2015; Goyal *et al.*, 2016; Gudivada *et al.*, 2014; Liang *et al.*, 2015; Mehmood *et al.*, 2017; Ogunyadeka *et al.*, 2016). For instance, Assunção *et al.* (2015) discussed the problems that occur in a relational database which is a challenge in handling big data; the problems are how to process data variety and data velocity. The relational database has structured data, fixed schema, vertical scalability, and stores data in the table that follows the same schema (Grolinger *et al*., 2013). Currently, a relational database management system (RDBMS) is inefficient in handling applications and software requirements of big data, such as supporting horizontal scaling for a distributed environment, and inability to achieve effective data portability (Ogunyadeka *et al*., 2016; Chickerur *et al*., 2015; Liang *et al*., 2015; Hashem *et al*., 2015). Therefore, many organizations are looking forward to the next generation of data management to support their business application (Atzeni *et al*., 2016, Rodríguez-Mazahua *et al*., 2016).

These issues and challenges have led to the development of a Not only SQL (NoSQL) database as a new technology to overcome the limitations of the relational database, such as designing a schema without strict constraints (Atzeni *et al.*, 2014; Hashem *et al.*, 2016). In addition, the NoSQL database can accept all types of structured, semi-structured, and unstructured data, and has many features, such as a support-distributed system, flexible schema, horizontally scalable, and easy replication (Chickerur *et al.*, 2015; Lombardo *et al.*, 2012; Rocha *et al.*, 2015a).

Nowadays, big organizations are looking forward to the NoSQL database as the next generation of data management to support their business application (Atzeni *et al*., 2016, Rodríguez-Mazahua *et al*., 2016). This is due to the exponential increase in the amount of data and the need for flexibility schema with semi-structured data. In addition, new technology like cloud, mobile, and social media has caused organizations to consider migrating from the traditional relational database to NoSQL database since their application cannot satisfy the scalability and availability requirements (Goyal *et al*., 2016; Gudivada *et al*., 2014; Han *et al*., 2012; Kanade *et al*., 2014; Bansel *et al*., 2016). Therefore, they consider the NoSQL database as an alternative to the relational database (Dharavath and Kumar, 2015, Anagnostopoulos *et al*., 2016, Bhogal and Choksi, 2015). For instance, Gannett has moved from a relational database to NoSQL database to improve its digital publishing platform. Also, Marriott relays the NoSQL database to modernize its reservation system. Telefonica has also migrated from Oracle to MongoDB to personalize their customer services, be more agile, and save cost (Couchbase).

1.2 NoSQL database

NoSQL database has a different data model concept that is classified according to the storage and retrieval of data; therefore, each database has different ways of designing, storing, and processing data. The NoSQL database is classified into a key-value database, column database, document-oriented database, and graph database.

The key-value database uses a hash table that consists of a key and a set of values (Grolinger *et al*., 2013; Nayak *et al*., 2013). The values can be a set of simple or complex data, and the key is used to retrieve these data. The column database stores data in columns and rows (Tauro *et al*., 2012). Each row has many columns called column families, which may hold numerous columns and act as keys for these columns. In addition, the graph database stores data as nodes that are considering the entities (Jayathilake *et al*., 2012). The properties of the nodes represent the attributes, whereas the edges represent the relationships among the nodes (Kaur *et al*., 2013). Finally, the document-oriented database uses to manage and store data in document format and collections. The central concept of a document-oriented database is the notion of a document whose contents are encapsulated or encoded in some standard format to store and retrieve the data, such as JavaScript Object Notion (JSON) or

Binary encoding of JavaScript Object Notation (BSON), and EXtensible markup language (XML) (Li *et al.*, 2014). Each document has a unique primary key (Mapanga *et al.*, 2013). The document can have different data types, such as complex data structure, nested objects, array, and embedded document (dos Santos Ferreira *et al.*, 2013).

Based on Figure 1.1, the document-oriented database can be a suitable type to represent a semi-structured data format and support scalability in data size (Ruflin *et al.*, 2011).

