INVESTIGATION IN MYSQL DATABASE AND NEO4J DATABASE

ZAHRAA MUSTAFA ABDULRAHMAN AL-ANI

JUNE 2015

INVESTIGATION IN MYSQL DATABASE AND NEO4J DATABASE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF
ÇANKAYA UNIVERSITY

BY

ZAHRAA MUSTAFA ABDULRAHMAN AL-ANI

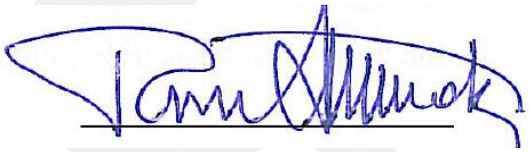
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF
MATHEMATICS AND COMPUTER SCIENCE\INFORMATION TECHNOLOGY PROGRAM

JUNE 2015

Title of the Thesis: **Investigation In MySQL Database And Neo4j Database**

Submitted by **Zahraa Al-ANI**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.

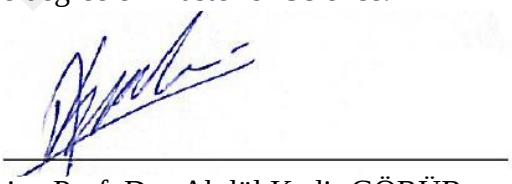


Prof. Dr. Taner ALTUNOK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Billur KAYMAKÇALAN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.



Assist. Prof. Dr. Abdül Kadir GÖRÜR
Supervisor

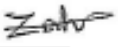
Examination Date: 18.06.2015

Examining Committee Members

Assist. Prof. Dr. Özgür Tolga PUSATLI	(Çankaya Univ.)
Assist. Prof. Dr. Abdül Kadir GÖRÜR	(Çankaya Univ.)
Assoc. Prof. Dr. Fahd JARAD	(UTAA Univ.)

STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name	:	Zahraa Al-ANI
Signature	:	
Date	:	18.06.2015

ABSTRACT

INVESTIGATION IN MYSQL DATABASE AND NEO4J DATABASE

AL-ANI, Zahraa

M.Sc., Department of Mathematics and Computer Science

Information Technology Program

Supervisor: Assist. Prof. Dr. Abdül Kadir GÖRÜR

June 2015, 48 Pages

Currently, there are two major of database management systems which are used to deal with data, the first one called Relational Database Management System (RDBMS) which is the traditional relational databases, it deals with structured data and have been popular since decades from 1970, while the second one called Not only Structure Query Language databases (NoSQL), they have been dealing with semi-structured and unstructured data; the NoSQL term was introduced for the first time in 1998 by Carlo Strozzi and Eric Evans reintroduced the term NoSQL in early 2009, and now the NoSQL types are gaining their popularity with the development of the internet and the social media. NoSQL are intending to override the cons of RDBMS, such as fixed schemas, JOIN operations and handling the scalability problems. With the appearance of Big Data, there was clearly a need for more flexible databases.

In this study, a theoretical study of investigating in two types of databases MySQL one of the traditional relational databases, and Neo4j one of the graph databases. First of all, choosing these two types of the databases according to their features,

both of them depending on the replication and sharding in their systems. Secondly, a brief background with literature review will introduce the opinions for some researchers with some concepts. Thirdly, mentioning relational database and NoSQL in generally and MySQL and Neo4j in specifically, then try to make a comparison between the features for both of them. Moreover, all the websites, the researches, and the journals are the resources for this study. In addition, this study is like mentioned before is a theoretical study so there was no implementation work. However, the comparison presented the result in this study. All in all, we try to provide an understanding for MySQL and Neo4j, that leads us to find both of MySQL and Neo4j have their features, pros and cons that differ from each other.

Keywords: RDBMS, NoSQL, MySQL, Neo4j, Replication, Sharding.

ÖZ

MYSQL VE NEO4J VERİTABANLARINDA İNCELEME

AL-ANI, Zahraa

Yüksek Lisans, Matematik ve Bilgisayar Bilimleri Bölümü
Bilgi Teknolojisi Programı
Tez Yöneticisi: Yrd. Doç. Dr. Abdül Kadir GÖRÜR

Haziran 2015, 48 Sayfa

Halihazırda verileri ele alacak iki önemli veritabanı yönetim sistemleri mevcuttur. Bunlardan ilki geleneksel ilişkisel veritabanları olan İlişkisel Veritabanı Yönetim Sistemi (RDBMS)'dir. Bu sistem yapılandırı verileri ele alır ve 1970'lerden beri popülerdir. İkinci önemli veritabanı yönetim sistemi ise Yapılandırılmış Sorgu Dili ve Daha Fazlası veritabanları (NoSQL)'dır. Bunlar yarı-yapılandırılmış ve yapılandırılmamış verileri ele almaktadırlar. NoSQL terimi ilk kez 1998 yılında Carlo Strozzi tarafından ileri sürüldü ve Eric Evans tarafından da 2009 yılının başlarında yeniden kullanıldı. Ve şimdi internetin ve sosyal medyanın gelişmesiyle NoSQL tipleri popülerliklerini kazanmaktadırlar. NoSQL sabit şemalar, JOIN işlemleri ve ölçeklenirlik problemlerini idare etme gibi RDBMS'nin eksilerini bastırmayı hedeflemektedir. Big Data'nın varlığıyla, açıkça daha esnek veritabanlarına bir ihtiyaç bulunmaktaydı.

Bu çalışmada iki tür veritabanında incelemeye ilişkin teorik bir çalışma yapılması hedeflenmiştir. Bunlar; geleneksel ilişkisel veritabanlarından MySQL, ve grafik veritabanlarından Neo4j'dir. Öncelikle, bu iki tip veritabanı özelliklerine göre seçilmiştir. Her ikisi de sistemlerindeki kırılma ve replikasyona bağlıdır. İkinci olarak, bazı konularla birtakım araştırmacıların fikirleri literatür derlemesine sahip kısa bir özgeçmişle sunulacaktır. Üçüncü olarak, ilişkisel veritabanı ve NoSQL'e genel olarak, MySQL ve Neo4j'ye spesifik olarak değinilecek ve bunu takiben her ikisine ilişkin özellikler arasında bir karşılaştırma yapılmaya çalışılacaktır. Dahası, büyün websiteler, araştırmalar ve yayınlar bu çalışma için kaynak oluşturmaktadır. Buna ilaveten bu çalışma daha önce bahsedildiği gibi teorik bir çalışma olup, herhangi bir uygulama çalışması yapılmamıştır. Ancak, karşılaştırma çalışmadaki sonu göstermiştir. Sonuçta hem MySQL hem de Neo4j'nin birbirlerinden farklı olan özelliklerini, artı ve eksilerini bulmamızı sağlayan, MySQL ve Neo4j'ye ilişkin bir anlayış sağlamaya çalışıyoruz.

Anahtar kelimler: RDBMS, NoSQL, MySQL, Neo4j, Replikasyon, Kırılma.

ACKNOWLEDGEMENTS

First and foremost, I bow before Almighty Allah in a deep gratefulness that his limitless wisdom and merciful, granted me enough strength to complete my thesis and I express thanks from the core of my heart to the holy prophet Muhammad (peace be upon him).

I would like to express thanks to my supervisor, Dr. Abdül Kadir for his advice for my research, and I also would like to thank the faculty members at the university of Çankaya, department of Mathematics and Computer Science.

My words cannot possibly express my appropriate appreciation for my best partner in life my husband, thank you for your patience, infinite sacrifices and support.

At Last, but not least, I would like to thank my parents especially my mother and my brothers and sisters for their continuously prayers and unconditional love. This study is dedicated to all of you with love. Thanks to all of you.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ACID	Atomicity, Consistency, Isolation, Durability
BASE	Basically Available, Soft-state, Eventual consistency
CAP	Consistency, Availability, and Partition tolerance
DBS	Databases
DBMS	Database Management System
NoSQL	Not only SQL
RDBMS	Relational Database Management System
SQL	Structured Query Language
WAL	Write-Ahead Log

CHAPTER 1

INTRODUCTION

Edgar F. Codd had been published in 1970; a paper where he uncovered his ideas about how the information could be stored in the large databases, and how could be accessed without knowing the details of how or where mentioned information is stored. His revolutionary ideas gave birth to a new family of products, which known today as relational databases, where it started with IBM in the mid of 80's, and then continued into the 90's with Oracle, Microsoft SQL Server and MySQL [1]. Since the relational databases have become the prevalent choice to keep of financial records, personnel data, manufacturing, and other information.

For many years, people and organizations have been using the relational database for storing the structured data. The data are divided into groups which called tables; each table must identify a primary column, which is used for identifying a block from a row called the primary key, where tables can be related by using a foreign key that is pointing to the primary key. The tables stored the data in a specific units based on the type, size and other database value restriction rules known as constraints. Each unit of the data is called a column, while each unit of the group is called a row; the columns can have relationships, and defined themselves, as the relational database. The main idea of a relational database is to organize data into formally organized tables, so the database user can issue queries to obtain relevant information easily. As technologies improve, the amount of data generated by humans, the Internet, and machines has also increased in a huge way [2].

With the rise of the large Internet systems, with their large amount of the data and requests, there was a need for new solutions for the distribution of the databases. This led to an evolution for alternative within the last years, data management systems called NoSQL have been created, and are becoming more important. The term "NoSQL" is short for "Not Only SQL" and was introduced in 2009, when it was chosen as the title of a conference “for folks interested in distributed structured data

storage" [3]. The term "NoSQL" should not be taken in an illusory way. Rather than replacing the relational database, the NoSQL approach point is to work in applications where there are scalability issues, and where data need some structure. To meet the demand of the enormous increasing in data size, the increasing in a volume of data has also led to increase a process of complex data's volume, and the connectivity of relationships between the data [3].

There are four emerging categories of NoSQL. The first one is the Key-Value stores, which is based on a research paper published by Amazon and called Dynamo. Because of the huge amount of data, Dynamo was built by Amazon, a Key-Value store's database, to solve the scalability problem; the data model of the key-value stores is similar to the hash table. The second category of NoSQL is the Column Family or the Big Table clones. Similar to Amazon Dynamo this was the purpose of creating Big Table, which is to solve the problem of scalability. The main difference between Dynamo and Big Table is that each individual column in Big Table can have its own schema. The Big Table of Google is a powerful tool to capture semi-structured data [4]. The third category is documented databases, and the purpose of this category is the same as others that mentioned before, solving the problem of scalability. The most famous implementation of a documented database is CouchDB, which was inspired by Lotus Notes, and the document is typically expressed in JavaScript Object Notation (JSON) or in eXtensible Markup Language (XML) format. The fourth category of NoSQL is a graph database. The Graph theory inspires all graph databases. The data model of the database is similar to a documented database. The Document database has JSON documents, and the documents are key-value pairs. These documents are called nodes in a graph database. Nodes can consist of key-value pairs [5].

1.1 Motivation

Recently, there are wide changes that are ongoing within many companies; some of them said that the changes are making the way open for great opportunities. With the increase of the popularity of social networking and the large volume of data that related with it, there was a necessary need for a new type of database, which can store and manipulate complex data which is NoSQL. Our motivation of this thesis was to understand a new type of database. The major objective of this thesis is to review an emerging technology called Neo4j and compare it to an existing relational database.

