Comparison of Graph Databases and Relational Databases When Handling Large-Scale Social Data

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Chen, Yaowen

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Abstract

Over the past few years, with the rapid development of mobile technology, more people use mobile social applications, such as Facebook, Twitter and Weibo, in their daily lives, and there is an increasing amount of social data. Thus, finding a suitable storage approach to store and process the social data, especially for the large-scale social data, should be important for the social network companies. Traditionally, a relational database, which represents data in terms of tables, is widely used in the legacy applications. However, a graph database, which is a kind of NoSQL databases, is in a rapid development to handle the growing amount of unstructured or semi-structured data. The two kinds of storage approaches have their own advantages. For example, a relational database should be a more mature storage approach, and a graph database can handle graph-like data in an easier way.

In this research, a comparison of capabilities for storing and processing large-scale social data between relational databases and graph databases is applied. Two kinds of analysis, the quantitative research analysis of storage cost and executing time and the qualitative analysis of five criteria, including maturity, ease of programming, flexibility, security and data visualization, are taken into the comparison to evaluate the performance of relational databases and graph databases when handling large-scale social data. Also, a simple mobile social application is developed for experiments. The comparison is used to figure out which kind of database is more suitable for handling large-scale social data, and it can compare more graph database models with real-world social data sets in the future research.

Acknowledgements

There are so many people I want to thank for helping me through this journey. First of all, I must acknowledge the irreplaceable contribution of my supervisor, Dr. Ralph Deters. With his continuous support and advice throughout my graduate studies at the University of Saskatchewan. His guidance helped me in all the time of research and writing of this thesis. I learned a lot from him, and it is my fortunate to have him as my supervisor.

Also, I would like to express my heartfelt thanks to the other members of my committee, Dr. Julita Vassileva, Dr. Gordon McCalla, and my external examiner Dr. Anh Dinh for their encouraging words, thoughtful criticism, valuable comments and constructive suggestions to help me to complete my studies and write this thesis.

Moreover, I am very thankful to Gwen Lancaster and other staffs at the Department of Computer Science and my colleagues from Multi-Agent, Distributed, Mobile and Ubiquitous Computing (MADMUC) Lab for their help throughout my graduate studies.

Finally, I need to acknowledge the love and support from my parents, Shengzhan and Ruyu. Thank them for their encouragement to keep me focusing on my studies, for their understanding, and for their endless love throughout my graduate studies.

Contents

List of Tables

List of Figures

List of Abbreviations

LOF	List of Figures
LOT	List of Tables
ACID	Atomicity, Consistency, Isolation and Durability
GDB	Graph Database
RDB	Relational Database
PK	Primary Key
FK	Foreign Key
NF	Normalization Form
RDF	Resource Description Framework
NoSQL	Not Only SQL
OS	Operation System
RDBMs	Relational Database Management System
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
WWW	World Wide Web
W3C	World Wide Web Consortium
API	Application Programming Interface
SQL	Structured Query Language

Chapter 1

Introduction

In recent years, there has been increasing importance in storing and processing data in the form of graphs, which is one of the basic data structures in computer science. According to Mashaghi[52], it can use graphs for modeling purposes in many types of relations and processes in physical, biological, social and information systems, and also for representing many practical problems. Meanwhile, several companies, especially the social media companies, which are interested in representing social networks in graphs, are trying to apply graph models for practical applications.

A graph contains lots of nodes and edges[77], and in a social graph, the nodes refer to the social actors in the social network, and the edges represent the social relationships between the social actors. An edge can describe what it is, where it comes from, and where it goes to. Lots of social information is implicit in social graphs. Also, graph structure can enable users to process several graph operations, such as graph transposing, graph complement, graph product and graph minor, on the social data[77]. These operations offer more possibilities to process the social data to explore more social information. Therefore, there is an increasing need to store and query the graphs.

Figure 1.1: Social Graph[49]

In these contexts, it may be unsuitable to use the relational database management systems, which is a traditional storage approach storing data in terms of tables, to handle the graph data, since they hardly represent the inherent graph structure, and even the best relational database systems so for may not match the requests for serving social graph data, although they are ACID (Atomicity, Consistency, Isolation, and Durability) compliant and have high performance when handling large-scale data. Meanwhile, a new storage approach, called GDBs (Graph Database Systems), is emerging to provide a solution for storing and working with graphs.

Graph databases, the new storage model, have been adopted in the past few years. They use graph structure to represent and store data, and enable consequently semantic queries with nodes, edges and properties[7]. Thus, a graph database can offer a cost-saving solution for storing graph-like data, compared with the relational model. For example, processing some graph operations to query data from relational databases can be very inefficient, because it may need complex join operations or subqueries to assist. On the other hand, it can be easily handled in graph databases.

This research provides a benchmark study to compare the performance and capabilities of the relational databases and the graph databases on storing and processing large-scale social graph data. The main highlights of this thesis will include the following three points:

- 1. By comparing the performances of relational database systems and graph database systems on storing and processing large-scale social graph data, this thesis highlights the capabilities of the relational databases and the graph databases.
- 2. The designed queries can be used to implement a database benchmark for analyzing the capabilities of relational databases and graph databases for storing and processing large-scale graph data.
- 3. This research implements a simple mobile social application, which applies relational databases and graph databases as the backend data storage approaches. It can be used to simulate the practical social applications for evaluating the performances of relational databases and graph databases in real-world social applications.

The rest of this thesis is organized as follows: Chapter 2 discusses the problem definition. Chapter 3 provides the literature review associated with relational databases, graph databases, database benchmarks and other work related to the thesis. Chapter 4 describes the database setup and the kernels of the benchmark. Chapter 5 focuses on social graph generation and implementation of the social mobile application and the middleware. Chapter 6 provides the experiments, experimental results, and evaluation and discussion. Chapter 7 concludes this thesis with directions for future work.

Chapter 2

Problem Definition

This research is focusing on comparing the performances and capabilities of graph databases and relational databases on storing and processing large-scale social graph data. Besides comparing the storage cost and query performance, the proposed simple mobile social application can be used to evaluate relational databases and graph databases in real-world applications with qualitative analysis based on five subjective judgments, including maturity and level of support, ease of programming, flexibility, security and data visualization.

In order to achieve the goal of this research, the following key questions should be answered in this thesis:

- 1. What are the advantages of applying the graph databases and the relational databases to store and process large-scale social graphs data?
    1. From the hardware perspective, which kind of the storage approaches can reduce the storage cost for storing large-scale social graph data?
    2. Which kind of the storage approaches has better query performance, specifically, shorter execution time?
- 2. How is the performance of the relational databases and the graph databases on the reliability?
    1. Which storage approach has the higher maturity and more support for storing and processing large-scale social data?
    2. How is the performance of both relational databases and graph databases on enforcing the data security?
- 3. How practicability are the relational database systems and the graph database systems?
    1. Which the storage approach, the relational database or the graph database, is easier for developers to apply in the practical applications?
    2. Which kind of database systems is more flexible for handling the unstructured or semi-structured data, especially for the graph-like data?
    3. How can relational databases and graph databases support the data visualization?

In order to answer these questions, three challenges should be addressed:

- Challenge 1: Graphs Generator: Normally, the nodes and edges in random graphs can be similar. However, in this research, the stored and processed graphs should be large-scale social graphs, and such graphs are highly right-skewed, meaning the large majority of nodes have low degrees and only a small number of nodes have high degrees[78]. Thus, the nodes should be different in a graph, and it makes the social graphs different to the random graphs. Thus, generating large social graphs, which meet this property, is the first challenge in this research.

- Challenge 2: Cross-Platform Mobile Social Application: With the development of mobile technology, mobile applications become more common in people's daily life, especially for the social applications. The mobile social applications are highly interested in storing and processing the social data in terms of graphs. Besides the quantitative analysis, which is based on the performance on storage cost and query execution time, the qualitative analysis should be necessary to complete the understanding of the storage approaches. Moreover, building a simple mobile application to simulate some practical functions should be significant for both quantitative analysis and qualitative analysis, and it also is a challenge in this research.

- Challenge 3: Evaluation Criteria: The core of this research is comparing the performances of the relational database (RDB) and the graph database (GDB) on storing and processing large-scale social graph data. Therefore, it is essential to benchmark the performance of the database systems. Although there are several benchmarks of relational databases or graph databases, it lacks unified criteria for evaluating the performances of relational databases and graph databases in one case. Therefore, the third challenge here is to propose sufficient criteria to compare the different types of database systems.

Besides the main goal of thesis, the follows list two sub-goals that should be achieved in this study as well:

- Goal 1: To develop a simple hybird mobile application, which can run on different mobile devices, to be familiar with the cross-platform mobile application development.

    1. Cross-platform mobile application running on various mobile platforms
    2. Hybrid application applying the Web-technology with the native shell
    3. Applying jQuery Mobile and PhoneGap

- Goal 2: To construct a database benchmark for evaluating database performance on storing and processing large-scale graph data

    1. Large-scale social graph data
    2. Suitable for graph databases and relational databases
    3. Robust, reliable, repeatable benchmark

Chapter 3

Literature Review

The literature review is organized as follows:

The first section, Section 3.1, introduces the relational database, which is a key point in this research. After reviewing the related literature, it provides the basic knowledge about the relational databases. In this section, it reviews several important technologies or terminologies about the relational databases including ACID (Atomicity, Consistency, Isolation, and Durability), which is a set of properties that guarantee the reliability of database transactions; Primary Key and Foreign Key, which are the key concepts to build relationships in a relational database; Database Normalization, which is a process to decompose the data into smaller relations to minimize the data redundancy.

In addition, Section 3.2 talks about the graph databases, which is another focus of this research besides the relational databases. Firstly, the concept of the NoSQL databases is introduced in subsection 3.2.1. In addition, the three types of graph model in the graph databases: property graph, hypergraph, and RDF Triple, are reviewed in subsection 3.2.2, 3.2.3 and 3.2.4.