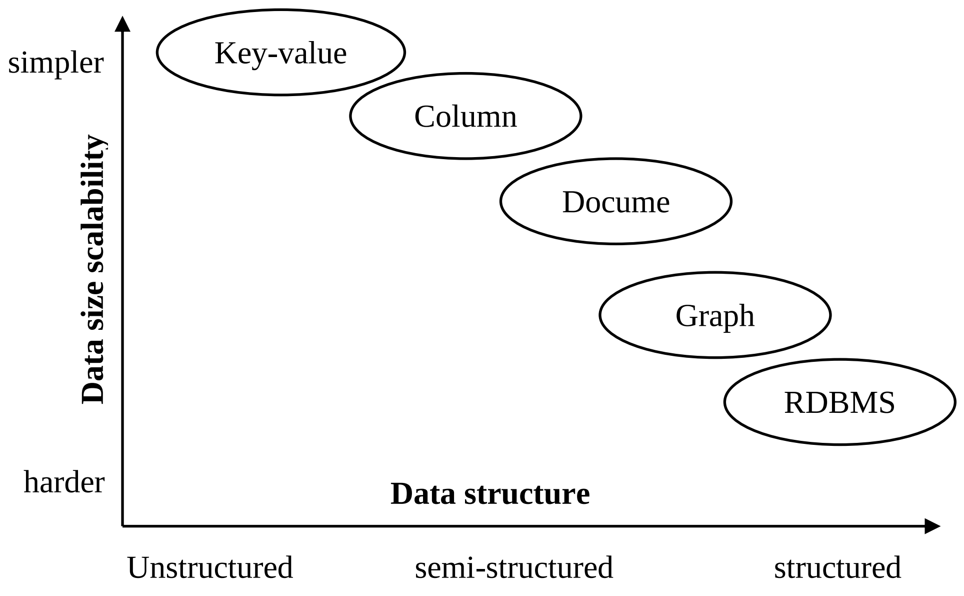

Figure 1.1 Scalability of data size depending on the data structure (Ruflin *et al.*, 2011)

The difference between the key-value, column, document-oriented, and graph database is that the document-oriented database supports the transparency of the data (Atzeni *et al.*, 2016). The document-oriented database use to store, retrieve, and manage data using a semi-structured data format, as well as provides high performance, availability, and scalability (Stanescu *et al.*, 2016). Also, the document-oriented database can define a field into a document, and the application can query this document by using the fields. In addition, the document-oriented databases can create an index over fields, and these indexes can help optimize queries that are used as a reference to the fields (Hashem *et al.*, 2016). This capability means that a document-oriented database can be more suitable than other NoSQL database for storing a large amount of data that needs to be retrieved based on more complex criteria (Corbellini *et al.*, 2017). Therefore, this study uses the document-oriented database to achieve the objectives of this thesis.

Finally, the low efficiency of relational databases is a problem for the current applications and demand migration of their system from relational databases to a document-oriented database (Corbellini *et al.*, 2017; Dharavath *et al.*, 2015; Karnitis *et al.*, 2015; Yoon *et al.*, 2016). This is because the concept of a document-oriented database is to be fast and efficient in data processing in terms of scalability, variability, agility, and performance (Grolinger *et al.*, 2013). Moreover, a document-oriented database supports flexible schema, semi-structured or unstructured data, and horizontal scalability (Rodríguez-Mazahua *et al.*, 2016).

1.3 Motivation

With the rise of big data, a relational database has been unable to fit the dimensions of big data, especially data velocity and variety (Abourezq *et al.*, 2016). According to Younas (2019), storing and managing big data requires new data models and technologies. Also, the increasing use of web applications has raised the demand to use a document-oriented database because traditional databases are unable to handle the rapidly growing data volume (Corbellini *et al.*, 2017). In addition, there is a need to deal with semi-structured data with a flexible schema for different applications (Goyal *et al.*, 2016).

The document-oriented database provides a new characteristic solution to adapt to big data challenges. The first one is that it supports horizontal scalability by automatically spreading data over many servers and adding more machines to a pool of resources (Chandra, 2015). A relational database supports vertical scalability by adding more hardware infrastructure to the existing machines. This comes with many problems, such as expensive and difficult hardware limitations, complex joint operations, non-reliability in some cases, and data distribution over many servers (Chandra, 2015, Atzeni *et al*., 2014). The second characteristic is its high availability towards highly distributed databases through availability and partition-tolerance in the data store, while RDBMS supports consistency and availability (Grolinger *et al*., 2013; Chandra, 2015). Therefore, the low efficiency of the relational database becomes a problem for the current application and demand migration of their system from a relational database to a document-oriented database (Corbellini *et al.*, 2017; Dharavath *et al.*, 2015; Karnitis *et al.*, 2015; Yoon *et al.*, 2016).

As Abdelhedi *et al.* (2018) mentioned, a document-oriented database has proven to be the most adapted solution that supports a larger volume of data and provide flexible schema. Moreover, Younas (2019) found that the document-oriented database can be suitable for high development productivity and low maintenance cost of modern Web 2.0 applications for two main reasons: First, these applications have a constant evolution of data schema and benefits from the flexible schema of the document-oriented database; second, Web 2.0 applications support data models such as JSON with tight integration with popular programming languages such as Python, JavaScript, and Ruby. Therefore, many programming languages can support a document-oriented database as it can provide high throughput than many other databases (Bathla *et al.*, 2018).

This study discusses the factors that motivate studying the migration from a relational database to a document-oriented database. First, there is a need for a conceptual model that can represent a schema for a document-oriented database. In addition, there is a need for a method to migrate all the database properties from a relational database to be adapted with a document-oriented database. At the same time, the issue of how to normalize and de-normalize data in a document-oriented database is still not handled.