1.2 Aim of the Study

This research aims to provide an overview about the relational and the non-relational database with their backgrounds and characteristics of each of them, specifically Neo4j one of the graph database in a comparison with MySQL one of the RDBMS in the terms of availability, scalability and the cluster that is used in both of the databases. It also describes their concepts that form the base of the RDBMs and the NoSQL databases like CAP theorem, ACID and BASE.

1.3 Scope of the Study

The scope of this thesis is to investigate between two types of databases, MySQL and Neo4j. By mentioning the features of MySQL and Neo4j that is stated by other researchers.

1.4 Thesis Question

What are the features of MySQL and Neo4j and which is suitable to deal with the data?

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

Introduction

In the literature review, we introduce various concepts that presented over the years by E. F. Codd and others researches, the concepts of normalization, the CAP theorem, the ACID properties, and other concepts that have plays the role of basis in RDBMS databases. The concepts helped the developers of creating the NoSQL database. The NoSQL databases have different advantages based on their data model structure and there are various types of the NoSQL databases that each one of them stores the data differently.

2.1 Data

In the world of computing and processing data, the data are the featured information that is translated usually into special models that are suitable for processing. The data can be any forms that are performed by the operations of the computer, such as characters, symbols or signals. The data can be structured in graphics with a group of connected nodes, or like a tree with nodes, that is, having a relationship like parent-child or as tables with columns and rows [6].

2.2 Database

Databases can be defined as a set of data that has been organized in order to facilitate the data retrieval process in an efficient manner. As is common, the database is the term usually used to refer to the entire database system, but in fact it refers to the collection and to the data. The database management system is the naming calling on system that deals with the storage and retrieval and data modification. Linked lists

have been used early in the process of creating relationships between data and modify the selected data. However, these models were not uniform and require extensive training to use it efficiently [6].

2.3 Relational Database Management System

To store and access the data, was always been challenging. It is very important to organize the store information in a method that makes the task of managing and accessing data easy. The future performance of the system depends entirely on how to build the system in the first phase. Association and displaying of data plays the important role in how easy or difficult it is to manage the data. In late 1960, use the hierarchical data model to process information stored in the form of a tree structure. An example for hierarchic system is IBM's Information Management System (IMS). There is another model used alongside with the hierarchical called the network data model. The idea of the relational model presented by Codd in 1970, where the relational model emphasized on the idea of organizing data in the form of two-dimensional table, so-called "relationships" [7]. A relationship consists of a set of domain, known as attributes or column having tuples with data for each domain, and can be a set of rows and the idea of the establishment of the relational model for multiple purposes. The relational model has been designed in order to address the following issues:

1. Hide from the users that how the data is organized in a machine.
2. Protect the users' activities and the programs of the applications from the internal data changes and increasingly the data types.
3. Remove data inconsistency.

The main goal of designing the relational model is to store information in databases using relationships. This means software that allows storing, modifying, and accessing of the information that's stored in a computer system (Server) [8]. This model is also used by other types of software such as symbol table, which is in the form of graph structure used by interpreter and compilers. In the relational model

analysis reveals that it has certain types of the relational model. They are identified as relation, domain, entity, tuple, attribute, and attribute value. These integrated components defined structure chart in the relational model [9].

Relational databases store definite relations wherever entirely tuples names and forms must be defined in advanced. Each row in the table must contain the same number of columns and specifications of these columns must already know. Properties are typically unique, indexes, null able values, keys, etc. In addition, every relationship can also contain rows called foreign key which refers to the primary key of the other relationships, which creates a link between the two different relationships. This means, complex queries can be made through multiple relations, resulting in new relations that are based on the query [6].

SQL is a programming language for special purposes have been designed to manage data in a relational database management system. The Traditional RDBMS guarantees that transactions implemented to sustain ACID properties before they are persevered. This ratio is easy to handle in the central environment, but soon it becomes difficult to assets upon scaling the horizon for the entire database to a few servers or distributed, leading to the loss of performance according to strict consistency model [1].

2.4 Structured Query Language (SQL)

Many advantages the relational model has which make it popular and wanted once it was released in the IT industry. For accessing the database there was a need for an interface, which it was a language that it start from sub-language and had passed through several phases to become the modern Structured Query Language (SQL).

Structured Query Language (SQL) is an interact language that are deal with the RDBMS and the object-relational DB [9].

Codd has presented the model of relational data through his research in the laboratory of the IBM; however, the language of SQL is a coherent dramatically with the appearance of the relational database model. SQL was having a history behind its

reputation; it is a cornerstone of any DBMS system. The first developed for SQL was by Chamberlin and Boyce during 1970's. SQL is the cornerstone for any system of DBMS; firstly, it was called Structured English Query Language (SEQUEL), and it was designed for manipulating and retrieving the information in the IBM's prototype relational database system. Later, the language name has been changed because another company has registered their proprietary to the acronym "SEQUEL" [1].

In very fast time, SQL achieved the actual role of the language for the RDBMS as it designed from the top sellers for their systems that the SQL become the standard language. Because of the popularity of the SQL, it has raised the need for a standard to be followed by everybody. In the 1980's, The American National Standards Institute (ANSI) and the International Standards Organization (ISO) had worked together and the first version was developed of the SQL standard under the name of SQL-86 (SQL1) and there is a strong relationship between RDBMS and SQL.

As mentioned previously, Codd has presented a relational data model that fits most forms of data storage structures. He also described how to interact with relational databases using the relational model. SQL aims to reach users who wish to process of data retrieval using the English language than they are retrieving this data using mathematical symbols. One of the main reasons behind the emergence of the SQL is the high cost of software development as well as to enable ordinary users from dealing with databases. The software development process is not based on the initial development stage, but it requires the development and maintenance of software continuously to keep up with the changing requirements of the users. Since most software is used by users who have simple or no experience with the use of computers and software, so it requires the developers to simplify operations to use this software to suit the level of awareness of these users. The benefits of the RDBMS are to support enterprise information systems that built on a low-cost hardware. The RDBMS is one of the most important information storage technology since its inception, where are implemented on a large scale in various industrial and commercial fields [6]. The RDBMS has been of great popularity and significance, till the appearance of what is known as cloud computing. The RDBMS scenario has changed since cloud computing penetrated the software industry. This puts a high

ultimatum of working with truly huge amounts of data with a painless scalability selection. The required from the RDBMS is to provide better scalability when deals with the large data. Nevertheless, with following ACID rules it is a big obstacle for the RDBM systems to achieve this task. A locking mechanism used by the RDBMS to guarantee the data consistency, however, this causes scalability which lead to being a problem in a distributed environment. The web-based services caused further deterioration and in this situation it makes it harder to achieve consistency, availability, and scalability at the same time [9].

2.4.1 SQL Statements

As a typical relational database, in MySQL the data are being organized in the relational model in tables, rows and columns, and to access to the databases use SQL. For processing the data MySQL was provided with so rich group of other statements. The basics are INSERT, SELECT, UPDATE, DELETE, which is matched with CRUD operations (create, read, update, delete), where CRUD is an acronym that stands for "Create, Read, Update, Delete". These are the four basic operations for managing the data in an application. Besides, other functionalities are being supported by MySQL like group by, join and views for collection data through multiple tables or stored procedures, triggers, functions and activities that can be worked depending on the user's requests or a specific schedule. Nevertheless, joins in relational database might be slow the system continued to be ridiculous, this happens when millions of users are searching against tables consist of millions of rows of data. To store images, documents, files in the relational database will make the data useless because it can't process, it is just stored as binary data. The tables in MySQL have a constant structure and can't be differ with the data, thus have to insert null values and that leads to inefficiency [10].

2.4.2 Normalization

The technical outlook behind the strategy of a relational database is to make sure that the planned relational is normalized. Normalization can be defined as the process of organizing and categorizing of data in the database [11]. Throughout the normalization there two goals to achieve, first is to protect the data and make the database more flexible by eliminating the dependency of asymmetric as well as the elimination of duplication of data. The above objectives can be achieved through the establishment of a relationship between database tables according to the following laws and rules:

1. First Normal Form (1NF): Frequently denoted as 1NF in applied applications, this model provides the main guidelines for the rules of the organization of data as follows:
   - Elimination of frequent data through the creation of separate tables for each set of relevant data.
   - Determine each group of interrelated data sets with a primary key.
2. Second Normal Form (2NF): is written as 2NF, this model is as well confirms the idea of eliminating duplicate data as follows:
   - Fulfill all the requirements for the First Normal Form.
   - Create separate tables for a set of values, if they apply to multiple records.
   - Linking these tables by a foreign key.
3. Third Normal Form (3NF): it refers to 3NF in practical application, the third Normal Form set the following rules:
   - Fulfill all the conditions that are required from the Second Normal Form.
   - Eliminate the fields that are not depending on the primary key from a table, and place them in another table if it is needed.
4. Boyce-Codd Normal Form (BCNF or 3.5NF): The Boyce-Codd Normal Form which is also named "third and half (3.5) Normal Form". It appends one more rule to the 3NF to address an anomaly, not treated by 3NF:
   - Meet the requirements of the Third Normal Form.
   - Each determinant should be either primary key or a candidate key.

5. Fourth Normal Form (4NF): written as 4NF in practical application, Fourth Normal Form sets the following rules:
   - Fulfill the whole requirements of the Third Normal Form.
   - Two or more multi-valued attributes should not be included in the same relation; it should not have any multi-valued dependencies. Multi-Valued dependencies are occurring when the existence of one or more rows in a table means the presence of one or more other rows in that same table.
6. Fifth Normal Form (5NF): Likewise denoted as 5NF, Fifth Normal Form sets the following rules:
   - Fulfill all the conditions of the Fourth Normal Form.
   - Every non-trivial join dependency is included in the table by the candidate keys.

It should be mentioned that for the database scheme in order to satisfy 2NF, it should fulfill the requirements of 1NF. Also, to satisfy the requirements of 3NF it must satisfy the requirements of 2NF. The 4NF and 5NF are rarely considered in the operational design. Consequently, the database considered a normalized database if it fulfills the first three normal forms [11].

2.5 Literature Review

When Carlo Strozzi describe the NoSQL database for the first time mentioned it by "A lightweight, open-source and non-relational database that did not expose an SQL interface, and the NoSQL departs from the relational model altogether; therefore it should have been in more appropriate to called 'NoREL', or something to that effect"[6]. While Michael Stonebraker was having another definition for the NoSQL and defined it as a movement; the movement of NoSQL is a modern approach to the continuation of the data by using novel storage approaches. The NoSQL databases providing the flexibility and the performance as well as they are not limited by the traditional relational databases approach to the data storage. Each type of the NoSQL database is been designed to store a different and specific type of the data and the technology of each type does not impose to the data to be limited by the relational

model, but attempts to make the database (as much as is feasible) compatible with the data it wishes to store. The NoSQL movement itself has become very popular because of the large widely known and successful companies creating databases storage implementations for their services, all of which are not of the relational model [12].

Aaron Schram and Kenneth M. Anderson have an opinion which is "NoSQL is a term that is used to describe a wide class of technologies that is providing an alternative approach to data storage comparing it with traditional relation database management systems. Usually, these technologies are providing the user with low-cost solutions for the problems of high availability and scalability for losing of a flexible query language, which make ad hoc access of the data generally more difficult. The term NoSQL was initially meant to make this clear by standing for “no SQL” but the term has more recently been updated to mean “not only SQL” as very few production systems eliminate relational databases altogether"[13].