Moreover, the database benchmark is important for evaluating the performance of databases, and Section 3.3 represents the works about benchmarking the databases.

Furthermore, a key concept in this research is the social network, so Section 3.4 is talking about the knowledge about the social network. Specifically, subsection 3.4.1 describes the social network analysis to express the importance of social relationships in the research.

Finally, in order to show the impacts of mobile technology on computing, the information about mobile computing is reviewed in Section 3.5. Additionally, this research needs to build up a cross-platform mobile application that can run on different mobile OSs. Therefore, subsection 3.5.1 introduces the cross-platform mobile application development technologies.

3.1 Relational Database

How to store and access data safely and securely has been a challenging topic for a long time. In 1970, Edgar Codd [19] proposed the idea about the relational model, and since then the relational model has almost maintained the entire database market, and dominated the database development until the emergence of NoSQL technologies. Currently, there are many different commercial vendors of the relational database

management systems (RDBMs), and their products vary significantly in capabilities and cost. Some leading vendors are listed as follows:

- Computer Associates: INGRES
- IBM: DB2
- INFORMIX Software: INFORMIX
- Oracle Corporation: Oracle
- Microsoft Corporation: MS Access
- Microsoft Corporation: SQL Server
- MySQL AB: MySQL
- PostgreSQL Dvlp Grp: PostgreSQL
- Sybase :Sybase 11

The relational model, as proposed by Codd[21], organizes data into one or more tables of rows and columns, and each row should be identified uniquely. The relational model stores information using tables (relations) to enable software storing, accessing and modifying data that stored in the server side. Table 3.1 represent an example table in a relational database.

Table 3.1: Relational Database Example

FirstName	LastName	Gender	Age	Hometown
Tom	Smith	Male	25	2
Roy	Brown	Male	51	4
Odie	Howard	Male	42	3
Yaowen	Chen	Male	28	5
Nan	Chen	Male	29	6
Ruyu	Zhang	Female	51	1
Shengzhan	Chen	Male	54	1

There are five relational attributes, including “FirstName”, “LastName”, “Gender”, “Age” and “Hometown”. Each attribute is assigned with value, and a row of data, such as “Yaowen”, “Chen”, “Male”, “28”

and “5”, represents a tuple. The number of tuples is called cardinality, and the number of attributes is called degree, so the cardinality of Table 3.1 is 7, and the degree is 5. In addition, Table 3.2 is showing more relational model terminologies with their explanations[21].

Table 3.2: Relational Model Terminology

Relational Model Terminology	Explanation
Relation	Table in a database
Domain	Type of column in a table
Attribute	Column of a table
Attribute value	Column value
Tuple	Row of a table
Entity	Name of a table
Degree	Number of columns in a table
Cardinality	Number of rows in a table

Moreover, Codd represented the properties of the relations in [21][19] as follows:

- A row in a table represent a tuple in a relation.
- Each row should be distinct to avoid duplicate row in a table.
- The order of rows should be meaningless.
- The order of columns should be meaningless.
- All table values should be atomic.

3.1.1 ACID

In database systems, a transaction refers to a single logical operation on the data, for example, inserting data into a database. In order to attain the reliability of a transaction, a set of properties, including Atomicity, Consistency, Isolation and Durability (ACID), were proposed in 1983[36].

- Atomicity

  Each transaction should be atomic (all or nothing)[27], which means a transaction should be viewed as a whole unit. If any part of the transaction fails, the whole transaction should fail without making any changes on the databases.

- Consistency

  Consistency property ensures a transaction should bring the database from one valid state to another after the transaction processes completely and successfully[61]. No matter whether the transactions is correct or not, it needs to keep the database consistent.

- Isolation

  Isolation property determines each running transaction should be independent on other concurrent transactions until one has been completed and committed successfully[36]. Thus, the effects of incomplete transactions should be invisible to other transactions.

- Durability

  Durability means once a transaction commits, the changes in the database should remain the same, even in the case of system crashes, power loss or error[27]. This property ensures the execution results can be recorded permanently.

Although the relational databases guarantee the reliability of a transaction through ACID, with the development of web technologies, the data storage techniques have processed revolutionary changes. Scalability and availability, especially in distributed environments, have played a more important role than before. Because the web-based data is very huge data and distributed naturally in the most cases, it may be challenging for RDMSs to store and process, and this leads to the inception of the NoSQL databases.

3.1.2 Primary Key and Foreign Key

A Primary Key (PK) in a table is a single or a set of columns with unique values that can be used to identify each record in the table uniquely[60]. The value, or the combination of values, of the PK attributes for a tuple should be unique, and not be repeated by other tuples in the table to identify the tuple. Usually, the database management system can assign a constraint to ensure the uniqueness of the Primary Key. The Primary Key can reduce the redundant data in the database.

The Primary Keys need to work with the Foreign Keys in a relational database. A Foreign Key (FK) in a table is a field representing a reference to the Primary Key of another table[60]. In addition, there could be multiple FKs in a table to point to multiple PKs in multiple tables.

Figure 3.1 shows a complex data schema of a relational database. There are six tables, namely “Students”, “students_classes”, “teachers”, “Classes”, “sections” and “departments”. Each table has its own Primary Key, the “id” attribute in each table. The attributes “student_id”, “class_id” and “section_id” in table “students_classes” are the Foreign Keys and referencing to the “id” of “Students”, the “id” of “Classes” and the “id” of “sections” respectively. The attributes “teacher_id” and “department_id” in table “Classes” are the Foreign Keys and referencing to the “id” of “department” and the “id” of “teachers” respectively. Also, the attributes “class_id” and “teacher_id” in table “sections” are the Foreign Keys and referencing to the “id” of “Classes” and the “id” of “teachers” respectively.

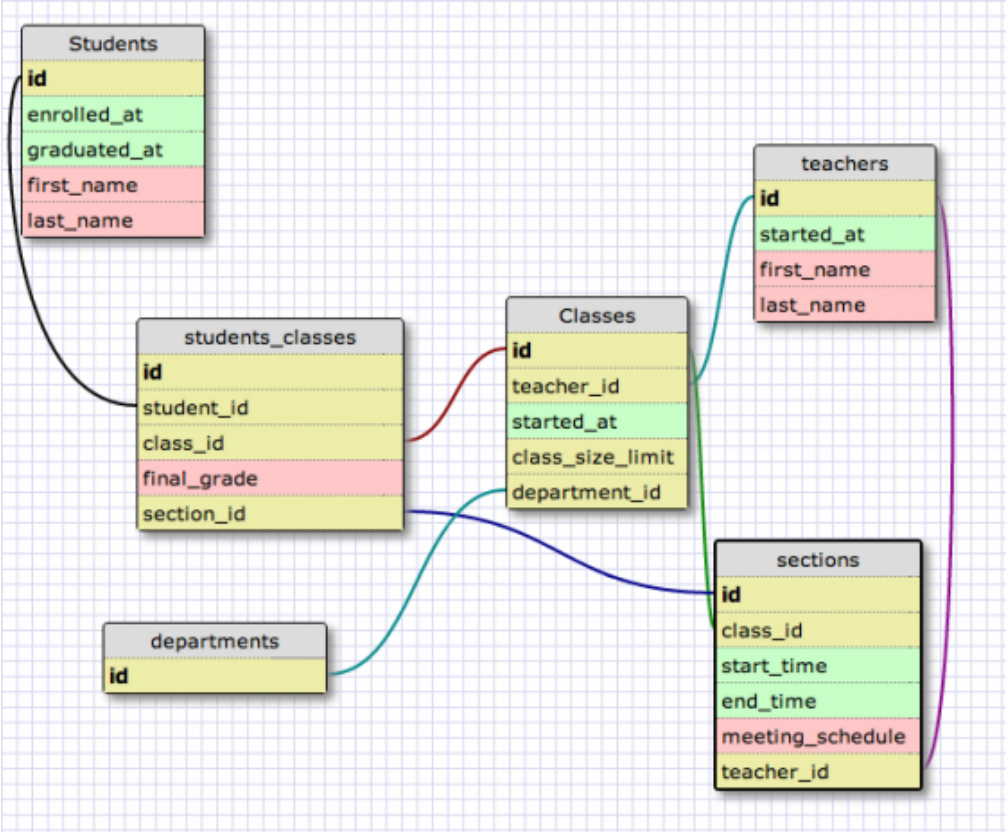

Figure 3.1: Primary Key and Foreign Key Example[79]

3.1.3 Database Normalization

In a database system, when a field repeated in two or more tables, data redundancy may occur. For example, as shown in Table 3.3, data redundancy is occurring in the "CourseID" and "TeacherID" column. Amount of data, such as "CMPT101", "CMPT102", "ARGI100" and"CHIN101", appears several times in the table. The redundancy of data may cause data inconsistency because the same attribute may store with different values, once they do not update consistently. Also, data redundancy increase the required amount of storage. In order to minimize data redundancy, Edgar Codd [20] introduced a concept of database normalization, meaning the process of organizing the attributes and tables of a relational database to minimize data redundancy. In addition, normalization broke down a complex domain into independent sub-domains, and each sub-domain can be linked with each other through the Primary Key-Foreign Key relation[23].

Table 3.3: Data Redundancy Example

ID	FirstName	LastName	CourseID	TeacherID
1	Yaowen	Chen	CMPT101	TEACH1001
			CMPT102	TEACH2003
			ECON203	TEACH3231
			STAT245	TEACH4013
2	Sam	James	CMPT101	TEACH1001
			HIST204	TEACH2243
			ECON403	TEACH3724
			MATH205	TEACH3053
3	Mike	Tom	CMPT204	TEACH1341
			ARGI100	TEACH2183
			CHIN101	TEACH1291
			MATH205	TEACH3053
4	Ryan	Jackson	CMPT101	TEACH1001
			CMPT102	TEACH2003
			ARGI100	TEACH2183
			CHIN101	TEACH1291

Furthermore, Codd [19][20] introduced three type of normal form, First Normal Form, Second Normal Form and Third Normal Form. Currently, the most of the relational databases meet the requirements of these three normal forms sufficiently, and the other normal forms, such as EKNF (Elementary Key Normal Form), BCNF (BoyceCodd Normal Form), ETNF (Essential Tuple Normal Form), and DKNF (Domain/Key Normal

Form) are more for academic purposes. Figure 3.2 represents a clear process of database normalization.