1.4 Research Problem

Migration from a relational database to the document-oriented database has become an important topic in the era of big data (Chouder *et al.*, 2017; Kim *et al.*, 2018). Big organizations, such as Telefonica, Financial giant RBS, Travelers insurance, Cisco, etc have begun to transform their relational database to a document-oriented database (Gudivada *et al*., 2014). However, Oliveira *et al.* (2018) found that there is no information about the migration methods and scenario configuration; also, there are no investigations to understand the migration process or the methodology of their migration. At the same time, a framework and methodology for migrate from a relational database to a document-oriented database are needed (Lee *at al*., 2015; Győrödi *et al*., 2015; El Alami *at al*., 2016; Hanine *et al*., 2016; Karnitis *et al*., 2015; Stanescu *et al*., 2017). Therefore, migration is becoming a challenge in this study area (Kim *et al*., 2018).

Evidently, many researchers have proposed methods to migrate the relational databases to the document-oriented database (Chickerur *et al.*, 2015; Corbellini *et al.*, 2017; Dharavath *et al.*, 2015; El Alami *et al.*, 2016; Goyal *et al.*, 2016; Győrödi *et al.*, 2015; Hanine *et al.*, 2016; Imam *et al.*, 2018; Kanade *et al.*, 2014; Karnitis *et al.*, 2015; C.-H. Lee *et al.*, 2015; Mason, 2015; Rocha *et al.*, 2015a; Stanescu *et al.*, 2016, 2017;

Yoon *et al.*, 2016; Zhao *et al.*, 2013). For instance, El Alami *et al.* (2016); Hanine *et al.* (2016); Mason (2015); Stanescu *et al.* (2016, 2017) have focused on migrating a relational database to a document-oriented database based on the concept of embedded and reference documents.

However, these migration methods are facing various issues; the first issue is that there is no specification that can be recognized to define a schema for a document-oriented database (Mohan, 2013; Moore *et al*., 2014; Kanade *et al*., 2014) due to the various ways of storage, management, and implementation in document-oriented database (Tauro *et al*., 2012; Goyal *et al*., 2016). The lack of presenting a schema leads to present many challenges and complex problems in migration as because design a schema for the document-oriented database is important for defining the principles and overcoming the issues of relationship types for document-oriented databases (Roy-Hubara, 2019). Also, it may lead to incorrect or inappropriate schema design especially when handling the relationships based on normalizing and de-normalizing data. In addition, the migration methods still need to improve on how to represent all database properties, such as entity types (whether it is a strong entity or weak entity), attributes types (whether is it ordinary, multi-value, composite, primary key, foreign key), and the types of the relationships (1:1, 1: M, M: M, and unary). For instance, the method of Stanescu *et al*. (2017), did not migrate all the database properties in a proper way specially, the multi-values, weak entity, relationship types. Some migration result is an embedded document while it should be migrated by using array data type as it contains one field with many values. In addition, if there is any table refereed by more than two other tables and has more than one foreign key. These cases were missing in the Stanescu *et al*. (2017) algorithm.

Second, there is no technique method to normalize or de-normalize data in order to implement the embedded and reference document for handling the various types of relationships (Guimaraes *et al*., 2015; Khan and Mane, 2013; Hanine *et al*., 2016; Mehmood *et al*., 2017). Handling relationships based on embedded and reference documents has not been considered in document-oriented databases despite its importance probably because it is not recommended in creating a collection for each entity or using a reference document for all because of the need to execute a complex joint operation. Furthermore, storing all the entities as embedded documents into one collection is not beneficial because it will cause many redundant and inconsistent data (Atzeni *et al.*, 2016). In addition, it will load all the data when updated and may reduce performance. According to Mehmood *et al.* (2017), normalization (reference document) and de-normalization (embedded document) are the two techniques that must be considered when designing a schema. These techniques can affect the performance and storage effectively as the databases grow rapidly. González-Aparicio *et al*. (2017) observed that the normalization of the data model is one of the important research issues and there are no standard principles of normalization in the document-oriented database.

Finally, the migration from a relational database to a document-oriented database does not consider all the database properties, especially on how to handle various types of relationships (Colombo *et al.*, 2019; El Alami *et al.*, 2016; Győrödi *et al.*, 2015; Hanine *et al.*, 2016; Stanescu *et al.*, 2017). As because migration without any specification or methodology to normalize and de-normalized the various types of relationships will cause incorrect migration. In addition, the data migration may be missing or duplicated that will affect the performance.

1.5 Research Questions

This study aims to answer the questions that need to be addressed for the research area, such as:

i) How can a schema for a document-oriented database be designed?

ii) How can a relational database schema be mapped so that it can be adapted with a document-oriented database schema?

iii) How can a relational database be migrated to a document-oriented database?

1.6 Objectives

The main goal of this study is to migrate from a relational database to a document-oriented database based on a document-oriented data schema. To achieve the objectives, this study proposed the following:

i) To propose a feature and specification for designing a document-oriented data schema (DODS) based on the entity-relationship (ER) model in order to represent a conceptual schema for the document-oriented database.

ii) To enhance the transformation rules to map a relational schema to a document-oriented data schema in order to define a strategy of normalization and de-normalization of data in a document-oriented database.

iii) To migrate the relational database to a document-oriented database based on a document-oriented data schema through a methodology that can cover all database properties (entities, fields, relationships, data, and constraints) in migrations.