The term of NoSQL database was chosen generally for a specified class of non-relational data stores. Most of such databases do not use SQL as their query language, therefore the term of NoSQL is confusing and the community of the database is explicating it that means “not only SQL”. Sometimes the term post-relational is used for the NoSQL data stores. This opinion is by Jaroslav Pokorny [14].

According to the White Paper of Datastax (2012), that the term “NoSQL” is in sometimes gets abused by different software vendors and the professionals' technology and misused, in generally, the NoSQL refers to advanced engines for the data management that exceed the legacy of the relational databases in satisfying the needs of the modern business applications today’s. The NoSQL is having a data model that is very flexible, horizontal scalability, distributed architectures, and the uses of the interfaces languages which are “not only” SQL typically characterize NoSQL technology [15].

Guinard, Trifa, Pham, and Liechti their opinion about the NoSQL was "the NoSQL databases are consisting of a family, which is generally trying to provide a high synchronous of read-write, access to the data and efficient comprehensive storage for

the data, scalability and high availability. usually, the main features of the NoSQL databases are a huge amount of the data, easy replication support, no existence for ACID, simple API and schema-free, thus, it’s worth for taking an account that NoSQL databases don’t natively support ACID transactions, and also these databases can compromise the consistency unless the manual support is provided [16].

Stefan Edlich and Achim Friedland say about the NoSQL in their definition, "The definition of the NoSQL database can be extended with other views, by adopting the four criteria which they are non-relational, distributed, open-source and horizontal scalable. With the addition the NoSQL system has weak schema constraints or is schema free, shared-nothing means there is a simple data replication (the nodes are independently from each other), the NoSQL provides a simple API and with the consistency model there is not ACID anymore. However, the NoSQL databases sometimes use some of these criteria and not necessarily to use all of them. Generally, NoSQL databases don't use JOINs or constraints anymore. Also, the NoSQL databases do not aimed to deal with the complex queries over the multiple tables. The best architecture for such queries is still the relational model"[17].

Neal Leavitt had mentioned what he see in IEEE Computer Society about the NoSQL "the non-relational database which is called now NoSQL database is gaining their popularity and start to gain the market tractions. Due to the fact that most of various NoSQL databases takes various approaches depending on the way for storing the database, and what they have prevalent is that they’re not a relational database. Their primary advantage, unlike the relational databases, is that they are dealing with the unstructured data such as word-processing files, e-mail, and multimedia efficiently. Also the NoSQL databases are in sometimes faster than the relational databases because their data models are simpler and they don’t have all the technical requirements that relational databases have, the supporters of the NoSQL says that most major of the NoSQL systems are flexible enough for better enable so the developers can use the applications in ways that meet their needs. NoSQL databases face several challenges, which they are Consistency, Limited eco-structure, Overhead and complexity, Reliability and Unfamiliarity with the technology"[18].

Rick Cattell in his paper put six important key features that for the NoSQL systems and they are:

- The possibility for a horizontal scaling data “simple operation” passed on across many servers.
- The possibility to replicate and to partition the data across many servers.
- A simple call level interface in contrast to a SQL binding.
- A weaker model of consistency than the ACID transactions in most relational database systems.
- The possibility for adding dynamically the new features to the data records.
- The effective usage of the distributed indexes and the random access memory for the data storage [19].

2.6 NoSQL Systems

It can be actually confusing and annoying to understand some principles of the non-relational database when it comes for the people who understand and have strong experiences in relational databases. NoSQL systems can be more complex than traditional databases, and in other ways they are simpler. Because they are schema-less. They are not based on a single model like the relational model of RDBMS and each database, depending on their target-functionality, adopt a different one [18]. NoSQL systems can scale much bigger group of the database. The data model in NoSQL databases, in general, is more limited than in what it is in the SQL databases [20].

The term "NoSQL" was first appeared in 1998; it used to describe a trivial, open-source relational database developed by Carlo Strozzi that provided no form of the SQL language for querying. This use remains relatively limited, but it's not related to some extent to the development of the NoSQL known at the present time. The term of the NoSQL has emerged in use for only nine years; it appeared in the year 2009. To discuss new database technologies, there was a meeting up group for promoting of choosing the term of NoSQL as a twitter hash-tag in San Francisco and the meeting up was organized by Johan Oskarsson, a developer visiting from London

while the term was suggested by Eric Evans, a developer at Rack-space. The term was only meant for having a short lifespan. The market needed a term to describe the new technologies, and with the big explosion for the new databases, such as Cassandra, MongoDB, and CouchDB that followed in the wake of Google's Big table and Amazon's Dynamo. This mainly means that the term was to describe the movement of the non-relational database that has its assets in marketing. This basically means that the term came to describe the movement of the non-relational database that has its assets in marketing. All NoSQL solutions providers generally agree on only one thing which is the NoSQL term is not perfect, but it is attractive Most agree that the "No" symbolizes for "Not only" is accepted, the point of it is not to reject SQL, however, instead of that to recompense for technical limitations that is shared by the majority of the applications of relational database. Actually, NoSQL is more refusals for specific software and hardware architecture of databases than any single a language, product, or technology [9].

The main reasons to use NoSQL databases are their flexibility, speed, scalability, and high availability, so according to these reasons, NoSQL has the opportunity in the manufacture, but it is still not mature enough and shortage to the standards. For any of the new databases, you may have Pig, Hive, SPARQL, Mongo Query Language, Cypher, or others. These languages have little in common. The NoSQL system meets the companies' requirements that are dealing with terabytes of data. For example, a typical PC probably had 10 gigabytes of storage in 2000. Amazon the world's biggest retail store is dealing with more than 42 terabytes of data. However, in the present time the Facebook swallowed 500 terabytes from a new data for every day; the spreading of the Smart phones, the data that they are create and consume; will lead soon in billions of new and constantly updated the data, feeds containing environmental, location, and other information, including video. In the mostly the data is stored in a distributed environment and don't need for schemas or join operations to recover the data. There is no principle for Primary or Foreign keys in the NoSQL database since it doesn't store the data in tables and has no constraints. Due to the reason that NoSQL does not store the data in tables so it does not contain a restriction and concepts of foreign and primary keys. This data avoids the link

operations and does not require the storage of a fixed table scheme and do not try to provide the characteristics of ACID [21].

2.7 Architecture of NoSQL

The basic concept of NoSQL is to provide high-performance support for the provision of direct access to programming languages, which manages databases from the application layer. The NoSQL user can manage both data and applications from the application layer. Direct access to the database from the application layer provides a very great flexibility to run applications. In the relational database, the database can be managed only from the database layer and not from the application layer. This feature gives NoSQL great flexibility to manage applications in addition to the databases. Data management process is managed in relational databases through normalization. Normalization is the process for managing the fields and the tables of the database in this way the dependency of data and redundancy can avoid. Through normalizing the data in various tables, the relational database management system (RDMS) can maintain the key constants, which make it easy to maintain the consistent data. Normalizing the data of large dataset can have performance problems, because of the queries of the aggregate information which can be complicated with lots of JOIN type query. However, NoSQL does not have ad-hoc JOIN type of query functionality. In NoSQL a user can use the software patterns as MapReduce to handle large volumes of parallel data. NoSQL de-normalizes the data; as a result, we can have significant enhancement in the inquiry time to crash a database because less overhead while querying the database. The NoSQL avoids the aggregate and JOIN operations. Since de-normalization of data, there is a coincidental of data inconsistency, such as redundant or duplication data. To avoid duplication, the application layer must rely on data synchronization which avoids any inconsistency in the process of copying data. Also, a schema is another difference between relational databases and the NoSQL. The scheme should be defined before adding any field when dealing with the relational databases. The scheme is added to the database when you add records to the database [22].

2.8 NoSQL Categorization

The term of the NoSQL does not indicate specific types of data, but can be divided into types of non-relational databases that have different characteristics and suited to different types of data. With the high speed of the technology in this world almost every company tries to save their data and put it in big database which can contain all the information about the company, the information of the employees and the security of the web of the company, etc., in a short time and with a specific amount of money to fit the big data in the database. It is not possible to say that the NoSQL replace the relational databases, but it can be considered as a good and efficient solution for various types of business. We say that the SQL replace relational databases, but can be considered on it's considered a good and efficient solution for various types of business. We have to look at the data and then choose the best model for dealing with them instead of having a single model. Today existing relational databases provide important tools and mature to deal with different types of data, however, there are different needs of each application. Therefore, NoSQL databases are classified into four main categories [23]:

2.8.1 Key-Value Stores:

The key value Data store is completely free schema manner; it is effective, simple and powerful model. It is easy to implement, but this easily probable to be an obstacle because it becomes difficult when implement complex data structures through it. This approach is used in applications that store data in environments free of schemes such as hash table that contains the value for the key where the keys are used as indexes, and this make it faster than RDBMS. The key is defined by the programmer or it can be generated by the system. It is a similar mechanism for maps or dictionaries where by unique keys the data addressed and the values are arrays of continues bytes, where they are fully ambiguous to the system and the keys are the only way to restore the stored data. The data involves of two parts the first part is the key which consist of a string and the second part is the actual data which is referred

to be as value and the two parts creating a key-value pair. Stored data type depends on the programming language used and can be any data type that supported by this language like integer, float, string and object.

Data that is stored in a format such as this reduces the need for the existence of structured and fixed data. Any kind of values can be added as a new value at the run time and without a struggle from other stored data and without any effects on system availability. Gathering key value pairs into a group is the only possibility that is offered to add to the data model some kind of structure. High concurrency, mass storage with options and fast search are provided by key value stores. The first to use the key value structure is Amazon to resolve gradualism and availability issues using dynamo storage systems. Key value stores such as Dynamo are efficient in environments with the caching and the Distributed Hash Tables in order to enhance the high read performance and consistent hashing to partition its key space across its replicas and to ensure uniform load distribution [24].

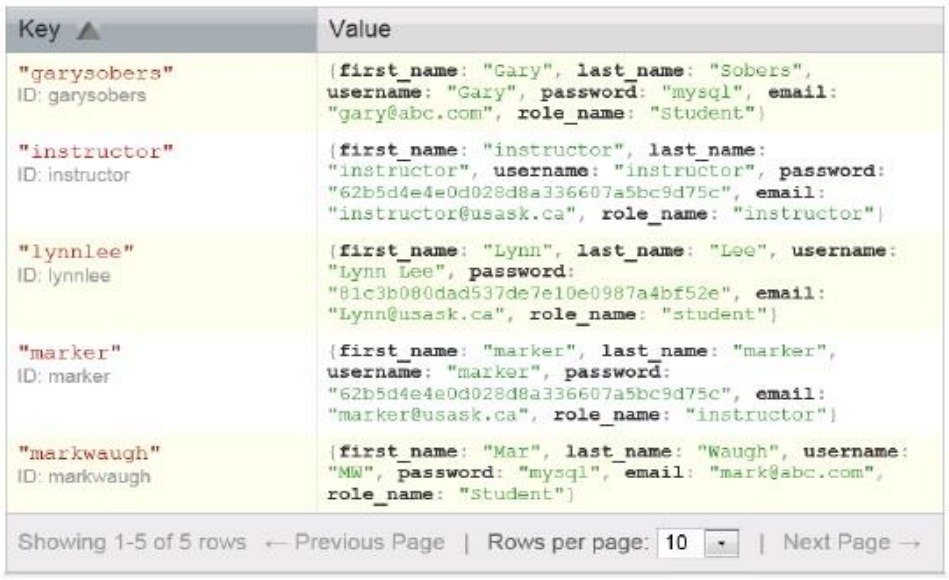


Figure 1 Key-Value Store [36]

2.8.2 Columnar Databases:

It is useful in the queries of MapReduce for allowing higher performance by hugely parallel handling. In columnar databases, data are stored in columns instead of rows, data are stored in a pair key value but the key is two-dimensional. In this structure requires the column and row key to access the stored data [25]. It can also be added the key as part of the timestamp as used in large scale Google. Facebook messenger also used HBase in order to support billions of messages every day. Row by row approach used in this design is saving all the information together in a single independent entity. From other hand, column by column are storing all the information with each other about the attribute.