- First Normal Form

  First Normal Form (1NF) was defined by Edgar Codd [19] in 1971. If a relation is in the First Normal Form, each attribute of the relation should contain atomic values only, and the value of the attributes should be a single value. It ensures the repeating values are eliminated in an individual table, and each set of related data can be stored in a separate table.

- Second Normal Form

  Second Normal Form (2NF) is the second step to normalize the databases, introduced by Edgar Codd in 1971[20]. Besides following the 1NF, the Second Normal Form requires the every non-prime attribute of the table should be fully dependent on the entire primary key.

- Third Normal Form

  Third Normal Form (3NF) is designed to reduce the duplication of data and enforce referential integrity[20]. A relation is in Third Normal Form if and only if it is in 2NF and every attribute in the table is only dependent on the primary key and not on any non-prime attributes.

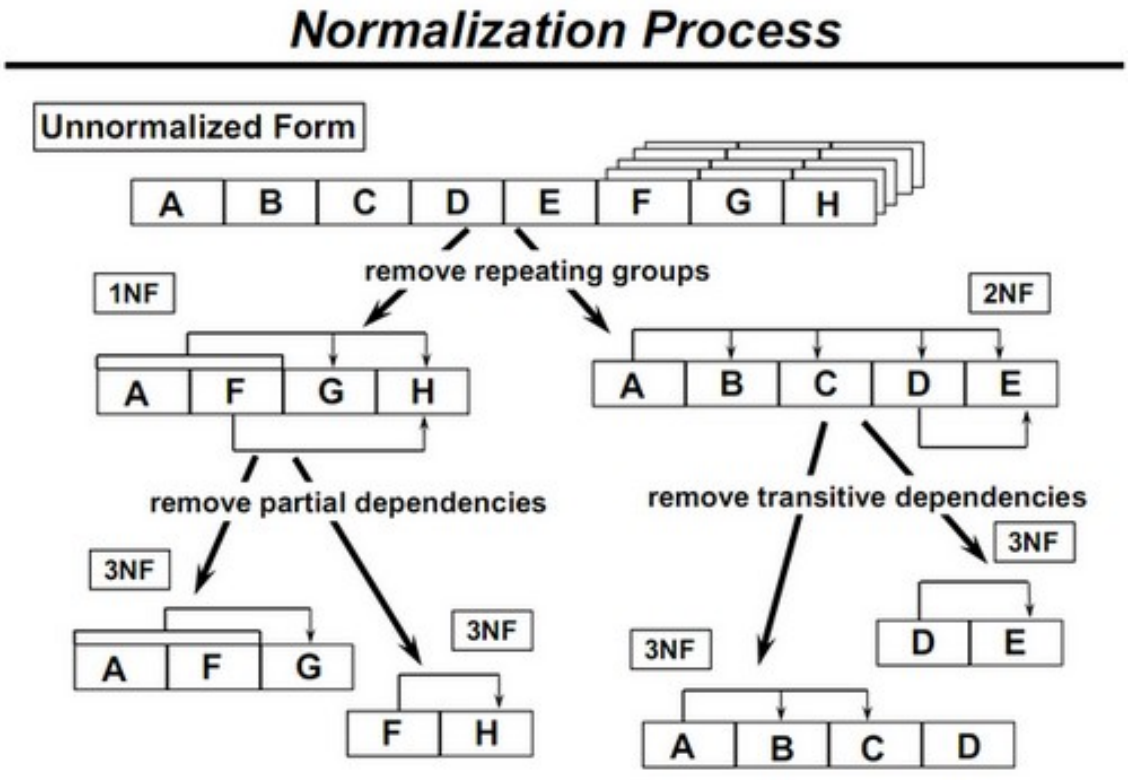

Figure 3.2: Database Normalization Process[1]

The database normalization enhanced the relational model and had an important effect on the success of the relational databases. This process decomposes the data into smaller relations and establishes more

meaningful relationships between them to reduce the need for restructuring the relations and make the relational model more informative to users[70]. Moreover, although database normalization process is widely used in RDMSs, it cannot be applied in every situation. Due to the scalability of the relational databases, it may need to add redundant copies of data to improve the readability sometimes. This process is called denormalization, and can improve the performance of database queries.

3.2 Graph Database

3.2.1 NoSQL

Because of the development of cloud computing with large-scale web applications, a new kind of database systems, namely NoSQL, has been adopted in the last decades[38]. NoSQL refers to "non SQL" or "non relational", so typically, the NoSQL databases apply a different mechanism for storage and retrieval of data other than the tabular relations, which is used in the relational databases[53]. NoSQL databases are becoming common in daily lives and used in more fields, and there are four major types of NoSQL databases, including key-value stores, column family stores, document stores and graph databases. Figure 3.3 represents the four categories of NoSQL graphically. Besides the graph databases, the other types of NoSQL are introduced as below:

- Key-value Stores

  Key-Value Stores store data as a typical hash table in a schemaless way[11]. The hash table contains two fields, a key with its value, so data is normally represented as a collection of Key-Value pairs in this model. Since the keys are the unique identification of the record, the data can be quickly found within the database by searching for the key. Because of the simple structure of this storage model, it provides high availability and scalability, and it is ease to use in applications, so it is very useful in distributed environment.

- Column Family Stores

  Column Family Stores store data in columns instead of rows, comparing with the relational databases. Also, it stores data in a Key-Value pair, but the key is two dimensional with a column key and a row key[42]. A column contains three elements: unique name (used to reference the column), value (the content of the column) and timestamp (used to determine the valid content). A Column Family is a collection of rows containing a number of columns, so the database can be viewed as one big table. Moreover, a row-by-row approach stores all data in a single entity, and a column-by-column approach ensures the information relating an attribute[56]. Thus, this database model is suitable for dealing with large-scale distributed data.

- Document Stores

  Document Stores allow applications to store data in terms of documents. The documents are indepen-

dent of each other, just like the rows in a relational database, but without any restrictions and not belonging to any schema[56]. Similar to the Key-Value Stores, the Document Stores take advantage from applying hash table. On the other hand, unlike relational databases, in which every instance of data has the same format as others, Document Databases store all related data together, and every instance of data can be different to others[15]. In addition, due to the semi-structure of the document, Document Stores offer good performance for large data sets and are flexible in dealing with data.

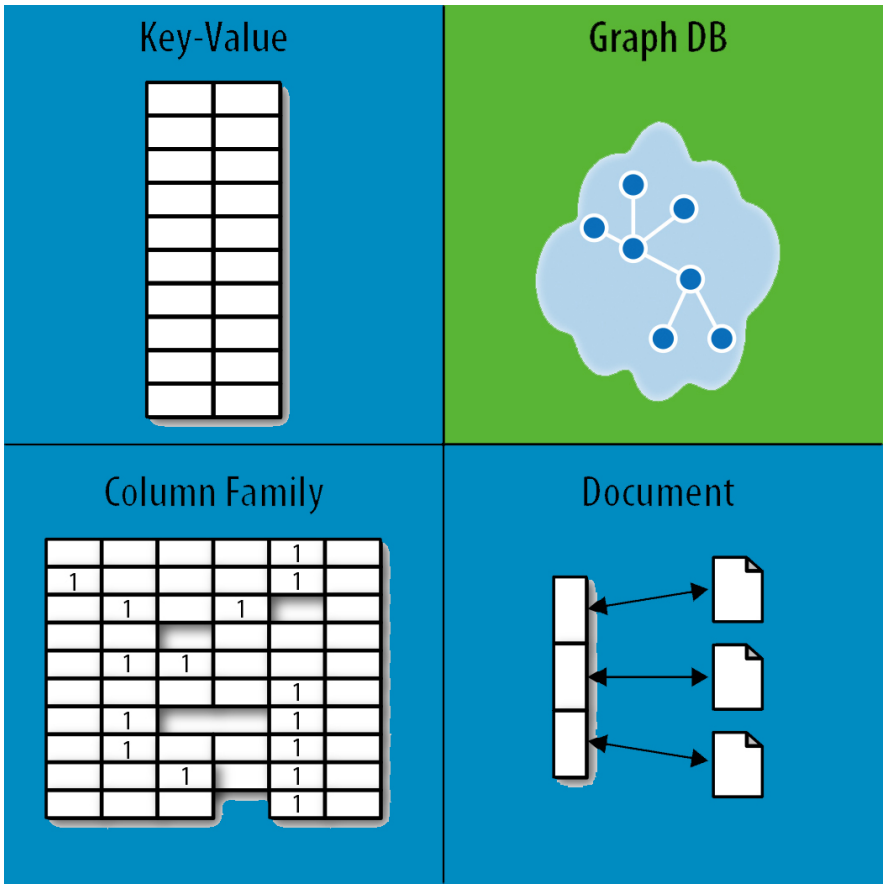

Figure 3.3: NoSQL Databases[67]

Besides these three types of NoSQL databases, the Graph database is another type of NoSQL databases. The Graph database applies graphs, which consist of nodes and edges, to store and manage data, instead of storing data in tables in a relational database[7]. Also, data can be stored as properties of nodes and edges in graph databases. The structure of graphs offers a high accessibility and scalability in a distributed environment.

Figure 3.4 represents a typical graph database example. There are three nodes in the graph, and each node contains its property, like Id, Name and Age. Also, there are six edges connecting the nodes with others to represent the relationship between nodes. An edge has its property as well, and in this case, the properties are Id, Label and timestamps. Remarkably, these edges can be directional, so the relationships

can be represented directly.