1.7 Contributions

This study overcomes the issues presented in the migration of a relational database to a document-oriented database by proposing specifications and features for designing a schema for a document-oriented database. This schema can be used as a conceptual model for a document-oriented database and can be used as a road map to migrate from a relational database to a document-oriented database. This migration method can be efficient and reliable for use in migrating from a relational database to a document-oriented database.

The main contributions of this study are summarized as follows; a new schema for a document-oriented database that represent conceptual data for a document-oriented database.

Also, enhancing the transformation rules to map a relational schema to document-oriented schema. Theses transformations rules can overcome the issues of handling the relationships of a complex database and can be used to implement normalization and de-normalization data in a document-oriented database.

Finally, migrate a relational database to a document-oriented database to fulfill the gaps of migrating all ER properties such as entities, fields, relationships, data, and constraints.

1.8 Scope

This study focused on two dimensions of big data; the first one is data variety, especially in a semi-structured data format, and the second one is the increasing volume of data with a consideration of the speed performance. Moreover, this study focused on a document-oriented database using the MongoDB database as a database management system. This study does not consider the distributed processing performance of a document-oriented database.

1.9 Thesis Structure

This thesis is structured as follows:

Chapter 2 presents a brief discussion on the database management system, the concepts of formatted, structured, and semi-structured data, and issues related to semi-structured data models. It also summarizes the NoSQL database, discusses and analyses the related models for migrating a relational database to a document-oriented database.

Chapter 3 presents the research methodology of the proposed method. It explains the research approach, the case study of this study, and the evaluations performed to evaluate the proposed method.

Chapter 4 describes the specification and features of DODS and the mapping of the specification of the ER schema to a document-oriented data schema.

Chapter 5 presents transformation rules to transform ER specifications to DODS using case studies. The DODS perform the following evaluations; (i) evaluate the flexibility schema; (ii) compare the DODS features with the semi-structure features; (iii) evaluate the performance of normalized and de-normalized data for a document-oriented database based on DODS;

Chapter 6 discusses the migration of a relational database to a document-oriented database using a case study.

Chapter 7 evaluates the proposed method through following evaluations; (i) the migration process through migration accuracy; (ii) the performance of the migration time between this study and Stanescu *et al*. (2017) to check whether this study can migrate all the database properties; (iii) evaluate the performance of the database operations between relational databases using Oracle and document-oriented database using MongoDB.

Chapter 8 presents the research conclusion, the main finding of the research, and the limitation; this chapter also presents the recommendations for future work.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

This chapter presents an overview of big data, relational databases, and the NoSQL database. It also highlights the NoSQL database and provides details for each model. The literature review discusses the methods of migrating from relational databases to document-oriented databases.

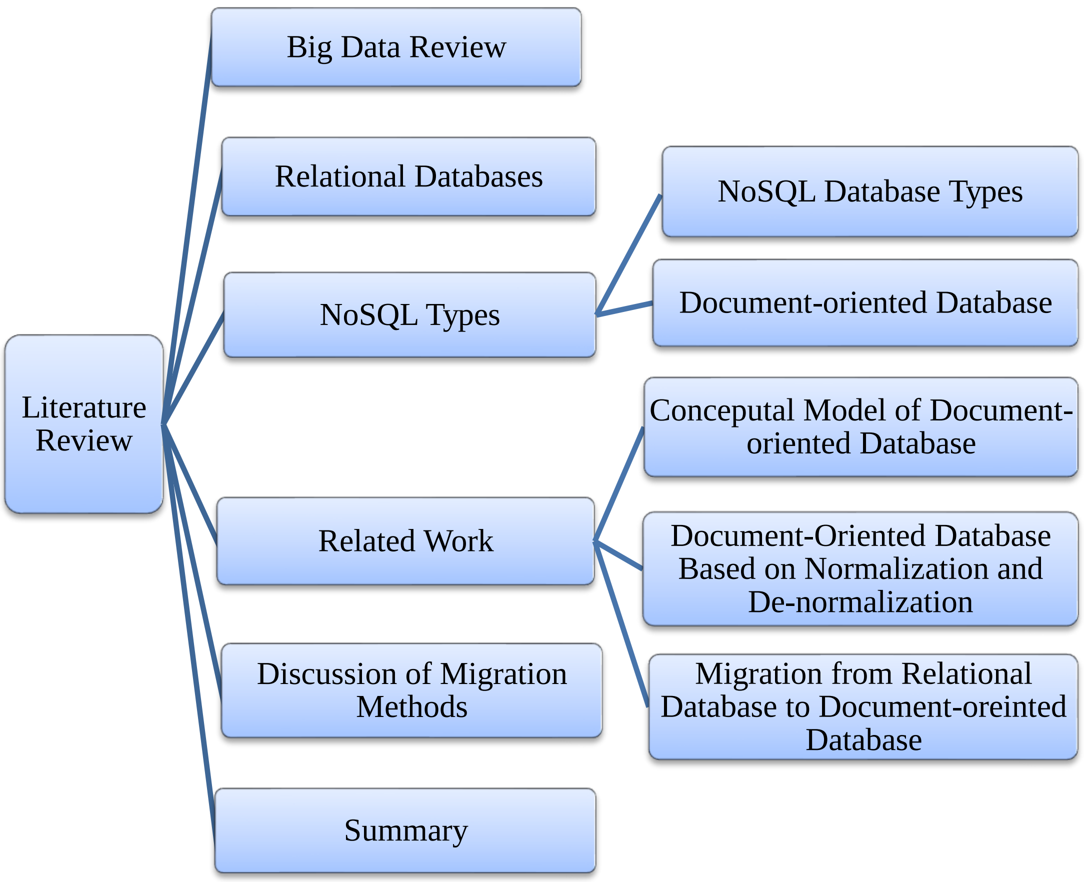
Literature Review
Big Data Review
Relational Databases
NoSQL Types
Related Work
Discussion of Migration Methods
Summary
NoSQL Database Types
Document-oriented Database
Conceptual Model of Document-oriented Database
Document-Oriented Database Based on Normalization and De-normalization
Migration from Relational Database to Document-oreinted Database