In order to understand the rules of columnar database structure, there are many conceptions to be learned. These conceptions comprise a column family, super column and column as follows:

2.8.2.1 Column

It is a data container and it is considered the most basic unit with a pair of key value. Google big a table also includes a time stamp.

2.8.2.2 Super Column

It is a tuple that contains a name and a value. SuperColumn does not contain a time stamp, such as what exists in the ordinary Column tuple. Value part is a map containing the number of columns and be linked to the name of the column. SuperColumn looks like a catalog or be in the form of a set of other columns combines with certain many columns. Grouping multiple columns using SuperColumn de-normalizes the whole row into one SuperColumn. Data stored discretely in a column family as a row can also be assembled within a SuperColumn family as a SuperColumn. This strategy improves the search process for this data instead of using the value row key [25].

2.8.2.3 Column Family

This method (Column Family) defines how to store the data on the disk. Super columns or Columns it may be a part of the column family. Columns keys are grouped to form a group named "column families". The column family is considered as a peer to the table in a relational database. Columns are defined from certain types of data in a relational table, and then the data is stored in the form of rows that were inserted by applications. Each row contains the same number of columns. Column family can be defined as a set of rows that contain any number of columns in the sparse form may allow a different set of columns per row. Column family is used in the form of a prefix for each column members in the column family. Column family may be dynamic or static. In a static Column Family the column metadata is predefined for each column while a dynamic Column Family permits the application to fund column names in order to store data [25].

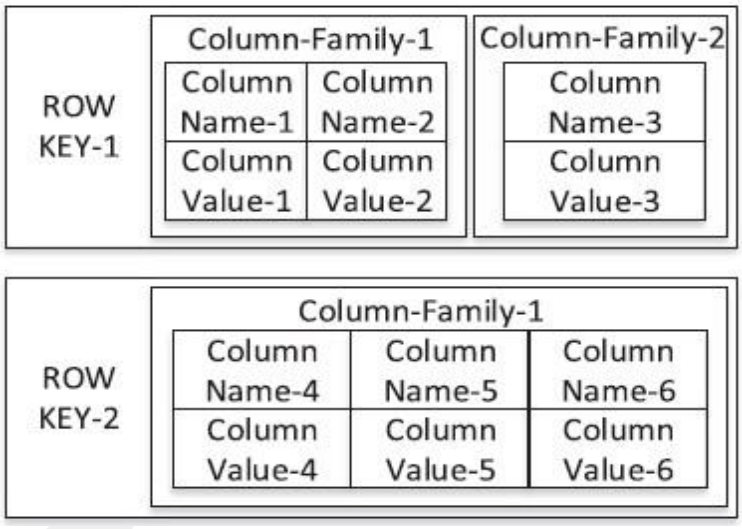


Figure 2 Columnar Database [36]

2.8.3 Document Store

It refers to the databases that store the data in the form of documents. In document stores the key value pairs are packaged within documents and the keys have to be unique. Each document have it is special key "ID", which is unique also and have a collection of documents, thus identifies each document explicitly [24]. On the opposite of key value stores the values are not ambiguous to the system and can query also. Inside the document oriented database the documents are similar to the records in the relational database but they are very much flexible because they are schema less. Document store has achieved attention in the field of NoSQL at the same time they are growing speedily. Document store is a special type of key value store, since they are not storing the document in the form of a block of data and using the data for indexing the document. For that reason there are requirements on the data in the document so it must be organized in a certain way. This achieved by using JSON, XML or something else that the database can be understandable [25].

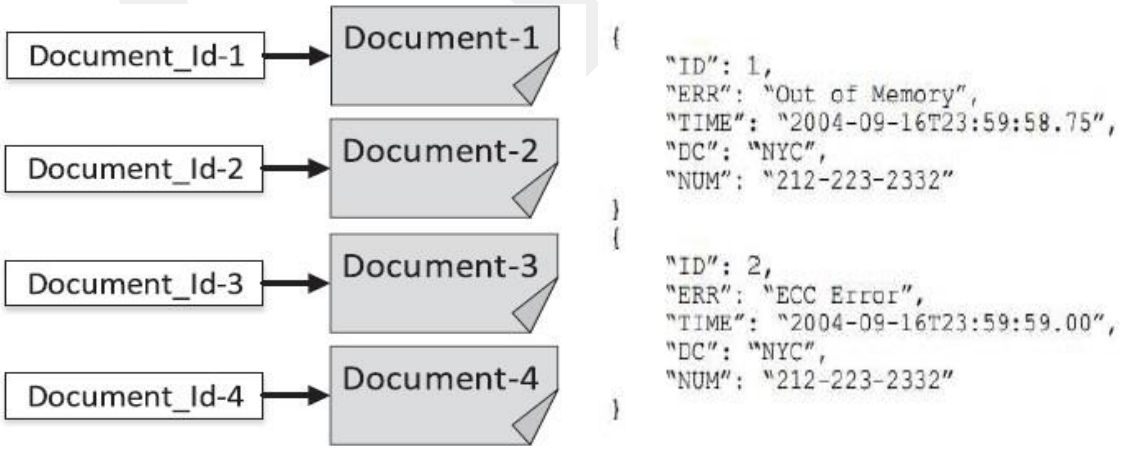


Figure 3 Document Stores [38]

This is usually accomplice by XML, JSON or something else that the database can understand. This allows inquiries on data and not only the keys as is the case of a key value store. It helps us to reach more flexible explanations than the relational

database management system because the database does not contain a scheme. Furthermore, this type does not contain problems in adding attributes after it has been entered into the database even if the attribute is not imaginable at design time. This will facilitate the process of storing data in more flexible, this difficult to achieve with traditional relational database management system, from the famous examples of this type are CouchDB and MongoDB [25].

2.8.4 Graph Database

Stores object with relations, this model is very efficient in certain specific areas such as to calculate the shortest path from one node to another. In this type of database the data is stored as nodes and edges and the data is stored between nodes. Edges have a name and type while the nodes shall be either to have attributes or characteristics. Data is extracted by reversing the edges and nodes in different ways. For easy access to the nodes, some of edges contain certain ways of indexing. That is why there is possibility to identify a specific edge which is applicable only between specific types of nodes [26]. The graphs in the graph database have contained information for each node and edge. The edge can define the relation or the connection between two nodes and the relations having their properties for providing the information about this relation [26]. As well, the nodes have also their own properties. The cost to move between two nodes is calculated by number of jumps (hops) between the nodes, this is because of the primary nature of the graph database. this nature of the graphs make the movement regardless to the data type that is used to store the information and the moving from node to another will remain constant The most important benefits of this type of storage are the possibility to navigate the nodes with identified mathematical diagram of traversing algorithms. As is the case with all the different ways to store data, it only works if the data fit the model. There will be certain problems if the data is tabulated in the nature of the data with a little of relation between the nodes or if totally absent. The storing data concept in something else besides the relational database management system is nothing new. There are many projects for as long as there have been computers. There is an increasing growth of

data in recent years and the need to use other things, non-relational databases management system grows dramatically [27].

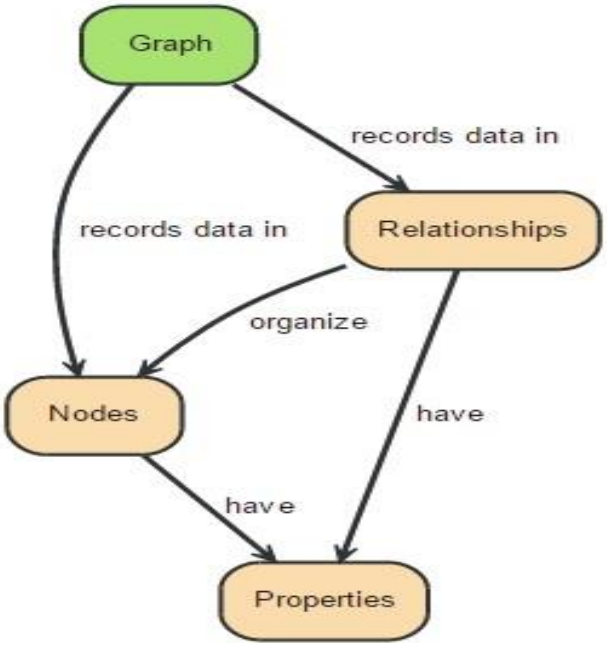


Figure 4 Graph Database [58]

2.9 The CAP Theorem

When there is a need to evaluate NoSQL or any other distributed systems, it will be all about the "CAP theorem". To be able to discuss the different database solutions that exist today, it is important to have an understanding of the CAP theorem. In 2000, Eric Brewer proposed the idea that it consist on three properties, and all of them are desirable for all Web services; but he made the following suspicion " it is impossible in a distributed system can continually maintain perfect consistency, availability, and partition tolerance simultaneously only two of them can meet" . The CAP theorem consists of the following properties [28]:

- **Consistency**

  Any two operations synchronized in the same state see the data at the same time. This has imposed duplication on all the data updates for all nodes in the cluster before writing transaction is completed.

- **Availability**

  It is guaranteed that all requests will answer, in another word the system at all times is expected to be available.

- **Partition Tolerance**

  The distributed system has to continue to work even in the event of a failure of a part of the system.

The CAP theorem is described in sometimes incorrectly as a simple design-time decision, where the fact the theorem allows for the systems to make trade-offs at the run time, for containing different requirements. In 2002, Gilbert and Lynch has proved that Brewer was right for asynchronous network, which the network is where the nodes does not share all the time, but have to rely on the messages that are sent between them. Since this is the case for most web services, it has the main effect on the decision for choosing the right model for storing the data. It is up to the architecture to choose two of three criteria that state by the CAP theorem. Consistency and availability by most relational databases can promise and this is good for a smaller system [29]. The data fits the relational model and there are no requirements for uptime. If this is the main goal, so the relational model is a good choice. If the data is huge or there are requirements on uptime, maybe there is a necessity for partitioning the data between several of nodes and make a compromise on one of the other. One node can never guarantee a given uptime, for some companies it is very important that they can tolerate a database that is inconsistent just to guarantee availability in sometimes. There is an important note that inconsistency isn't always inconsistency, this only means that the database can not ensure that each node have the precisely same picture for the data in all times. But

they do ensure that all nodes at some time will have the same picture, but not all the time. This is referred to as eventual consistency and as the term implies, at some time the database will be consistent, but not all the time [30].

The expression of the CAP theorem has served its purpose; it was to open the minds of the designers for a wider range of systems and trade-offs. Actually, in the last decade, a wide range of new systems appear, as well as much discussion about the comparative advantages for consistency and availability.