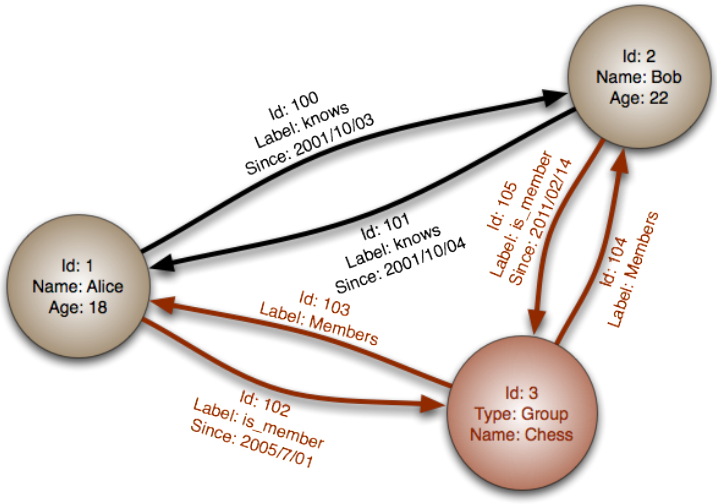

Figure 3.4: Graph Database Example[4]

Graph databases store data in term of graphs, which are a kind of highly interconnected data structure[33]. It is very useful for social networking websites, such as Facebook and Twitter, since it is easy to represent social actors as the nodes, edges as the relationship between users, and properties as the social information about users.

Currently, there still is very limited number of studies having been done about modeling data in the graph database domain, so it is difficult to provide hard rules based on practical applications. Robinson, Webber and Eifrem[65] suggest some general rules, including:

- Nodes can be used to represent entities.
- Edges can be used to represent relationships to connect the entities and to establish the semantic context for each entity.
- Node properties can be used to represent entity attributes.
- Edge properties can be used to express the strength, weight, or quality of a relationship.
- Relationship direction can clarify relationship more semantically. For bidirectional relationships, direction can be ignored.

Graph database is a type of databases, and there are three main group graph databases with different models, which are introduced in the following subsections:

- Property Graphs
- Hypergraphs
- Resource Description Framework (RDF) triples

3.2.2 Property Graphs

The property graph model is a model that is widely used in the most graph database systems. A property graph has the following elements[65]:

- A set of vertices:
    1. Each vertex has its unique identifier
    2. Each vertex has a number of incoming edges
    3. Each vertex has a number of outgoing edges
    4. Each vertex has a number of properties associated with it, defined by a map from key to value
- A set of edges:
    1. Each edge has its unique identifier
    2. Each edge has an incoming head vertex
    3. Each edge has an outgoing tail vertex
    4. Each edge has a number of properties associated with it, defined by a map from key to value
    5. Each edge has a label to denote the relationship between the incoming vertex and outgoing vertex.

Figure 3.5 represents an example of a property graph. It contains three social actors: "Alice", "Bob" and "Chris". They are nodes in the graph, and their information, such as name and age, are stored as the properties of nodes. Meanwhile, they are connected by edges, and the information about the relationships is stored as the properties of edges. It provides a clearer explanation of the relationships than the hypergraph, which is introduced in the following subsection.

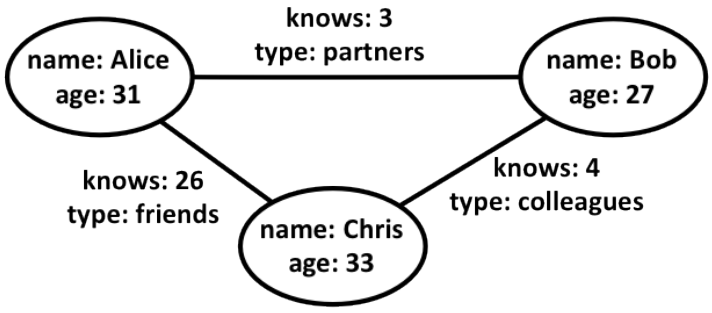

Figure 3.5: Property Graph[3]

3.2.3 Hypergraphs

Basically, graph databases store and represent data in terms of graphs, which comprise of nodes and edges, but unlike the property graph, which is an one-to-one relationship, a hyper edge can connect multiple nodes in the hypergraph model[45]. The graph model can easily store and process the many-to-many relationships, which is hard to handle in a property graph.

Figure 3.6 represents a hypergraph example. There are six vertexes in the graph, v1, v2, v3, v4, v5 and v6, four hyper edges, e1, e2, e3 and e4, connecting these nodes. e1 connects v1, v2, v3 and v4; e2 connects v1, v3 and v5; v3, v4 and v7 are connected by e3; similarly, e4 is the link between v5, v6 and v7.

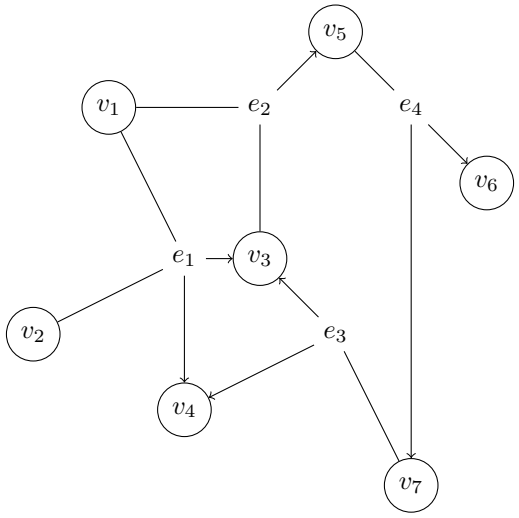

Figure 3.6: Hypergraph Example

Clearly, this model is very simple, but may require description for understanding the relationship. Also, because hyper edges are multidimensional, hypergraphs is a very general model, and it can be translated into a property graph with more relationships, but it increases the cost of storage.

3.2.4 RDF Triples

The Resource Description Framework (RDF) is a standard model for data interchange on the Web[62][8]. It has been used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax notations and data serialization formats. In addition, it decomposes any types of knowledge into small pieces, with some rules of the semantics, or meaning, of those pieces. There are some facts about the RDF format[62][48][59]:

- It is a data model in which the basic unit of information is known as an RDF triple (see Figure 3.7).
- The RDF represents information based upon the idea of subject-predicate-object expressions. It also could be considered as a resource identifier, an attribute or property name, or an attribute or property

value.

- To remove any ambiguity from the information stated by a given triple, the triple's subject and predicate must be URIs.

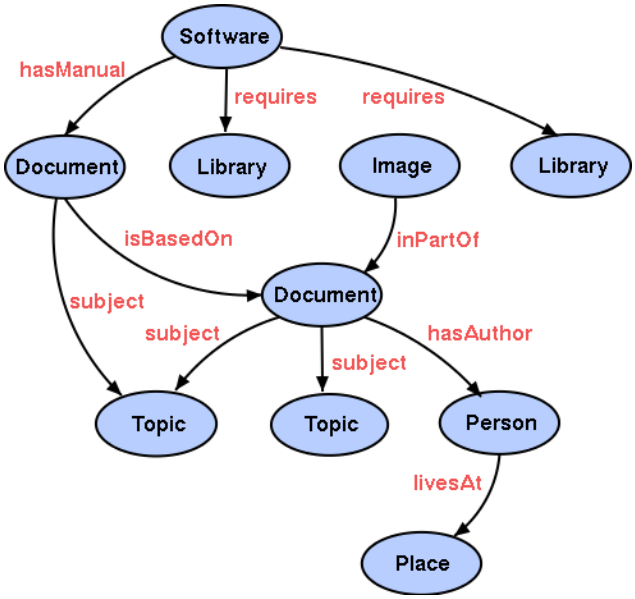

Figure 3.7: RDF Triple[2]

If converting the graph of the RDF triple in the Figure 3.7 into the " subject-predicate-object", it will be as follows:

Table 3.4: RDF subject-predicate-object

Subject	Predicate	Object
Sofeware	hasManual	Document
Sofeware	requires	Library
Document	isBasedOn	Document
Document	subject	Topic
Image	inPartOf	Document
Document	hasAuthor	Person
Person	livesAt	Place

Currently, RDF is one of the three foundational Semantic Web technologies, and widely used in the Semantic Web as a graph database. On the Semantic Web, there may be no sufficient information to determine

if two nodes are same or not. In order to resolve the identity problem, RDF applies the notion of the URI. URIs work very well for expressing identity on the WWW, so using the URI as a standard for global identifiers allows for a worldwide reference for any symbol[12]. It means that people can identify whether any two users in the anywhere in the world are referring to the same thing. This property of URI provides a simple way for a standards organization to specify the meaning of certain terms. The World Wide Web Consortium (W3C) has defined a number of standard namespaces for use with Web technologies including xsd: for XML schema definition; xmlns: for XML namespaces; and so on[62][8].