Figure 2.1 Overview of literature review

2.1 Big Data

Big data becomes an important research topic in developing technologies that can manage and process large volumes of data (Rodríguez-Mazahua *et al.*, 2015; Stanescu *et al.*, 2016; Truica *et al.*, 2015). Big data is characterized by five main dimensions known as the 5V characteristics (Chen *et al.*, 2013; Katal *et al.*, 2013; Pokorny, 2013; Rodríguez-Mazahua *et al.*, 2016). These dimensions are 1) volume, which concerns the data scalability as data grow every day; 2) velocity, which represents the speed of data and indicates the critical point of big data by measuring the performance of transactions; 3) variety, which represents the data format, such as structured, semi-structured, and unstructured; 4) veracity, which is concerned with data accuracy; and 5) value, which describes the importance of data. According to Bhogal *et al.* (2015), the 5Vs of big data should be evaluated using the NoSQL database in the future. The problem of big data will become increasingly prominent as solutions continue to be developed to meet emerging needs (Ahuja *et al.*, 2013). The elements of the big fata taxonomy are described in Figure 2.2.

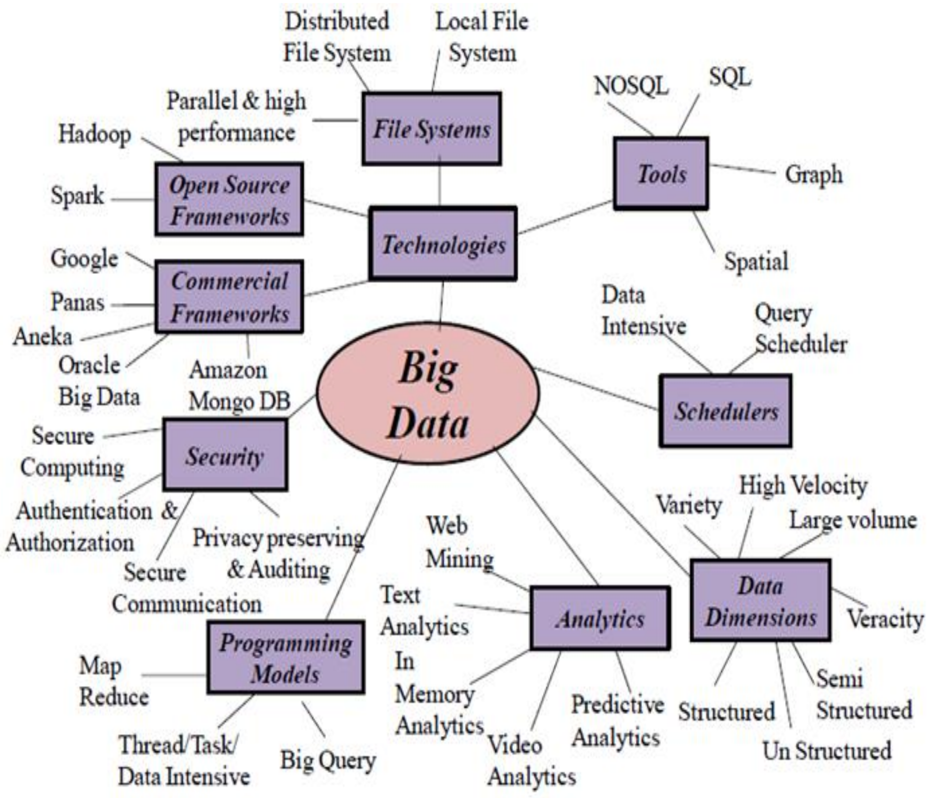

Figure 2.2 Big data taxonomy (Pokorny, 2013)

Figure 2.2 shows many issues and challenges pertaining to big data that need to be addressed in research and industries. According to Stanescu *et al.* (2016), dealing with increases in data volume and storing and analyzing these data are pressing concerns for big data. Assunção *et al.* (2015) presented another important issue which the method of handling and processing a semi-structured data with a flexible schema. Semi-structured data (Yaish *et al.*, 2013) are required to store and handle large amounts of data with flexibility schemas, which have emerged as one of the biggest data models for handling large amounts of data (Qureshi *et al.*, 2011). Figure 2.3 describes the growth of structured, semi-structured, and unstructured data (Feng *et al.*, 2015). Semi-structured data have the biggest data growth, with more than 2.5 trillion gigabytes as of 2014.