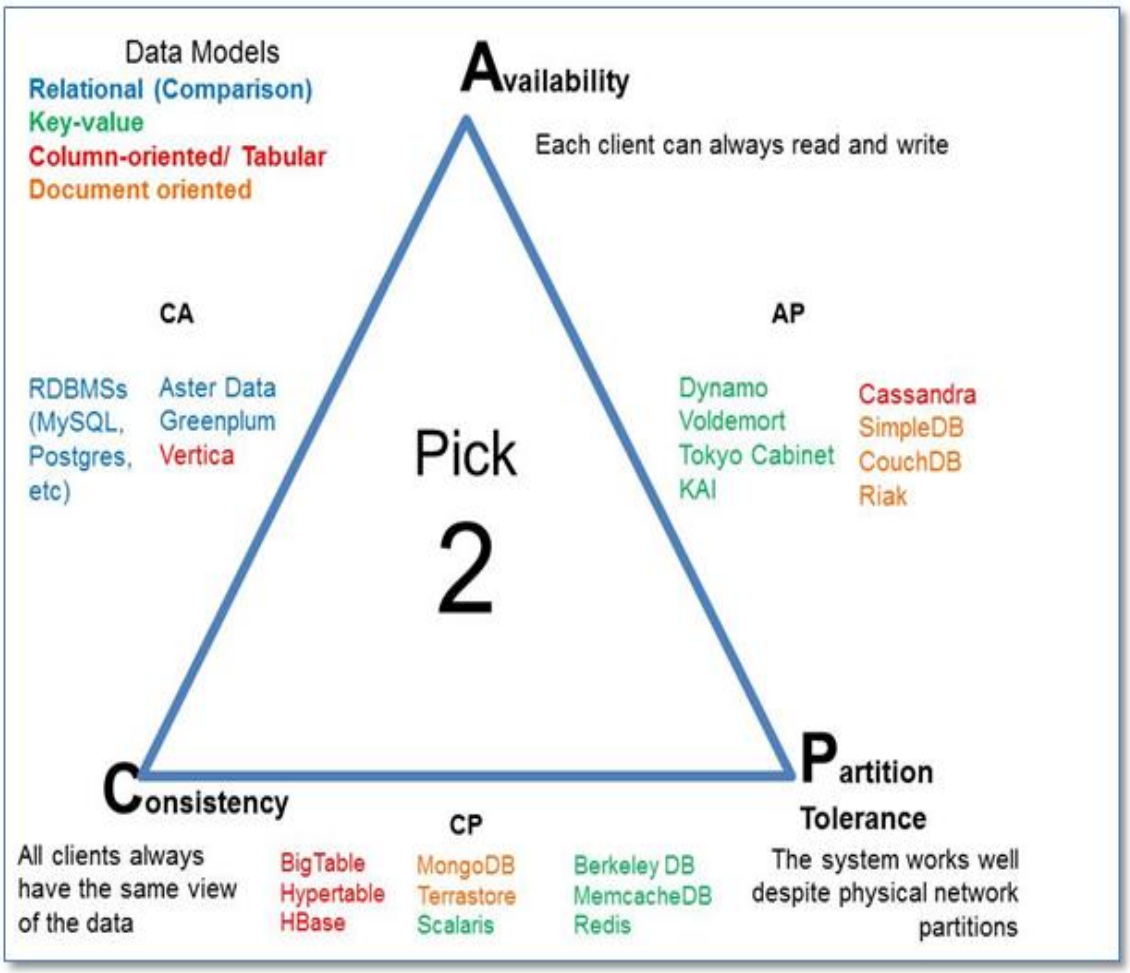


Figure 5 The CAP Theorem [57]

2.10 ACID VS. BASE

If there are requirements that a database needs to be separated then the theorem of CAP conditions that it needs to choose to give up either C (consistency) or A (availability). ACID was been used from The Classic distributed RDBMs, that is why the RDBMs chose consistency and partition-tolerance. In many applications the availability is much more important than consistency. For some database application which that chooses Partition-tolerance and Availability it will be much more useful. Many NoSQL databases have been chose such an approach. There are two different ways to do that by using ACID (Atomicity Consistent Isolation Durability) and BASE (Basically, Available, Soft state, eventually consistent). ACID and BASE are not databases but more of association diagrams that gives strategies to how a database can work to be as good as it can be for the third standard [23].

2.10.1 The ACID Properties

They are referred to the four characteristics to the relational database that guarantee the reliability (integrity and data consistency). Applications that go back to the traditional relational databases management systems have focused on ACID transactions. ACID is a term refers to Atomicity, Consistency, Isolation, and Durability.

- **Atomicity:**
  It means that either the implementation is done for all the units or the steps of the transaction or nothing is executed. It means that, if any unit of a database transaction fails, then the deviations made to other units are trolled back to the inventive situations they were before the transaction started.

- **Consistency:**
  A Consistency property of the database ensures that the database will remain in harmony condition before and after the transaction, regardless of if the

transaction has succeeded or failed. For example, if there is an error has occurred in the process of accounting X moving fund to account Y, the systems will rollback automatically the part that completed of the transaction so that fund is not deducted from account X and fund is not added to account Y. So we can say that success in the completion of the transaction process will mean that all steps have been implemented correctly and that the system will remain in validity status.

- **Isolation:**

  Isolation property in the databases means that the transaction being completed is completely isolated from the other transactions that accomplish a similar task and work at the same time. This means that transactions that operate on the same data do not interfere with each other.

- **Durability:**

  The database durability will guaranty that a successfully finished transaction is dedicated permanently to the database and changes cannot be lost later. Considering the fund transfer from account X to Account Y, once the system confirms that account Y has been credited, the changes persist even if the system crashes [8].

2.10.2 Write Ahead Log (WAL)

The Database has a system that's designed for supporting atomic transactions, have durability, consistency and isolation. Write Ahead Log (WAL) uses in transactional systems to provide Atomicity and Durability which is a technique used widely in the database for transactional. Through a transactional all the changes that are requested are still exist on the disk but are not virtually stored. When the transactional is finished, only the changes of the transactional will really stored in the database and from the temporary place the transactional will be removed. For example, in neo4j, WAL have three files to store, one of them call as a marker file which keep track of

the two other files where they are the current active log file. The transaction when it finished normally, the two active log files will be removed while the marker file will be marked as good file. When the process is a breakdown, the marked file will be referred to as not finished, and the marker file will check the last active log. If the last active log has all the information about the transaction, then it will do a recovery by creating a new transaction. Otherwise, the marker file will do the rollback, where this is mean that it will remove the transaction files that exist in temporary space. This technique guaranteed the Atomicity in the transaction process [31].

2.10.3 The BASE

If the ACID achieves consistency option how can we achieve the availability instead. Suppose that ACID is the pessimist component and forcing each process when completed to implement the consistency, therefore, BASE is the optimistic component and accept the continuous change in the consistency of the database. Then it seems it is impossible to deal with it and it is, in fact, can be controlled completely and will lead to levels of scalability that cannot be obtained with ACID. Availability can be achieved with the BASE through support the partial failure of the system without support of total failure. For example, if we partitioned the users via five database servers, BASE design will encourage the crafting process in such a way that a user database disappointment effect single the 20 percent of the users on that specific host. There is no magic intricate, but this leads to higher apparent availability of the system.

Dan Pritchett, EBay has presented solutions to this problem by proposing that trading some consistency for availability can lead to theatrical enhancements in scalability. The solution that he has is an acronym that is called BASE (Basically Available, Soft state, eventually consistent) and he uses the partitioning as a method for dividing the data. The main point is to allow some of the data to be inconsistent in some time but not all the time, therefore the eventual consistency is a part of the acronym [32]. The purpose of this model is to absorbed the flexibility offered by NoSQL and similar

other ways to manage and create a non-structured data. In general, BASE consists of three main principles:

- **Basic Availability:**

  The concept of the NoSQL databases focuses on the availability of data even in the event of significant failures by employing organization distribution ways to manage databases. Instead of maintaining and storing the data in a single and large approach, each NoSQL publish data across multiple storage systems with a high degree of replication. In a potential third-party case is that the failure disables access to the data that will not significantly lead to database outages entirely.

- **Soft State:**

  The databases of the BASE kind give up from the consistency requirements that belong to ACID model. The important concept that we must know when dealing with BASE is that the consistency is the developer problem and should not be handled by the database.

- **Eventual Consistency:**

  The only condition in the NoSQL databases about the consistency rules is that at some point in the future of data there is a convergence to consistence state. That is a comprehensive exit from the immediate consistency requirement of ACID that prohibits a transaction from executing until the prior transaction has completed and the database has been transformed to a consistent state.

The BASE model is a successful alternative to the ACID model in the work on databases that do not require strict adherence to the relational model [24].

2.11 NoSQL Concepts

2.11.1 MapReduce

MapReduce is model of programming that used for processing and generating huge groups of data. The system was built in 2003 about the programming model in order to simplify the inverse index construction to handle the search in google.com. Google was the first to use this model. In the MapReduce library the user for it express the calculation for it in two functions: Map and Reduce. For the users they have to be specifying Map function where it is written by the user which is processing a key value pair to generate a group of intermediate key value pairs. The user also will write the function of Reduce, receives an intermediate key and a set of values for that key, it is use to merge all intermediate values with the same intermediate key. MapReduce consolidates all of these values with each other to form a group of the youngest group of the original values. For helping in illustrate the MapReduce programming model, and look to the problem of calculating the number of repetition for each word in a big collection of documents. The system runtime take care of partitioning the input data details, execution across a set of machines by scheduling the program, dealing with machine failures and manage communication that is required for inter-machine. Characteristically only zero or one output value produces per Reduce invocation. This helps us to deal with lists of values that are too large to fit existing memory size [33].

2.11.2 Scalability

Means the ability of the system for expanding to deal with the load increases. The significant growth in the size of data and the request for processing more data in a short period are put a huge pressure on the current database. It can be achieved either vertically scaling or horizontally scaling the database level [19]:

- **Vertical Scalability (Scale Up):**

  When the database scaling vertically the tables of the database will be separated through different database models on properly featured machines in order that each server will be assigned to a specific task. The results of this approach are effective in loading balance and the vertical scalability depends on the ability to continuously raising the level of the existing hardware and on the availability of separate components logically from the database [19].

- **Horizontal Scalability (Scale Out):**

  In the scaling horizontally the structure of the database keeps without change through all the models of the database. Reaching horizontal partitioning needs to a consistent maintenance among the various models and their copies, effective balance load and I/O query algorithms until transferring latency is minimized in the data. From theoretical database can achieved speed up by horizontal scaling is matching proportionally with the number of the recently added machines.

The database can be developed through three different ways: It can be scalable through the amount of read processes, the number of write processes and the database size. Currently there are two different technologies to develop this: Replication and Sharding [19].

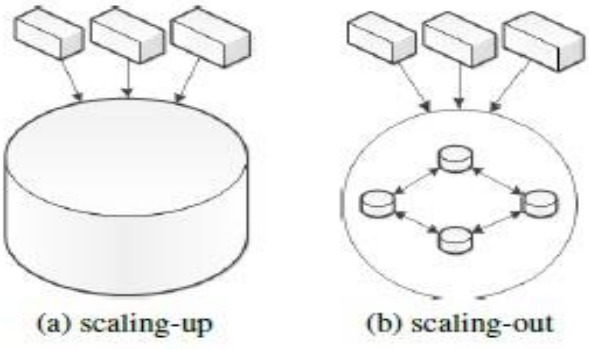


Figure 6 Scaling-up, Scaling-out.