RDF uses the infrastructure of the Web to represent agreements on how to refer to a particular entity. The RDF standard itself takes advantage of the namespace infrastructure to define a small number of standard identifiers in a namespace defined in the standard, a namespace called rdf. rdf:type is a property that provides an elementary typing system in RDF. rdf:Property is an identifier that is used as a type in RDF to indicate when another identifier is to be used as a predicate rather than as a subject or an object[8][59]. In a social network, a typical RDF file that is written by XML can be shownas follows:

```
$<$?xml version=``1.0''?$>$
$<$rdf:RDF
xmlns:rdf=``http://www.w3.org/1999/02/22-rdf-syntax-ns\#''
xmlns:company=``http://company/Supervisor\#''$>$
$<$rdf:Description
rdf:about=`` http://company/Supervisor''$>$
  $<$Supervisor:name$>$Supervisor A$<$/Supervisor:name$>$
  $<$Supervisor:country$>$USA$<$/Supervisor:country$>$
  $<$Supervisor:company$>$Company A$<$/Supervisor:company$>$
  $<$Supervisor:gender$>$ Male$<$/Supervisor:gender$>$
  $<$Supervisor:age$>$36$<$/Supervisor:age$>$
$<$/rdf:Description$>$
$<$rdf:about=`` http://company/Supervisor''$>$
  $<$Supervisor:name$>$Supervisor B$<$/Supervisor:name$>$
  $<$Supervisor:country$>$USA$<$/Supervisor:country$>$
  $<$Supervisor:company$>$Company A$<$/Supervisor:company$>$
  $<$Supervisor:gender$>$ Male$<$/Supervisor:gender$>$
  $<$Supervisor:age$>$45$<$/Supervisor:age$>$
$<$/rdf:Description$>$
$<$/rdf:RDF$>$
```

3.3 Database Benchmark

Performance can be viewed as one of the major issues when evaluating different database hardware, software and configurations, and database benchmarks, which usually are running several of standard tests, are an important tool to assess the performance of databases. Benchmarking is a continuous and systematic process to compare the performance of applications, services and processes in order to improve the outcomes by identifying and implementing the best practice approaches. According to Lee and Jeong[46], the database benchmark normally can be classified into two categories: the generic benchmark and the custom benchmark:

- Generic Benchmarks

  This kind of benchmark is created to measure the performance and processes of organizations in unrelated industries. Because a generic benchmark is developed to be a domain specific benchmark to focus on a particular application domain, it can be generally used by a large group of customers with the applications to improve performance and processes as well as create new standards.

- Custom Benchmarks

  A custom benchmark is implemented by a particular customer for a specific application. It can measure the performance of a database precisely based on customer's need. However, because it is for a specific application, it cannot be used generally; the cost of designing and implementing the benchmark may be very expensive.

In recent years, because of the development of graph databases techniques, more graph database systems have emerged, so more work about the comparison on the performance of different graph databases has been done. In [26], Neo4j is compared with other scalable graph databases: Jena, HypergraphDB and DEX. The HPC scalable graph analysis benchmark tests the performance of each database for different typical graph operations and graph sizes. Although it tests 1k, 32k and 1M nodes from 9k relationships to 8.4 million, it is still a small amount of a large social network.

Also, Renzo Angles[7] proposed a way to compare current graph database models. That work compares some basic features of nine graph data stores, including AllergroGraph, DEX, Filament, G-Store, HyperGraphDB, InfiniteGraph, Neo4j, Sones and vertexDB. But the features about data storing and querying, like the data structures, query languages and constraints just provide a general view about the database systems. It is hard to know the performance of graph database systems on dealing with graph data.

In addition, there is some work comparing the performance of graph databases and relational databases. LinkBench is a recently developed benchmark, which is based on traces from the databases storing social graph data from Facebook, which is a major social networking website[9]. This work characterizes the data and query workload to construct the benchmark, LinkBench. Although this work has compared the performance of Neo4j with MySQL on storing social graph data in some fields, it lacks the comparison on the database properties.

Moreover, based on the artificial data model to simulate users of a social networking, the benchmark, BG, is used to evaluate interactive social networking actions[10]. Since there are numbers of interactive actions in a social networking application, such as posting new photos, browsing a profile, and generating a friend request, comparing such performances can be valuable in practical usage. However, BG compares SQL-X, MongoDB and CASQL, without any graph database systems, and it does not measure the capability of handling graph-type data.

Also, since there are different query languages available for different graph database systems, such as SPARQL for AllegroGraph (RDF Triple Store), Cypher for Neo4j (a widely used graph database) and GraphQL for some designed client applications, Holzschuher and Peinl[43] proposed a benchmark to performance of query languages running on different database systems. The query language is a great part in the success of a database system, but this benchmark does not focus on the performance of the database, which is the key to this study.

3.4 Social Network

A social network is a social structure made up of a set of actors such as individuals, groups and organizations, and a complex set of the edges tying between these actors[75][69]. The concepts of social networks can be applied in conjunction with the semantic web technology to form the Semantic Social Network. In this context, the social network perspective provides a clear way to analyze the structure of whole society[13]. Like the other networks, a social network also could be represented as a graph (Figure 3.8), G = (V, E), where V is the set of nodes representing people and E is the set of the edges. (V * V) means the relationships, such as friendship, kinship, and conflict, among the nodes, or saying people in the network. The social network is a map of all of the relevant ties between the nodes[30][69]. In a social graph, there can be strong ties, weak ties, and positive ties and negative ties.

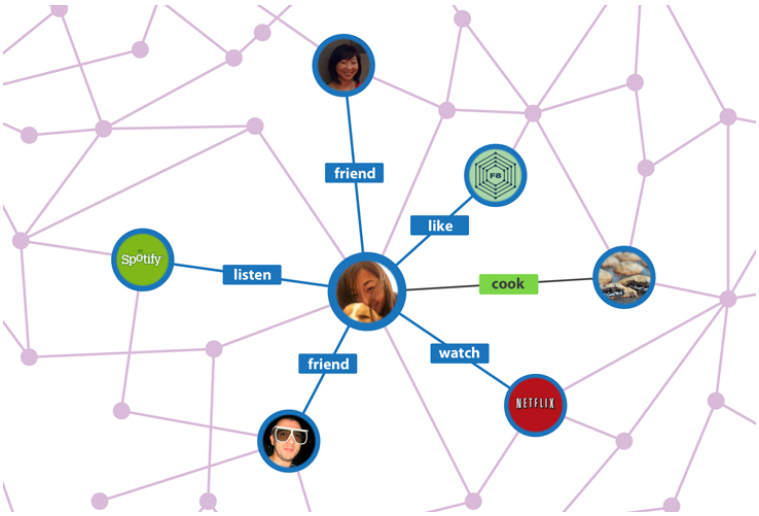

Figure 3.8: A Social Network in a Graph[25]

Definition: Strong Ties

Strong ties are the relationships that people are linked within the same community and involve frequent interactions[50][57][66]. People usually share the same body of knowledge and are able to transfer information quickly through the strong ties, because of the frequent interactions.

There is a hypothesis about strong ties that was defined by David Krackhardt[50]. Basically, strong ties can play a very important role in severe change and uncertainty. "People resist change and are uncomfortable with uncertainty. Strong ties constitute a base of trust that can reduce resistance and provide comfort in the face of uncertainty. Thus, it will be argued that change is not facilitated by weak ties, but rather by a particular type of strong tie[50]." This particular type of strong tie needs to fulfill the following three necessary and sufficient conditions[50]:

- Interaction: if A and B can be a strong tie, A and B must interact with each other.
- Affection: if A and B can be a strong tie, A must feel affection for B.
- Time: if A and B can be a strong tie, there must be a history of the interactions between A and B, and the interactions must be lasting over an extended period of time.

Definition: Weak Ties

Weak ties are relationships among people, who are not in the same community or coalitions. They are characterized with relatively infrequent interactions[32][31]. There is a hypothesis about the weak tie that was originally stated by Anatol Rapoport[63], a Russian-born American mathematical psychologist, in 1957. The hypothesis argues that if A is linked to both B and C, then there is a greater-than-chance probability that B and C are linked to each other (See Figure 3.9).

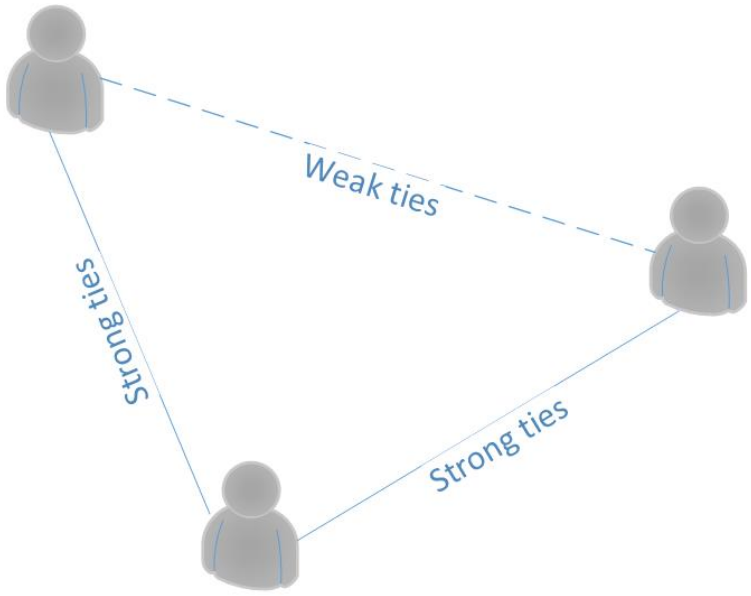

Figure 3.9: Strong Ties and Weak Ties

More specially, there are two randomly selected individuals, for example, A and B, from a set S, which contains A, B, C, D, E etc. of all persons with ties to either or both of them. If A have strong ties with both B and C, then, basing on the probability arguments, the B always has a tie with C. In other words, given the other two strong ties, the B-C tie is always presenting, whether weak or strong. The benefit of knowing people with weak ties is that the weak ties can function as the crucial bridges between any two dense clusters of close friends.

Definition: Positive Ties and Negative Ties

In most cases, acquaintanceship is a positive tie. On the other hand, animosity among persons could be a negative tie[63]. In order to represent both positive and negative relations, a model called signed graphs is created. Frank Harary[41][14] stated an important structure theorem for signed graphs. A signed graph is balanced if the product of the signs of all relations in every cycle is positive. On the other hand, if the product is ever negative, it will be unbalanced. The theorem represented that if a network of interrelated positive and negative ties is balanced, then it consists of two subnetworks such that each has positive ties among its nodes and negative ties between nodes in distinct subnetworks. In other words, "my friend's enemy is my enemy". Therefore, this theorem could be applied into the social analysis to find potential competitors.