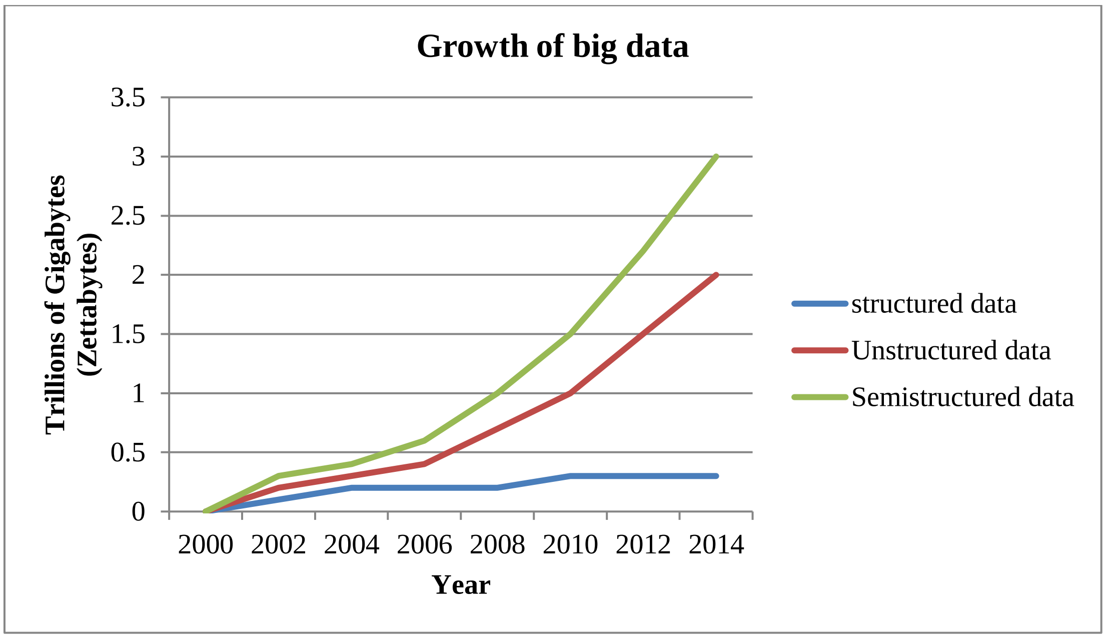

Figure 2.3 Growth of big data (Feng *et al.*, 2015)

As shown in Figure 2.3, semi-structured data captured and stored data have become huge and need to process flexible schemas and speed through distributed systems (Lombardo *et al.*, 2012). Hashem *et al.* (2016) stated that NoSQL database distribution supports flexible schemas and it can handle and process semi-structured data.

Atzeni *et al*., (2013) explained that an efficient model needs to be designed for semi-structured data due to a lack of semi-structured data models with flexible schemas. These issues and challenges must be addressed by researchers when designing a method or algorithm to retrieve information from large amounts of data (Qureshi *et al*., 2011). These requirements contribute significantly to the development of semi-structured data (Yaish & Goyal, 2013).

Relational databases store data in the form of tables with records and rigid structures to organize the data that slow down data access and make it inconvenient (Liang *et al*., 2015). Recently, many organizations have been facing critical problems with relational databases in terms of handling increasing data volumes and the big data application demand for flexible data schemas (Kanade *et al*., 2014, Madison *et al*., 2015). These firms are aiming to adapt to the demands of scalability, high availability, and storage of huge amounts of data. Therefore, the low efficiency of relational databases poses a problem for current applications and requires system migration from a relational database to NoSQL database (Corbellini *et al*., 2017, Yoon *et al*., 2016, Dharavath & Kumar, 2015, Karnitis & Arnicans, 2015). Goli-Malekabadi *et al.* (2016) considered variety an important dimension for big data management because it describes the type and nature of data.

2.2 Relational Database

A relational database is based on the relational model, which allows the definition of data structures, storage and retrieval operations, and integrity constraints. In the relational database, the data and relationship are organized into tables (Zhao *et al*., 2013). Each table has rows (records) and columns (attributes). Additionally, the model uses SQL to access and process data (Neves *et al.*, 2015). The schema represents the interaction between the database designer and the end-user to depict the data structure.