2.11.2.1 Replication

The replication in distributed databases can be defined as a data storage process in more than one node. It allows for load balancing to distribute all read operations across multiple machines, it is considered very useful increasing the performance of reading from databases. It is too very valuable that it makes the cluster robust against failures of lone nodes. If one machine fails there are at least one other of the same data values that can replace the lost node.

It also useful in other times, such as data replication to different base stations serves to strengthen the data against disasters that occur in one area. It is useful also to be close from its users in order to decrease the delay. However, the disadvantages of this method are the writing process. The writing process on the duplicate database must be on each node that is supposed to store the expected element. To do so, the database has two options: The first is that the process of writing must be binding to all duplicated nodes before the database could reply the receipt notice. The second is that the process of writing done first on a node or a specified number node and then later supply a synchronously to other nodes. Database characteristics in terms of availability and consistency determined according to former two options [19].

2.11.2.2 Sharding

The word Sharding arises from the noun 'shard' and means that the data that is inside the database can be splatted too many shards, which can be distributed to several nodes. The data separating can, for instance, be done with a constant hash function which applies to the primary key of the data items in order to determine the related shard.

This implicates that a table is not stored on one solitary machine, but in a cluster of nodes. Its benefit is that nodes can be added to the cluster in order to increase the volume and performance of read and write processes with no the need to adjust the application. Also, it possible to restrict the size of the sharded database cluster when the demand decreases. The downside of sharding is that it makes some classic

database operations very complex and inefficient. The most important processes in relational databases are the join operative that can be used to materialize the relations of data objects. A join process works on two sets of data objects, a left one and a right one, that connected with a pair of characteristics. A distributed join in a sharded database would require that the database would have to search in the right set for all items that are associated to each item in the left set. This would require many requests to all machines that store data items from one of the two sets, which would cause a lot of network traffic. Because of this, most sharded databases do not support join operations.

The more nodes are used inside a sharded database cluster the more is the probability increased that one of the machines or one of the network connections fails. Therefore is sharding often combined with replication, which makes the cluster more robust against hardware failures [19].

2.12 Partitioning

It is a mechanism that assigns the rows of the arbitrary nodes based on the row keys, and so it dispenses load and data via cluster nodes. Partitioning can be theoretically touched by a single master node which could easily load balance all slave nodes, but this would be probably system bottleneck and surely would be a single point of failure. Two basic partitioners are used and they are Byte Ordered Partitioner and Random Partitioner. But any other custom partitioner can be formed by executing suitable java interface [35].

2.12.1 Random Partitioner

It is the most famous partitioner and the default one. Row placement is determined by hash of the row key. All nodes in a cluster are agreed in a ring and each node is given a token. Token value acts as a border of row key hashes which belong to that particular node. This way, if the hash function being used is strong and tokens are generated correctly, data is uniformly (and semi-randomly) spread across the cluster.

Maybe the only disadvantage is that read operation must be very probably executed on several nodes concurrently when querying a range of row keys, and this leads to higher nodes utilization and higher network traffic [35].

2.12.2 Byte Ordered Partitioner

It is not the case for ByteOrderedPartitioner as it partitions rows that are based on their raw principles of row keys. So each node then serves particular range of row keys. Since rows are kept sorted by row keys within each node, querying range of row keys is globally cheaper as it would be very probably served by one node only. Inconvenience is that we have to know (or control somehow) distribution of row keys and assign tokens properly to ensure data being uniformly distributed. Or we would end up with data being distributed very non-uniformly. So this partitioner using single very occasionally in very superior cases, which it is really beneficial [35].

CHAPTER 3

CASE STUDY

3.1 MySQL and Neo4j with Description for Some Features

In the previous chapter we generally mentioned Graph databases together with other types of NoSQL database; however, since one of the main goals of this thesis is giving a simple analysis for two systems, it is necessary to understand what main features and what these systems have. Consequently, in this chapter we will find what are the most Databases that have the best availability and scalability. First of all, we will choose a simplest type of the relational database and describe it which is MySQL. Secondly, will choose one type of the NoSQL database and try to analyze it and we will choose Neo4j which it is a graph database .

3.2 MySQL:

- MySQL server is one of the most widely used open source database management system nowadays; it was developed by MySQL AB in 1995 and now it is owned by Oracle Corporation. In the scientific and academic fields MySQL has great access to the market. Moreover, both of the manufacturers and the community of the user, having great support for MySQL.
- MySQL falls into the category of relational databases. Relational databases are the databases that divide the data into small chunk rather them putting them all together in one data source. The data is stored in tables which are referenced with other tables so that accessibility and flexibility is maximized.
- MySQL was originally designed to handle the requirement of large database faster and more consistent than any other database. Keeping in mind the ease of use, security, connectivity, scalability and availability.
- As for the data structure of MySQL data are stored as rows in a table. It kind of looks like an Excel spreadsheet. Each table stores a specific category of data in

columns and has a predefined list of properties for every element in that table. All data elements in a table must have the same properties if there is no data then put NULL instead.

- ACID (Atomicity, consistency, isolation, and durability) is a group of principles that are controlling the relational model. MySQL is perfect for robust and easy interaction with the database .

- **Buffering and Caching**

- To store, handle, and get data from the database MySQL use storage engines. Different storage engines have various features and performance characteristics, all these storage engines are supported by MySQL. It supports rolling back, transactions with commit, foreign key constraints to maintain data integrity, and crash recovery.
- A buffer pool means linked list of pages, saving frequently the accessed the data on the head of the list by using the diversity of the least recently used algorithm (LRU). To prevent the suffocation that happens in the buffer pool because of the multiple threads access at the same time, the users can allow multiple buffer pools to reach the maximum of 64 instances.
- In addition, to store SELECT statements and their results MySQL used a query cache. The result will be recovered from the cache instead of being executed again, if the same statement is queried again. Between sessions the query cache is being participated in. All queries cache that is using a table will be removed if this table is modified.

- **Scalability and Availability:**

- The scalability refers to the balance between the load of the server and various servers attach to the database by adding hardware and processing power.

- The availability means to the increment of the possibility to access the database and to make sure that any failure of hardware or software doesn't effect on its availability.
- MySQL uses different techniques to provide availability and scalability in the systems. These techniques are:

- **Replication**

- MySQL Replication: it is a technique that provides asynchronous method of replication, which can be stopped and restart at any time. The Data is replicated from the master node; due to the asynchronous nature of replication it will not guarantee that the data will be replicated from the master node to all the slave nodes immediately. Thus, this technique is useful based on the nature of the application.
- It follows the model of Master/Slave; it allows data to be replicated from MySQL database server (the master) to more MySQL database server (the slave). Any change in the Master will recorded into the binary log like events. While each slave will receive a copy of the binary log and keeps read and execute the events. It means that it is a process allows maintaining multiple copies of MySQL data easily by getting the data to automatically copy from a Master to a Slave database.
- Slaves need not be connected permanently to receive updates from the master. The slaves don't need to be connected all the time to receive the updates from the master. That means the updates can happen through the communications of long-distance, even though intermittent or temporary connections like dial-up services.
- According on the configuration, it can choose a database, replicate the database or can even choose the tables to include in a database. Each slave will follow the position, before that the log has been processed, and that's why the slave can catch up with the master whenever it is ready.

- On the other side, the users can configure the master to determine which databases to be written to this log, and to configure for each slave to nominate which event is to be executed from the log. Thus, there is possibility to replicate various databases to various slaves.
- In MySQL replication is asynchronous by default and this means that the master doesn't know when the slaves get the binary log and process it. It can be helpful because of many reasons, including facilitating to have a backup of the data; it is a way to analyze it without use the main database, or just a means to scale out.
- Although, on at least one slave semi synchronous replication can be enabled. In this case, the thread will block and wait until gaining a receipt from the slave which lead to the binary log and the log has been copied to the slave; this will happen to the master after the transaction is being committed.
- Among the master and the slave's MySQL don't provide an official solution for the auto failover, which means if there is a failure the user will be the responsible for checking if the master is outdated, and switch the role to the slave.

- **Sharding**

- MySQL supports the divisions into portions of an individual table, after that distribute the storage into multiple disks and directories. The result of this is that the queries can be done on a smaller set of data.
- This also may help to reduce the I/O difference where multiple partitions can be placed on different physical drives. Sharding is external to the database in MySQL.
- MySQL Cluster is supported Auto-Sharding. MySQL Cluster it is a technique that provides a synchronous method of replication. Unlike replication of master-slave replication method, it creates a cluster. In cluster the Data is replicated among all the nodes in the cluster and accessibility to/from all the nodes is immediate. Main restriction by using this technique is that, all the

nodes to be replicated in the cluster should be within the LAN only means that, this technique does not support nodes that are geographically located.

- Nevertheless, the basic MySQL is not providing an official feature Sharding, where the alternative at the application level is to perform Sharding. This approach is working by having multiple databases of the same structure in multiple servers, and divided the data to these servers that is based on a specific shard key which it is a set of the table columns.
- The application is the responsibility for arranging the access to the data through multiple shards, and redirects the read and the write requests to the correct shard. However, this approach adds lots of complexity to the development of the database and to the work of the administration. First of all, it is difficult task to make sure the load balance among the shards manually. Secondly, to make sure the integrity of the data like the constraints of the foreign key or the transaction is incapable through multiple shards those are the features of MySQL.
- In addition, the horizontal queries (like average or sum) that are needed to be solved against all of those nodes can have greater latency with the increasing of access time of the data besides the number of nodes.
- MySQL doesn't have appropriate asynchronous communication like MapReduce, which can parallelize the operation and collect the results. As a result, the implementation can be very complicated and not safe with lots of branching and connections with the following operations. Also Sharding can be done at the proxy layer of MySQL.
- MySQL Proxy is an application that is being placed among MySQL servers and clients, it has the ability to intercept and direct the queries into a specific server. Nevertheless, MySQL Proxy is not used within production environment and currently it is at Alpha version.

3.3 Graph Database:

- All of NoSQL Databases share the same constraints, where they derive from their distributed architecture.
- They are all of them victims to the CAP Theorem, thus they can only provide either eventual consistency or sacrifice some amount of availability.
- There is one weak point in all of the databases that they don't offer functionality to the relations model on the side of the database, that is in almost all the cases and this has to be done in the application layer.
- There is a line between the several of the NoSQL databases, but it is very thin, although they still have some small but very important differences.
- One of the NoSQL databases that we choose is Graph database. The graph database uses data model, where the data is stored as nodes and relationships; the nodes are represented entities in the database, and the data on the connection between two entities are represented by the relationship. The model of graph database is gaining their popularity due to its flexibility and rapid development time.
- Helping to design new features, it is easier to quickly by adding any new functionality without affecting on the previous deployments. There are various types of graph databases available and Neo4j is one of them.

3.4 Neo4j

- It is a graph database due to the model of the data that is used to express it, it is used to store nodes that are connected with relationships, for both constructs the properties are support defined by the user.
- It is open source was written in Java, developed by the company Neo Technology in 2007, and Embeddable which means that it can be added to an application and just used it like any other library.

- Even though the term of the graph has been known since the end of the 19th century, it has been one of the most generic of the data structures, and only in the last decades there were being strong attention in the graph idea.

- **Why Neo4j?**

- Neo4j is a conventional, distributed database because it can be embedded into various programming languages, such as Java, Python, Ruby, etc.
- Neo4j has a traversal, a traversal means moving from a node to its edge. Another reason for picking Neo4j is because Neo4j is a well-documented database.
- Although the Neo4j team is working to make it writable, this problem is yet to be resolved. On the other hand, manipulation of graph database is done from an application layer.
- The Neo4j model is created more to read than to compute, so the amount of reading is more than the amount of writing for a graph database.