In the social networks, there is a type of graphs in which most nodes can be reached from every other by a small number of steps. This kind of social network is defined as small-world network. According to [35][54], in the late 1960's, Stanley Milgram did a sociology experiment to investigate social networks. Milgram addressed letters to a particular stockbroker in New York and gave them to people randomly picked at various locations in the U.S.A, and far away from the receivers in that experiment. People could send the letters to people who they knew personally by first name. Eventually, most of the letters reached the destination, and the average number of steps was around six. That phenomenon was called "six degrees of separation". This kind of network, in which people are linked to each other by only a few links in a social network is known as a "small world network". This theory could be applied into social network analysis to find the links between people[76].

Moreover, the social network benefits people in their everyday lives, especially for the information transferring and searching. People could share information with their friends, and the friends will share this information with their friends as well, so there will be an exponential transfer of information. Especially on the Internet, people share lots of social information on the social network sites, and the information is spreading very quickly. For example, the followers retweeting a message on Twitter could easily make the message known by more and more users on Twitter.

On social network sites, there are some common social network features listed as follows[73]:

- Social actors:

  Every user must register a user account and log into the social network site when they are using the social network site. The social networking site should support several actor types besides users, such

as groups, organizations or companies.

- Social relations:

  There are several different types of social relations existing in current social networking sites. A user may need to confirm a relation when it is bidirectional such as the friendship in Facebook, but for the followers in Twitter, it is a unidirectional relation, which cannot be confirmed. Furthermore, the other types of relations between social actors are also noticeable such as blocking other users.

- Content:

  There could be multiple types of content being managed by social networking sites such as text posts, pictures, videos, events and links to external sites. Also, some content-oriented social networking sites may provide specific kinds of content. But the social networking sites are supposed to provide support for the most common content types. Furthermore, content visibility should support several options including keeping content private, sharing with contacts, sharing with specific users and sharing the content publicly.

- Activities timeline:

  The activities timeline is the stream of more-recent ordered actions performed by social actors. For example, "Yaowen posts a new status", "Yaowen uploads a new photo", and "Yaowen shares a video". The most popular timeline is the home activities timeline made up of all the activities in which the social actor?s followings are involved.

3.4.1 Social Network Analysis

Social network analysis is the analysis of social networks. It will view the social relationships in terms of the network theory. Therefore, the nodes will represent individual actors within the social network, and ties will represent relationships between the individuals, such as friendship, kinship, organizations etc. In addition, according to Scott in 1988, "Social network analysis has emerged as a set of methods for the analysis of social structures, methods which are specifically geared towards an investigation of the relational aspects of these structures. The use of these methods, therefore, depends on the availability of relational rather than attribute data[68]."

Therefore, the social network analysis is the study about the social relationships among a set of actors. It depends on the assumption of the importance of relationships among the social actors in the social network. There are lots of studies of social network analysis focusing on the relationships among social entities and the patterns and the implications of the relationships. As Wasserman described, "the social network perspective encompasses theories, models, and applications that are expressed in terms of relational concepts or processes. Along with growing interest and increased use of network analysis has come a consensus about the central principles underlying the network perspective[75]". Therefore, excepting the relational concepts, the following concepts are important[75]:

- Actors and their actions are regarded as interdependent rather than independent, autonomous units.
- Relational ties between actors are channels for the resources transfer or flow (either material or non-material).
- Network models focusing on individuals treat the network structural environment as providing opportunities for constraints on individual action.
- Network models conceptualize structures such as social, economic, political and so on, as lasting patterns of relations among actors.

As Robert A. Hanneman[39][40] stated, the social network analysis is more a branch of mathematical sociology than of statistical or quantitative analysis. Mathematical approaches to network analysis tend to consider the data as conclusive, which means the measured relationships and relationship strengths are regarded as accurately reflecting the final status of the network. Also it assumes the observations are regarded as the population of interest rather than a sample of some larger population of possible observations. On the other hand, statistical analysts tend to treat the particular scores on relationship strengths as probabilistic realizations of an underlying true tendency or probability distribution of relationship strengths. Additionally, it tends to consider a particular set of network data as a sample of a larger set or population of such networks or network elements.

There are lots of strategies for deciding how to collect measurements on the relations among the people within the social network. One of those approaches, full network method, can yield the maximum of information, but it will be costly and hard to execute; also, it may be difficult to generalize. The full network method needs to collect information about each actor's relationships with all other actors. This approach is taking a census of all ties in a population of actors, rather than a sample in fact. Because it collects information about ties between all pairs, full network data provides a detailed view of relations in the social network. Full network data is necessary to properly define and measure many of the structural concepts of network analysis[40].

The full network method tries to describe and analyze the social structures completely. However, the data of the full network is very expensive and difficult to collect, unfortunately. It is very difficult to obtain data from everyone in a group, and it is very challenging for every member to rate every other member. It only is possible in a very small group. When there is a limited number of specific individuals with whom they are tying with, the full network method will be more manageable. However, if the population of the social network is very large, the method is impossible to execute in the real world[40].

On the other hand, the snowball method gets considerably less information about the network structure, but it is less costly. Also, it allows easier generalization from the observations in the sample to some larger populations. This approach begins with a focal actor or a set of actors. Each of these actors is asked to name some or all of the actors who they are tying with. Then, the actors being named go to name the actors who they are tying with. The process continues until all actors are named[40][47]. Remarkably, with the

development of the Big Data technology, the large social companies, like Facebook, have more capability to process such approach to deeply analyze the social network to get more social information.

But there are two major potential limitations and weaknesses of the snowball method. First, there may be some actors having no connection with others, namely isolates. They cannot be located by this approach, so that the connectedness may be over-stated. Meanwhile, the presence and the number of isolates may be very important features of populations for some analytic topics. Second, there is no guaranteed way of finding all of the connected individuals in the group. It is important to start at a right place. Otherwise, it may miss a whole sub-set of the actors, who are connected but not attached to the starting points. One remarkable fact is that snowball method can be strengthened by selecting good initial nodes. In many studies, there may be a natural starting point. Although this approach may miss the isolators, the approach is very effective to capture the information of network[40][47].

3.5 Mobile Computing

According to Tomasz Imielinski and Henry F. Korth[44], the rapidly expanding technology of cellular communication, wireless LANs, and satellite services, has allowed information be accessed from anywhere at any time. This fact encourages more people to use laptops, mobile phones and tablets in their daily life. Regardless of size, the mobile devices can be equipped with a wireless connection to the fixed part of the network, or connect with other mobile devices. The resulting computing environment is called mobile computing that no longer requires people to maintain a fixed and universally known position in the network and enables almost unrestricted mobility. As James Bryan Zimmerman[80] pointed out, mobile computing enables improvements in information quality, information accessibility, operation efficiency, and management effectiveness.

- **Information Quality:**

  First of all, mobile computing enables information to be captured at its point of creation, once it has been created. Meanwhile, complex and complete information can be captured more quickly and easily than before. Therefore, the accuracy, relevance, completeness, conciseness, and scope of information will be improved in content dimension.

  Additionally, mobile computing has specifications for hardware and software, so that information can be viewed in the format that is easy for the mobile users to use and unambiguous to improve the form dimension of information quality by improving the characteristics of clarity, detail, order, presentation, and media. Therefore, mobile computing improves the time, content and form dimensions of information quality. Although the fidelity of mobile devices may be weak sometimes, for example, when there is the audio on phone calls, which results in very poor quality; or when the mobile device is out of range of the internet, generally speaking, the resulting overall quality of information generated is improved, since the most dimensions of information are improved.

- **Information Accessibility:**

Additionally, mobile computing improves the information accessibility as well. Mobile computing technology offers a wide range of options to fit the different needs of each individual. The improvements in information accessibility cause improved information flow both to and from the central fixed information system. When the mobile users transmit data to the fixed information system, the data can be processed in the information system, and then, the data will be available for all other users. Also, mobile computing can increase the information accessibility for the media such as facsimile, audio files, and images.

- **Operation Efficiency:**

  Moreover, operation efficiency can be improved from mobile computing. Mobile computing integrates the technology into the fixed information system, which allows the computing power and information contained within the fixed information system to be structured around the optimum workflow of a mobile worker, rather than altering the mobile worker?s workflow to meet the optimum configuration for computing. The example of news reporting, hotel operations, and health care show mobile technology can be applied to a diverse range of problems and achieve a similar improvement in operational efficiency.

- **Management Effectiveness:**

  Furthermore, mobile computing can improve the management effectiveness, based on the improved information quality, information accessibility, and operation efficiency. Mobile devices can provide more available updated and accurate information to managers, so that they can improve their ability to track work, which is in progress, and the capability to communicate with mobile personnel. Additionally, mobile devices also provide better information to mobile employees. The mobile devices allow them to make more informed decisions locally and minimize the cost and need for management decisions.

Because of the mobility of the mobile devices, people can benefit from using mobile computing, especially in the business. According to IT Policy Compliance Group[34], the biggest benefits of using smartphones and tablets include being able to access business information from anywhere, at any time, access to business applications, access to suppliers and partners from any location, and improve communication capability. On the other hand, although the mobile computing has advantage in mobility, it also has several limitations as follows[17][29][58]:

- **Security Risks:**

  Mobile devices are dependent on the public network normally, which can be accessed by lots of people. The mobile devices are easily attacked through a large number of networks interconnected. In addition, the security issue may be further complicated if users install unknown sources applications. Therefore, security is the main concern when considering mobile computing.

- **Power Issue:**

  Mobile devices are entirely dependent on battery power if external power is not available. Considering

the compact size of the many mobile devices such as smartphones and tablets, it is hard to obtain the necessary battery life in daily usage.

- **Human Interface with Devices:**
  Screens and keyboards are relatively small, due to the small size of the mobile devices. Meanwhile, some people may not be able to operate the screen-touch mobile devices accurately. These facts may cause the mobile devices to be hard to operate. Also, the alternate input methods, such as speech recognition, may still need to develop.

- **Transmission Interfaces:**
  Weather, terrain, and the range from the nearest signal points can have an effect on the signal reception of the mobile devices. Therefore, when in tunnels, some buildings, and rural areas, the reception can be very weak.