As Kune *et al.* (2016) stated, most organizations use a relational database as structured data to store and access their data. However, with the rise of big data, relational databases have been failing to fit the dimensions of big data, especially data velocity and variety (Abourezq *et al.*, 2016). Thus, organizations encounter problems with speed and the increasing size of data. For this reason, the relational database application has become a bottleneck when data increase and require added scalability (Moore *et al.*, 2014).

Migrating a relational database involves structuring data to a document-oriented database, which is a semi-structured data format. This section reviews the specification of structured data to cover all the ER properties in the migration.

2.2.1 ER Model of Structured Data

An ER model is a graphical representation of the conceptual view of a database (relational database) and describes the requirements of any application through entities and relationships (J.-W. Lee *et al.*, 2002). The following table describes the standard notations used to design an ER model.

Table 2.1 Chen’s model notations for an ER model (Lee *et al*., 2002)

Abstraction Components	ER Model
Entity	▭
Relationship	◇
Attribute	⬭
Aggregation/Decomposition	◇⟶
Cardinality	1 ─── N
Connector	○

The main abstraction components are used to represent the schema of the ER model and can be described as follows.

i) The entity is represented by a rectangular shape with the name “entity”.

ii) The relationship is represented by a diamond shape between two entities.

iii) The attributes are represented by an oval shape with the name “attribute.”

iv) The aggregation/decomposition is represented by a diamond and a line between two entities to describe the joint operations.

v) The cardinality is represented by a line shape with the types of relationships between two entities.

2.3 NoSQL Database

Ahuja *et al.* (2013) indicated that the principal concept of the NoSQL database is to store data as key-value pairs. Key considers the field of the entity, while the values are associated with the key. These values can be any kind of data structure and differ from those in the relational database, where all the field data should have the same structure and may have null values. In addition, this key-value pair can be stored in a document, which can represent a field with a record in the relational database.

Consequently, a set of related documents is stored and represented through a collection that considers the table of the relational database.

NoSQL database overcomes big data requirements by supporting flexible schemas. These databases can accept all types of structured, semi-structured, and unstructured data more easily than a relational database (Chandra, 2015).

2.3.1 NoSQL Database Types

NoSQL database is classified as key-value, column, document-oriented, and graph databases. These categories have different ways of storing and managing data and have varied means of data modeling. Bansel *et al.* (2016); Grolinger *et al.* (2013); Gudivada *et al.* (2014); Tauro *et al.* (2012); Yaish *et al.* (2013) explained the types of NoSQL database as follows.

a) Key-value database: This type stores data as key-value pairs. A value can contain any data type and store any number length of values. This value is identified by a key, which is used as an index. The storage concept of a key-value pair can be a hash table, which is similar to a database with two columns (key and value) but is inefficient in query and update operations (Bansel *et al.*, 2016; Grolinger *et al.*, 2013; Nayak *et al.*, 2013). In particular, this model can be used in forums and online shopping. Popular examples of this model include social networks, Riak, Voldemort, and Redis. The advantage of this model is that new types of data can easily be added to the database as new key-value pairs; its disadvantage is that it does not have a data type, which has a significant impact on the query. Thus, communication with the database is limited to three operations only: put, get, and delete. Additionally, this database uses quick and efficient data management in distributed systems (e.g., Facebook and Amazon).

b) Column-oriented database: This database stores data in columns and rows. Each row has columns called column families (Tauro *et al.*, 2012), which may hold many columns and act as keys for the columns. In addition, a row having the same column is not important because it may have different columns, and each row can be identified by a primary key, which can be a unique row key (Naheman *et al.*, 2013). This model also supports index and query more than the key-value model, and the concepts of its data storage are similar to those of relational databases (Jayathilake *et al.*, 2012). A relational database uses rows to store and process attributes, whereas this model uses columns and does not support joint transactions on tables (Kune *et al.*, 2016). Popular databases of this model include Cassandra, HBase, Google Bigtable, SimpleDB, and DynamoDB.

c) Graph database: A graph database stores data as nodes, which are considered entities (Jayathilake *et al.*, 2012), and the properties of nodes represent the attributes (Kaur *et al.*, 2013). The edges represent the relationships between the nodes. A graph database has powerful speed with related data and this is useful when the information of data has associative information and data have considerable connectivity and relationships. In short, it represents data as a graph, which can benefit social networks. Neo4j is one of the best databases for this model because of its good performance and scalable structure.

d) Document-oriented database: In this database, each document contains the data as a key-value database. The data store the value and may have any data type. The value associated with the key is used to identify the values (Hashem *et al.*, 2016). It can also be suitable for representing complex data and support flexible schemas that can store semi-structured data (dos Santos Ferreira *et al.*, 2013). A document-oriented database stores data in flexible schemas and has consistency, partition tolerance, and master-slave replication (Bhogal *et al.*, 2015). This type of database is ideal for storing and managing big data-sized collections of documents. The sample databases of this model are CouchDB and MongoDB.

In summary, the difference between the types of NoSQL database that exists is that the data is represented by each model (Figure 2.4).