- **Data Model**

For each instance in Neo4j, there is exactly one graph. Neo4j is following the basic graph database model, by having two primitive types: nodes and relationships.

i. Nodes

Graph vertices are called nodes, and edges are called relationships. Each node contain of a set of properties. Every node can consist of one or more identifying attributes, this mean that every node has a unique ID which is a nonnegative integer and cannot be change and that serves as a primary identifier.

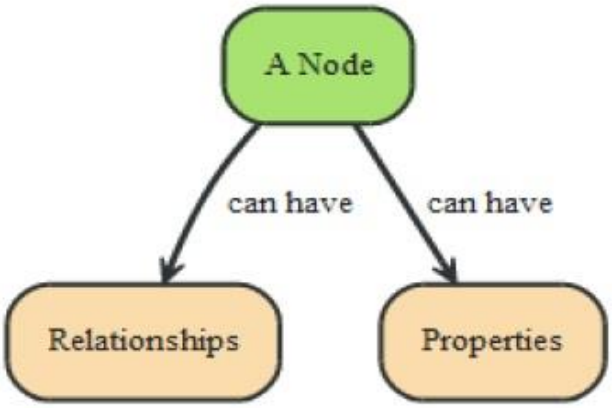


Figure 7 Node in Neo4j [59]

ii. Relationship

- It exists between two nodes only and can have their features only; the only feature that is required is a type feature. They are always directed and that is mean they have a start and end node where the start and the end node are not allowed to synchronize, which mean self-loops are not allowed.

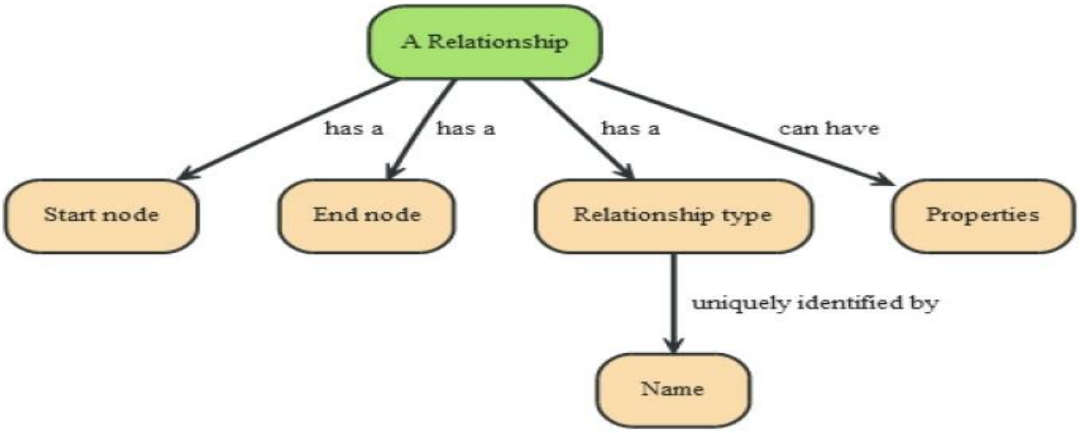


Figure 8 Relationship in Neo4j [59]

- Like nodes, relationships have a set of properties, they also always have types, and for the relationship type what is needed is a defined name, which is consisting of a Java string.

A relationship connects two nodes, and is guaranteed to have valid start and end nodes.

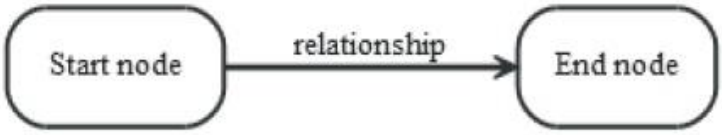


Figure 9 Node and Relationship in Neo4j [59]

As relationships are always directed, they can be viewed as outgoing or incoming relative to a node, which is useful when traversing the graph:

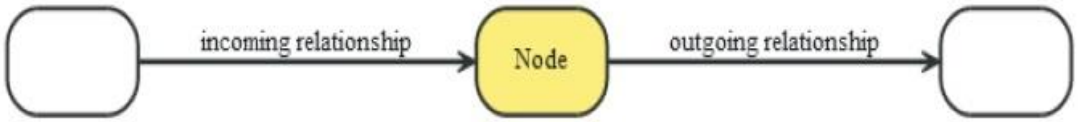


Figure 10 Incoming and Outgoing Relationship in Neo4j [59]

iii. Properties

The property is a key-value pair where the key is Java string, and the value is Java primitive (boolean, char, byte, short, int, long, float or double, String) or an array of any of these.

3.5 Neo4j Competitive Features

3.5.1 ACID Compliance

- Neo4j is database owns all the advantages of its kind, like flexible schema, horizontal scalability, high-performance and high availability. Nevertheless, there are some specific features make Neo4j very popular among the users, the developers and DBAs; including them is the appropriate behavior for ACID.
- The ACID transaction concept is having full support by Neo4j. This is can be achieved by having logging in memory transaction and a lock manager to apply locks on any database objects that is being changed during the transaction. When the transaction is completed on successful the changes in the log will removed from the disk, while the transaction is rollback which means only discarding from the log.
- In Neo4j it is the basis of data reliability. It imposes on all operations that are modifying the data that is happening into a transaction to guarantee data consistency. For this, the robustness does not apply only for single embedded graph instance, but it extends for high availability installations of multi sharding. Cache-based Sharding is used for large deployments, to get high availability for maintaining high performance with the dataset, that surpass the main memory space. It isn't sharding in the traditional meaning since it is expected that a full data set to be present on each database instance. Between other important features in neo4j is a Cypher query language which is simple but powerful and has highly efficient traversal mechanisms.
- In Neo4j database, it can be distributed on cloud serves by using the Master/Slave replication model. For any write request, it should be performed on a slave node, this slave will synchronize with the master and any updates will be in the eventually will be pushed to all the other slaves. As a result, the property of the database consistency is to lose for the eventual consistency while the rest of ACID characteristics stay the same. Nevertheless, the solution

for the database distribution is only being applied for the replication of the data which help the system to deal with higher read load.

3.5.2 Powerful Traversal

- It means visiting the nodes after the relationships depending for some rules. In the graph database they are all about connecting graph data. One of the main features in Neo4j is constant time traversals for relationships in the graph because of the double-linking on the storage level between nodes and relationships for both in depth and in breadth.
- Only in the most cases the sub graph is visited, where in the graph the interesting nodes and relationships are found. Neo4j is comes with a callback which is based on traversal Application Programming Interface that is allowed to specify the traversal rules. Combining between compact storage and memory caching for graphs will lead to efficient scaling up to billions of nodes on moderate hardware in one database.

3.5.3 Query Language

- Graph query languages are based on the fundamental concept of graph traversal. The strength of graph query languages is measured on their ability to efficiently traverse the graphs.
- Cypher is one of two primary languages that are declared as a graph query language used to traverse Neo4j graphs that allows for the data store to querying and updating in efficient and expressive from the graph database like SQL. By Cypher it can be easy to express very complicated database queries. However, unlike SQL, it doesn't require exactly to describe how doing it, because after all Cypher is comparatively simple but still very powerful language .
- It allows concentrating on the domain instead of worrying about the structure of the database. The structures that are designed to be human-readable are

based on the English language and iconography which help to make queries explain it more.

- Cypher is built upon established practices for expressive querying and inspired by a number of various approaches. It has inspired from SQL language that seeks to avoid the needing to write traversals in code and to describe patterns in graphs. Keywords like WHERE, RETURN and there are more other keywords that are inspired from SQL and SPARQL. Cypher didn't focus on how to the clarity of expressing retrieves from the graph; it is what to retrieve from the graph .
- In Cypher now nodes of the graph can be labelled and its properties can be used as indexes. This feature improves the performance of Cypher queries when graph size and nodes increases. It makes queries faster and easy.

3.6 Comparison between MySQL and Neo4j

As we can see from the features for both of MySQL and Neo4j there were some similarities and Differentiations between them since all of the NoSQL Database are depending on the relational Database when it comes to building a new database.

- The language that is used in both of MySQL and in Neo4j. The language in MySQL is full of statements and simple and readable for the users and the developers, its use CRUD operations and other functionalities, while the language that is have popular used in Neo4j is Cypher and this language was inspired from SQL and that means it have Similar of keywords like WHERE, ORDER BY and another group of statements.

- MySQL and Neo4j the two of them are considered as open source database.

- Between MySQL and Neo4j the flexibility for each one is higher than other types from their kind, but among them Neo4j is more flexible and rapid development database.

- MySQL and Neo4j having replication, which it is a technique of Master/Slave and this one of the technologies that is used in horizontal scaling, however, MySQL can have vertical scaling also.

- MySQL is followed schema before inserting data, while Neo4j is having flexible schema or schema free.

- Sharding in MySQL follows horizontal partitioning and sharding with MySQL Cluster, while in Neo4j follow cache-based sharding.

- MySQL and Neo4j both follow ACID properties, Neo4j follow the CAP theorem also to achieve high availability, while MySQL tries to achieve consistency.

- For data model MySQL was designed for the principle concept of the normalization, which allows Relational model to store any data without redundancy and loss of information, while for Neo4j there is specifically one graph and for each graph there are nodes, relationships and properties for both of them.

CHAPTER 4

CONCLUSION

The relational database can be more efficient because of the concept that is called normalization. The use of normalization in relational database reduces redundancy of data; therefore, this will lead to performance improvement in query time of the relational database. Until now, the popularity of the relational database has been wider if it is compared to the NoSQL database because the relational database is a mature and stable database. In addition to, the NoSQL database is evolving, furthermore, it can be considered as not only a stable database. However, the popularity is rapidly growing for the NoSQL. The development of NoSQL database is the result of the increasing in the data size for processing regularly. With NoSQL databases, there is a solution for the problems that is needed scalability. They are optimized for an application area and do their work in this field as well. In the next years, the NoSQL database is supposed to be more sophisticated and mature.

In conclusion, it is hard to point out a clear winner for the best database between MySQL and Neo4j because of the data types are different. In addition, each database has its own positives and negatives, and its own area of application. Therefore, we have concluded that both of MySQL and Neo4j are open source. Moreover, the flexibility for both of them is higher than other types from their kind, however, between them Neo4j is more flexible and rapid in the development of the database. Besides, the language in MySQL is simple and readable for both users and developers, while the language in Neo4j is inspired from MySQL. All in all, we prefer Neo4j which is graph database for these reasons, according for what mentioned above, because Neo4j can handle with large sets of data and embeddable which means that it can be added to an application and just used it like any other library.

4.1 Findings

In this study, we have found scalability and high availability with other features, are desirable in the distributed systems. Also, we found that both of MySQL and Neo4j are using to provide high availability and scalability with two techniques, they are the replication and the sharding. In MySQL, high availability following MySQL replication, which depends on Master/Slave technique and for the scalability it follows MySQL cluster, which is a technique support auto-sharding and horizontal partitioning. While for Neo4j, it is providing high availability for sharding with a technique that is called cache-based sharding and for scalability it is following Master/Slave also.

4.2 Limitation

To limit the domain and to increase our knowledge in the databases, we have some limitations in this study, which we did not provide any implementing work, so it is a theoretical study by investigating in MySQL and Neo4j. The other reason for limiting the study is for the lack of time.