- **Potential Health Issues:**
  Some people, who use mobile devices when they are driving may be distracted from driving, and it may cause traffic accidents. Also, cell phones may interfere with sensitive medical devices. Moreover, there is a growing number of questions about cell phone signals causing the health problems.

- **Range and Bandwidth:**
  Mobile devices access to the Internet via the GPRS, HSDPS and LTE network currently. Generally, it is slower than direct cable connections, and available with a range of commercial cell phone towers. Additionally, high-speed wireless LANs only have very limited range.

3.5.1 Cross-platform Mobile Application development

In the past few years, there is a great evolution of the mobile computing industry; and there are lots of powerful mobile devices emerging with the mobile operating systems having better functionalities. Thus, there are many useful features for people such as Global Positioning System (GPS), Music, Camera, Accelerometer, etc., besides the normal tasks like making phone calls and sending text messages[64][71]. These kinds of built-in features make the mobile devices such as smartphones and tablets, more popular in people's daily lives[28].

Additionally, as mobile applications become increasingly popular with the growth of the mobile industry, the demand for high-performance mobile applications is increasing as well. However, with the emergence of different mobile OSs, such as iOS, Android and Windows Phone, it is really challenging for developers to build applications that can run on different OSs, since each OS has its own language, different API (application programming interface), and unique Integrated Development Environments (IDE)[37][6]. Therefore, the demand for cross-platform mobile application development frameworks is growing, in order to develop the applications for multiple OSs from the single code base. The main benefits from those frameworks include[55][6][71][72]:

- **Reduction of Programming Complexity**

  Since the different OSs may require developers to write the applications in specific programming languages, such as Objective-C for iOS and Java for Android, to build the native applications, developers need to be familiar with the different languages before writing applications for different OSs. If applying the cross-platform mobile application development frameworks, developers can develop applications using one programming language. It reduces the programming complexity for developers.

- **Code Reusability**

  The cross-platform mobile application development frameworks enable the applications to be compiled for different OSs from the single code base. Therefore, developers can reuse the existing code rather than taking time to rewrite it. Thus, applying cross-platform development can reduce the time and cost for development.

- **Reduction of Long-Term Maintenance Cost**

  Since the cross-platform mobile application development frameworks compile the mobile applications from the single code base, developers do not need to debug for several different code bases, and applying cross-platform development does not need to maintain a large staff to support each platform, so it reduces the long-term maintenance cost.

- **Decrement of Required Knowledge**

  For cross-platform development, developers do not need to know the specification of each platform, like API and IDE. Instead, they only need to be familiar with knowledge of the selected framework for development.

- **Sharing the Strengths of Technology**

  Some technologies have strengths in some tasks to make them easier than others. For example, the programmatic drawing may be easier to be written in HTML and JavaScript than in Java or C#. Developing equivalent applications in the native code may be more complex and time-consuming than applying the cross-platform mobile application development frameworks on development.

- **Increment of Market Share**

  An application for a particular mobile OS may face a limited number of corresponding users. If the developers develop the applications using the cross-platform mobile application development frameworks for multiple OSs, the number of users must be increasing due to the larger user base. Therefore, the occupation of market share will increase as well, and developers can get more from the development of applications.

In the past few years, a lot of cross-platform mobile application development frameworks have emerged. Meanwhile, there has been an explosion of improvement in the mobile application development, because of wide adoption of mobile devices and the fast-growing mobile application market. Those mobile applications

can be categorized into three types: the native application, the Web-based application, and the hybrid application[18][6][72] (See Figure 3.10).

Figure 3.10: Different Kinds of Mobile Application[72]

Native applications development for a mobile OS is the traditional way to build the mobile applications. The native applications have binary executable files that can be downloaded from the app stores, such as Apple's App Store, Android's Play Store and BlackBerry's App World, and installed on the devices. Once the native applications have been installed, users can launch the applications like the other services that the device provides, since the native applications interface with the OS directly, without any intermediary or container[6][72]. Also, the native applications can access all of the device's capabilities, like GPS, Music, and NFC, by mobile platform vendor. Users benefit from this, especially for the unique features and functionalities that are typical of some particular mobile OS.

For native application development, developers need to write the source code with the additional resource, like images, audio, and various OS-specific declaration file, and then, the source code is compiled in order to create an executable in binary form that can be packaged with the other resource and deployed into the devices.

In addition, the modern mobile devices, normally, have powerful browsers that support HTML5 capability, Cascading Style Sheet 3 (CSS3) and advanced JavaScript. It enables developers to build browser-based applications by using web technologies. One of the most prominent advantages of web-based development is its low barriers of entry. Because the web-based applications are entirely based on the web technologies, such as HTML5, CSS3 and JavaScript, rather than the other complex programming languages, like C#, Ruby or Objective-C, which may be challenging for unskilled developers, more experienced web developers can develop the mobile applications in standard web languages[28][51][37]. Also, this characteristic of web-based applications lowers the cost of development and maintenance.

However, unlike native applications, which interface with the OS directly, web-based applications run

within the browser, so there may be a limited number of the OS APIs that are exposed to the web-based applications that run inside the browsers. Compared with the native applications, which have full access to the device capabilities, many device capabilities are only partially available, even unavailable, for the web-based applications[18][6]. Although web-based applications may develop due to the improvement of HTML5, it still is the major limitation for web-based applications currently.

Furthermore, combining the native development with web technology provides the third type of mobile applications: hybrid applications. Remarkably, using the hybrid approach enables developers to benefit from both native development and web technology. On the one hand, the native shell allows applications to access the native APIs to take advantage of the device capabilities that the device offers[18][6]. On the other hand, the code can be written using web language and shared among the different OS, which makes the development and ongoing maintenance process shorter and cost-effective[72].

However, the hybrid applications cannot allow users access when the devices are not connected to the network, as the content is not accessible when offline. In addition, although packaging the web code into the native shell enhances the performance and accessibility, it does not support remote updates for the content[72][51]. Furthermore, since the hybrid approach does not produce truly native applications, the performance of these applications will not be on par with the native applications, although they can still be very good.

3.6 Summary

After reviewing the literature, there are lots of technologies that can be employed in my thesis research to address several problems, and the major finding from the literature review are listed in Table 3.5.

- **What are the advantages of applying graph databases?**
  Since graph databases store and process data in term of a graph, they provide high accessibility and scalability. In addition, since social data is suitable to store and process in term of a graph, the graph database can be applied in the social applications to handle graph-related operations.

- **How to compare the performance of different databases?**
  When benchmarking the database, it is necessary to develop some tests for evaluation. Based on the evaluation targets, we can design some tests to evaluate the databases. In addition, simulating social actions of a social application can provide more information about the performance of databases in practical usage.

- **How to enable portability for mobile client sides?**
  Applying the cross-platform mobile application development frameworks to develop the mobile applications can generate applications that can work on different mobile devices. These applications provide same features and functionalities for the clients, even though they are running on different mobile OSs

and mobile devices.

Table 3.5: Literature Summary

Area	Finding
Relational Databases	The relational model stores data into one or more tables of rows and columns[12][8]. Primary Key is a single or a set of columns with unique value for identifying each record[60]. Foreign Key is a field representing a reference to a Primary Key of another table [60].
ACID	ACID (Atomicity, Consistency, Isolation and Durability) are a set of properties guarantee the reliability of database transactions[36][27]. Atomicity means a transaction should be viewed as a entire unit[27]. Consistency means after transactions complete, the database should be from one valid state to another [61]. Isolation ensures each transaction should be independent to others[36]. Durability guarantees the changes in database remain the same once a transaction commits[27]

Continued on next page

Table 3.5 – *Continued from previous page*

Area	Finding
Database Normalization	Database normalization is the process of organizing the attributes and tables of a relational database to minimize data redundancy[19][20]. In 1NF, each attribute of the relation contains atomic values only[19]. In 2NF, each non-primary attribute of the tables should be fully dependent on the primary key[20] In 3NF, each attribute in the tables is only dependent on the primary key and not on any non-prime attributes[20]
Graph Database	Graph databases store and process data in term of a graph, which consists of nodes and edges [33][65]. There are three main group graph databases with different models: property graph, hypergraph, and RDF triple[65].
Hypergraph	Hypergraph applies hyper-edges to connect multiple nodse with one hyper-edge[45]. Hypergraph model can easily store and process many-to-many relationships [45].
RDF	RDF is a general method to decompose the knowledge into small pieces to describe or model the information as a RDF triple (subject-predicate-object)[62][48][59]. RDF applies the notion of the URI to determine the uniqueness of information on the World Wide Web[62][8].

Continued on next page

Table 3.5 – *Continued from previous page*

Area	Finding
Social Network	Social network could be represented as a graph with nodes and edges[75][69]. David Krackhardt[50] defined a strong ties hypothesis that strong ties are important in severe change and uncertainty. Anatol Rapoport[63] stated the weak tie hypothesis that argues that if A is linked to both B and C, and then there may be a connection between B and C. Frank Harary[41][14] stated a signed graph is balanced if the product of the signs of all relations is positive. Otherwise, it is unbalanced. Stanley pointed out the "six degrees of separation", which means people are linked to each other by only a few links in social network[35][54].
Social Network Analysis	Social network analysis will more focus on the social relationships[68]. Full network method collects information about each of actors connecting with all other actors to describe and analyze the social structure powerfully and yield maximum of information, but it is costly and difficult to execute[39][40]. Snowball method only asks the connected actors of the beginning node continuously. Though it may be less costly and easy to operate, it gets less information about the network structure, and may face the isolate problem[39][40].