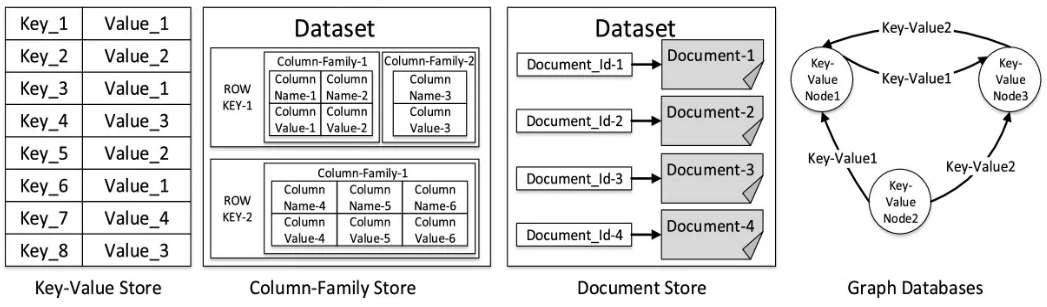

Figure 2.4 Data models of NoSQL database (Grolinger *et al*., 2013)

Many parameters, such as performance, consistency, scalability, and flexibility, should be considered in choosing suitable NoSQL database for applications, as proposed by (C.-H. Lee *et al.*, 2015; Yaish *et al.*, 2013). According to Arora *et al.* (2013), the document-oriented model has high consistency because it replicates data through the master-slave mode. It has high scalability, which can support current Web 2.0 applications. It also uses database collection and a memory-mapped file and thus has variable performance for a document-oriented database. In addition to that, flexibility is high in a document-oriented database because it works with JSON/binary JSON (BSON). It also supports the index and the secondary index for any attribute. Therefore, this study focuses on a document-oriented database.

2.3.2 Document-Oriented Database

Document-oriented databases are renowned NoSQL database for storing large amounts of information in terms of flexibility and simplicity (Imam *et al.*, 2018).

Document stores provide more complex data structures and richer capabilities than key-value systems (Younas, 2019). In document stores, the unit of data is called a document, which is an object that can contain an arbitrary set of fields, values, and even nested objects and arrays. The document-oriented database supports search and indexing by document field and attribute. Some of them can even support queries with constraints, aggregations, sorting, and evaluations (Younas, 2019). Unlike key-value stores, document stores generally support secondary indexes, nested objects, and lists (Younas, 2019).

Colombo *et al.* (2019) focused on document-oriented databases as hierarchical records and denoted documents whose fields either specify a primitive value or are, in turn, records composed of multiple fields. The documents are partitioned into collections, which are then grouped in a database. Typical applications of document-oriented databases include event logging and content management.

As stated by Bathla *et al.* (2018), one of the advantages of document-oriented databases is the ease of adding new attributes to some documents. This feature is different from a relational database's fixed schema structure, where each new attribute is to be added to all records; if the values are not known, then many null values will be added.

Document-oriented databases have many databases that can be used as database management systems. This study focuses on MongoDB databases to migrate from relational databases to document-oriented databases. The reasons for selecting MongoDB are discussed in this section.

Yaish *et al.* (2013) stated that MongoDB can be chosen for applications that have similar properties as those of relational databases. MongoDB is an open-source database with a flexible schema, which means it can accept any field, and having the same fields or structures is not important; it also accepts any data type for the common fields (Rafsanjani *et al.*, 2009). It stores data in BSON, which enables binary serialization on data (K. S. Kumar *et al.*, 2017). In addition, MongoDB is horizontally scalable and features rich queries and an embedding model; it reduces workload, executes complex queries (Lombardo *et al.*, 2012), and offers additional functionality.

MongoDB has many other features (Padhy *et al.*, 2011), such as support for full and secondary indexes, query use of JSON object to deal with data, and use of MapReduce to query complex operations and aggregation operations through JavaScript functions (Ruflin *et al.*, 2011). Also, MongoDB can scale horizontally through database sharding in cases with the heavy workload by automatically splitting the large database to many tables and processes on multiple servers (R. Kumar *et al.*, 2015). Furthermore, MongoDB can accept any kind of data structure, such as event, time, geospatial, and other data types.

Additionally, document-oriented databases work on the master-slave storage architecture and fault tolerance, which is the main feature of MongoDB (Bathla *et al.*, 2018). MongoDB allows data to be organized in the form of collections and not on tables. It supports the querying of specified records from this collection. Many programming languages are supported by this storage system (Bathla *et al.*, 2018). It is widely used for storing healthcare records in the form of documents (Kaur *et al.*, 2015). It can also provide higher throughput than many other databases.

Karnitis *et al.* (2015) evaluated the performance of SQL, MySQL, CouchDB, Couchbase, MongoDB, and PostgreSQL. The result showed that CouchDB performs well but has two problems: data modeling and lack of support for embedded documents. MongoDB is a suitable database model for applications that require good performance and have similar requirements to those of relational databases.