4.3 Future Work

In this study, we have investigated in MySQL and Neo4j database so the work that is presented here is theoretical work. As for the future, we can make it an implementation work by involving running experiments on both MySQL and Neo4j with sets of data. We plan to implement experiments for testing replication and Sharding in MySQL and in Neo4j, to find out which type is having high availability and scalability than the other.

REFERENCES

1. **Chamberlin D. D., Boyce R. F., (1974),** *"SEQUEL: A Structured English Query Language"*, Proceedings of the ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control, pp. 249-264.

2. **Berg K., Seymour T., and Coel R., (2013)**, *"History Of Databases"*, International Journal of Management and Information Services.

3. **Bruhn D., (2011),** *"Comparison Of Distribution Technologies In Different NoSQL Database Systems".*

4. **Chang F., Dean J., Ghemawat S., Hsieh W. C., Wallach D. A., Burrows M. C. T., M., Fikes., Gruber R. E., (2006),** *"Bigtable: A Distributed Storage System for Structured Data"*, ACM Trans. Comput. Syst.

5. **Tauro C. J. M.,, Aravindh S., Shreeharsha A. B., (2012),** "*Comparative Study of the New Generation, Agile, Scalable, High Performance NOSQL Database"*, International Journal of Computer Applications, vol. 48, no.20.

6. **Jatana N., Puri S., Ahuja M., Kathuria I., Gosain D., (2012),** "*A Survey and Comparison of Relational and Non-Relational Database*", International Journal of Engineering Re-search & Technology (IJERT), vol. 1, no. 6, pp.1-5.

7. **Batra S., Tyagi C., (2012),** *"Comparative Analysis of Relational and Graph Databases",* International Journal of Soft Computing and Engineering (IJSCE), vol. 2, no. 2.

8. **Gray J., Reuter A., (1993),** *"Transaction Processing: Concepts and Techniques",* SIGMOD Record, vol. 37, no. 2, pp. 38-40.

9. **Bartholomew D., (2010),** *"SQL vs. NoSQL",* Linux Journal.

10. **Codd E. F., (1970),** *"A Relational Model of Data for Large Shared Data Banks",* Communications of the ACM, vol. 13, no. 6, pp.377-387.

11. **Hussain T., Shamail M.M., Awais S., (2003),** *"Eliminating Process of Normalization in Relational Database Design",* Multi Topic Conference, INMIC 7th International, IEEE, pp. 408-413.

12. **Stonebraker M., (2010),** *"SQL Databases vs. NoSQL Databases",* Communications of the ACM, vol. 53, no. 4, pp. 10-11.

13. **Sharma V., Dave M., (2012),** *"SQL and NoSQL Databases",* IJARCSSE, vol. 2, no. 8.

14. **Pokorny J., (2011),** *"NoSQL Databases: A Step to Database Scalability in Web Environment",* in proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services, iiWAS'11, pp. 278–283.

15. **Datastax, (2012),** *"WHY NOSQL?"*, WHITE PAPER.

16. **Guinard D., Trifa V., Pham T., Liechti O., (2009),** *"Towards Physical Mashups in the Web of Things",* Networked Sensing Systems (INSS), Sixth International Conference.

17. **Weber S., (2012),** "*NOSQL DATABASES".*

18. **Leavitt N., (2010),** "*Will NoSQL Databases Live Up to Their Promise?*", vol. 43, pp. 12-14.

19. **Cattell R., (2011),** *"Scalable SQL and NoSQL Data Stores"*, SIGMOD Rec., vol. 39, no.4, pp. 12–27.

20. **Mapanga I**., **Kadebu P., (2013),** "*Database Management Systems: A NoSQL Analysis*", International Journal of Modern Communication Technologies & Research (IJMCTR), ISSN: 2321-0850, vol. 1, no. 7, pp. 12-18.

21. **Jatana N., Puri S., Ahuja M., Kathuria I., Gosain D., (2012),** "*A Survey and Comparison of Relational and Non-Relational Database*", International Journal of Engineering Re-search & Technology (IJERT), vol. 1, no. 6, pp.1-5.

22. **Padhy P. R., Patra R. M., Satapathy C. S., (2011),** *"RDBMs to NoSQL: Reviewing Some Next-Generation Non-Relational Databases",* International Journal Of Advanced Engineering Sciences and Technologies (IJAEST), ISSN: 2230-7818, vol. 11, no. 1, pp. 15-30.

23. **Tudorica B. G., Bucur C., (2011),** "*A Comparison between Several NoSQL Databases with Comments and Notes*", IEEE Conference on Commerce and Enterprise Computing, pp. 1-5.

24. **Moniruzzaman A. B. M., Hossain S. A., (2013),** *"NoSQL Database: New Era of Databases for Big Data Analytics-Classification, Characteristics and Comparison"*, International Journal of Database Theory and Application, vol. 6, no. 4.

25. **Abramova V., Bernardino J., (2013),** "*NoSQL Databases: Mongodb vs. Cassandra*", in Proceedings of the International C* Conference on Computer Science and Software Engineering, C3S2E '13, pp. 14–22.

26. **Angles R., Gutierrez C., (2008),** *"Survey of Graph Database Models"*, ACM Computing Surveys, vol. 40.

27. **Angles R., (2012),** *"A Comparison of Current Graph Database Models*", in Proceedings of the 2012 IEEE 28th International Conference on Data Engineering Workshops (ICDEW), IEEE 28th International Conference, pp. 171-177.

28. **Brewer E., (2012),** *"CAP Twelve Years Later: How The "Rules" Have Changed"*, IEEE Computer, vol. 45, pp. 23–29.

29. **Gilbert S., Lynch N., (2002),** *"Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services"*, ACM SIGACT News, vol. 33, no. 2, pp. 51-59.

30. **Gilbert S., Lynch N. A., (2012),** *"Perspectives on the CAP Theorem"*, MIT faculty, vol. 45, no. 2, pp. 30-36.

31. **Mohan C., Haderle D., Lindsay B., Pirahesh H., Schwarz P., (1992),** *"Aries: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging"*, ACM Trans. Database Syst, vol. 17, no. 1, pp. 94-162.

32. **Pritchett D., (2008),** *"BASE: An Acid Alternative"*, Magazine Queue - Object-Relational Mapping, vol. 6, no. 3, pp. 48-55.

33. **Dean J., Ghemawat S., (2004),** *"Mapreduce: Simplified Data Processing on Large Clusters"*, In Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation, Appear in OSDI 2004, vol. 6, pp. 10–10.

34. **Nasholm P., (2012),** *"Extracting Data from NoSQL Databases: A Step Towards Interactive Visual Analysis of NoSQL Data"*, Master's Thesis, University of Gothenburg, Sweden.

35. **Bjorklund A., (2011),** *"NoSQL Databases for Software Project Data"*.

36. **Mughees M., (2013),** *"Data Migration from Standard SQL to NOSQL"*.

37. **Cheng J., Yu J. X. (2011),** *"A Survey of Relational Approaches for Graph Pattern Matching Over Large Graphs"*, In Graph Data Management, Eds. IGI Global, vol. 5, no.11.

38. **Datastax, (2013),** *"NoSQL In Enterprise"*, White Paper.

39. **Digital Ocean, (2014)**, *"A Comparison of NoSQL Database Management Systems and Models"*.

40. **Ferreira L., (2011),** "*Bridging the Gap between SQL and NoSQL*", MI-STAR, pp. 187-197.

41. **Hadjigeorgiou C., (2013),** *"RDBMS vs. NoSQL: Performance and Scaling Comparison"*, The University of Edinburgh.

42. **Han J., Haihong E., Le G., Du J., (2011),** *"Survey on NoSQL Database"*, in Pervasive Computing and Applications (ICPCA), 6th International Conference, pp. 363-366.

43. **Hecht R., Jablonski S., (2011)**, *"NoSQL Evaluation: A Use Case Oriented Survey"*, Proc. Int. Conf. Cloud Serv., pp. 336-341.

44. **Holzschuher F., Peinl R., (2013),** *"Performance of Graph Query Languages: Comparison of Cypher, Gremlin and Native Access In Neo4j"*, in Proceedings of the Joint EDBT/ICDT Workshops, ser. EDBT '13, ACM.

45. **Lith A., Mattson J. (2010),** *"Investigating Storage Solutions for Large Data: A Comparison of Well Performing and Scalable Data Storage Solutions for Real Time Extraction and Batch Insertion of Data*", Department of Computer Science and Engineering, Chalmers University of Technology.

46. **Madhulatha T., (2012),** "*Graph Partitioning Advance Clustering Technique*", International Journal of Computer Sciences and Engineering Systems (IJCSES), Computer Sci. Eng. Surv., vol. 3, no. 1, pp. 91-104.

47. **Miller J. J., (2013),** *"Graph Database Applications and Concepts with Neo4j"*, Proceedings of the Southern Association for Information Systems Conference, pp. 141-147.

48. **Montag D., (2013),** *"Understanding Neo4j Scalability"*. Neo Technology.

49. **Mykletun E., Tsudik G., (2006),** *"Aggregation Queries in the Database-As-A-Service Model"*, 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security XX Lecture Notes in Computer Science, vol. 4127, pp. 89-103.

50. **Parker Z., Poe S., Vrbsky V. S., (2013),** *"Comparing NoSQL Mongodb to a SQL DB"*, ACMSE '13, in Proceedings of the 51st ACM Southeast Conference , no. 5, pp. 51–56.

51. **Payberah A. H., (2012),** *"NoSQL DATABASES"*.

52. **Rodriguez M. A., Neubauer P., (2010),** *"The Graph Traversal Pattern"*, Cornell University Library.

53. **Vicknair C., Macias M., Zhao Z., Nan X., Chen Y., Wilkins D., (2010),** "*A Comparison of a Graph Database and a Relational Database: A Data*

Provenance Perspective", In Proceedings of the 48th Annual Southeast Regional Conference, ACM SE '10, pp. 421-426.

54. **Vo H. T., Wang S., Agrawal D., Chen G., Ooi B. C., (2012),** "*Logbase: A Scalable Log-Structured Database System"*, Technical report, School of Computing, pp.1004-1015.

55. **Vogels W., (2009),** "*Eventually Consistent",* Communications of the ACM - Rural Engineering Development, vol. 52, no. 1, pp. 40-44.

56. **Wood P.T., (2012),** "*Query Languages for Graph Databases*", ACM SIGMOD Record, vol. 41, no. 1, pp. 50-60.

57. www.blog.flux7.com/blogs/nosql/cap-theorem-why-does-it-matter, (Data Download Date: 05.02.2015).

58. www.code.osehra.org/OSSPTutorial/Neo4J/DataModel.html, (Data Download Date: 11.03.2015).

59. www.tinman.cs.gsu.edu/~raj/8711/sp13/neo4j/NEO4J_Zhang_Li_Liu.pdf, (Data Downloda Date: 24.03.2015).

APPENDICES A

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: AL-ANI, Zahraa

Date and Place of Birth: 1st January 1983, Baghdad

Marital Status: Married

Phone: +90 534 9168 785

Email: pinkrosie467@yahoo.com

EDUCATION

Degree	Institution	Year of Graduation
M.Sc.	Çankaya University, Information Technology	2015
B.Sc.	Al-Ma'amon University	2005
High School	Al-resalah high school for girls	2000

FOREIN LANGUAGES

Advanced English, Beginner French

HOBBIES

Travel, Books, Swimming, Computer Games.