Continued on next page

Table 3.5 – *Continued from previous page*

Area	Finding
Mobile Computing	James Bryan Zimmerman[80] stated mobile computing enables improvements in information quality, information accessibility, operation efficiency, and management effectiveness. Security risks, power issue, human interface with devices, transmission interfaces, potential health issues and range and bandwidth may be the limitations of the mobile computing[17][29][58].
Cross-platform mobile application development	Due to the specifications of different mobile OSs, there is an increasing demand for cross-platform mobile application[72][6]. Cross-platform mobile application development enables reduction of required skilled, code reusability, reduction of long term maintenance cost, decrement of required knowledge, sharing the strengths of technology and increment of market share[55][6][71][72]. Native applications offer better performance and the ability to access the device capabilities[72][6]. Web-based applications apply the HTML5 technology to lower the barriers of entry and the cost[72][6][29]. Hybrid applications share the advantage of native applications and web -based applications, but may not be on par with them[72][6].

However, there are still some open questions related to the research that need to be resolved, namely:

- How can enough social data be generated, especially the social graphs that are used for storing and processing?
- What are suitable criteria to measure database performance?
- Which capabilities of databases are important to deal with large-scale data?

Chapter 4

Design and Architecture

4.1 Data Generation

One key point in this research is to store and process large-scale social graph data. Unlike the random graphs, the social networks or social graphs are normally considered as highly right-skewed, which means only a small number of nodes are highly connected, called "hubs", and the large majority of nodes have a low degree, where the degree of a node is the number of its connections with other nodes[78]. The "hubs" play a very important role in the graphs, which are very different from random graphs. Figure 4.1 shows a normal social graph and a random graph. Clearly, in the social graph, there are several nodes having a high degree, and most of nodes in the graph can be reached via these nodes. On the other hand, in the random graph, each node has similar degrees, and the nodes are randomly connected.

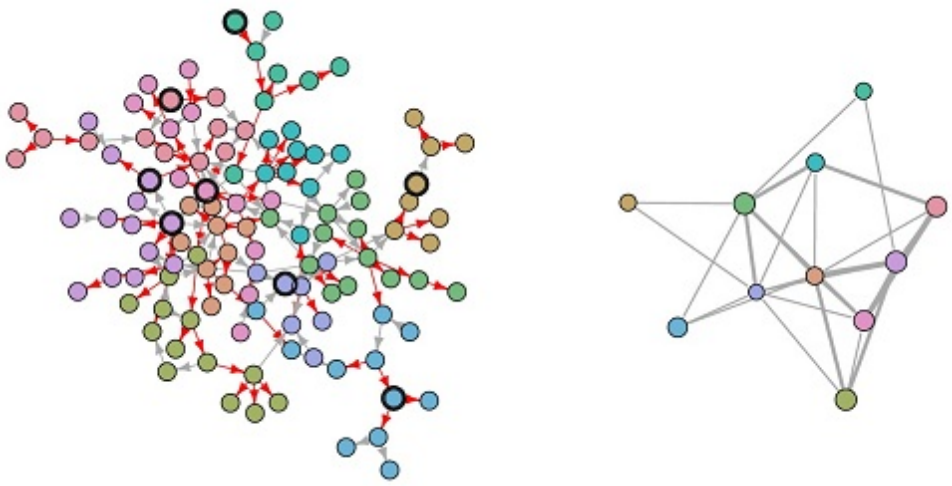

Figure 4.1: Scale-free Graph and Random Graph[22]

In this research, an algorithm called Recursive Matrix (R-MAT)[16] is applied to generate the social graphs, which have the "hubs" with a high degree. This algorithm starts at an empty N * N adjacency matrix A, where the matrix A represents a graph containing N nodes with the entry a (i, j) = 1 when the edge (i, j) exists, and 0 otherwise. The basic idea of the algorithm is to recursively subdivide the matrix into four equal-sized partitions, and add an edge into these partitions with unequal probabilities. Each generated

edge is distributed into one of the four partitions with probabilities $\alpha, \beta, \gamma, \delta$ respectively, and $\alpha+\beta+\gamma+\delta$=1. The chosen partition is subdivided into four smaller partitions recursively until it reaches the smallest cell, in this case, a 1 * 1 partition in the matrix, and the entry of this cell in the adjacency matrix will be changed to 1, meaning the cell is occupied by the edge. The procedure will process M times to generate M edges in the graph.

Algorithm 1 R-MAT Algorithm

1: **procedure** R-MAT($\alpha, \beta, \gamma, \delta$, A=$a_{ij}$) ▷ Input five parameters, $\alpha, \beta, \gamma, \delta$ are the possibilities, and $\alpha + \beta + \gamma + \delta$=1, A is a matrix containing N * N nodes
2: **if** $i == 1$ AND $j == 1$ **then**
3: $A_{ij} = 1$ ▷ Basic Case in the recursive
4: Subdividing matrix A into four equal size sub-matrix A_a, A_b, A_c, and A_d
5: Generate a random double r ▷ For comparing with the $\alpha, \beta, \gamma, \delta$
6: **if** $r >= 0$ AND $r<\alpha$ **then** ▷ Processing the algorithm into partition a of the matrix
7: R-MAT($\alpha, \beta, \gamma, \delta$, A_a)
8: **if** $r>=\alpha$ AND $r<\alpha + \beta$ **then** ▷ Processing the algorithm into partition b of the matrix
9: R-MAT($\alpha, \beta, \gamma, \delta$, A_b)
10: **if** $r>=\alpha + \beta$ AND $r<\alpha + \beta + \gamma$ **then** ▷ Processing the algorithm into partition c of the matrix
11: R-MAT($\alpha, \beta, \gamma, \delta$, A_c)
12: **if** $r>=\alpha + \beta + \gamma$ AND $r<1$ **then** ▷ Processing the algorithm into partition d of the matrix
13: R-MAT($\alpha, \beta, \gamma, \delta$, A_d)

Besides the large-scale social graph data, the social information of social actors in the social graphs should be generated as well. The social information contains first name, last name, age, gender, email and address. The first name and last name are 5-10 characters long, age is an integer between 0 and 100, and the gender is male or female. The email and address should be randomly generated as a string. Each social actor in the social graphs should be assigned an ID number. Also, the social relationship data would be generated as well. The social relationship data contains the Foreign Keys pointing to the people in the relationships, a string representing the type of relationships, and the date of the beginning time of the relationships.

4.2 Mobile Social Application

In order to simulate a practical social application, a simple mobile social application is proposed. This application should realize the fundamental functions of a mobile social application. The architecture of the application consists of three parts, including mobile client application, HTTP server, and the database backend, as shown in Figure 4.2.

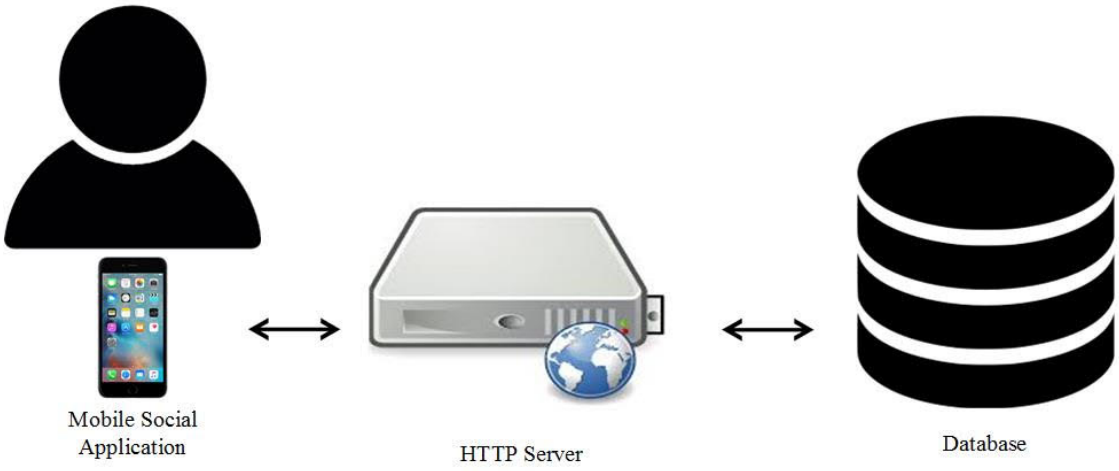

Figure 4.2: Architecture of Application

4.2.1 Architecture

The architecture of the simple mobile social application can be divided into three layers:

- **Mobile Client Application:**

  The client-side mobile application acts as the HTTP client side. Users can provide context information, view the social information, and send queries through the client application, so the main functionalities of this mobile application include capturing context information, sending requests to web server, and receiving the query data from the server side.

  Because currently, there are two major mobile OSs, iOS and Android, this application is implemented to enable users running it on difference mobile devices such as smartphones and tablets and various mobile OSs. Thus, the application will be developed with jQuery Mobile and PhoneGap, which is a set of JavaScript libraries for building mobile applications. The resulting application is cross-platform, which means it can run on different mobile devices and mobile OSs.

- **HTTP Server:**

  The server side is implemented by using Apache Tomcat, which is an open source web server and servlet container. The clients send the requests using Hypertext Transfer Protocol (HTTP) to the web server, and then the web server responds to each request, like capturing data from clients and returning query results to clients. Also, the server side needs to connect with the database backend. Moreover, the web service should be RESTful to provide scalability of component interactions and generality of interfaces.

- **Database Backend:**

  There would be two versions of this mobile application; one applies the relational databases as the data storage, and the other one uses the graph databases to store and process data. Specifically, in this case, MySQL is used in the relational databases version, and Neo4j is used as the backend graph

databases. Therefore, based on the performance of the different versions of this mobile application, the capability of the relational databases and the graph databases on handling large-scale social graph can be compared and evaluated.

4.2.2 Design Requirements

In order to achieve the goal of this research, this mobile application must meet the following requirements.

- **HTTP:**

  In the system, the client-side application should be able to GET and POST the social information to the server. Meanwhile, it also needs to support the push mechanism for the server side to return the data to the client side. Hypertext Transfer Protocol (HTTP) is an application-level protocol for the distributed hypermedia information system to format and transmit data. Also, it is the foundation of data communication on the World Wide Web (WWW). The application should be HTTP-based to enable the system exchanging and transferring the data.

- **Reliability:**

  Reliability guarantees the delivery of messages to the recipients in the context of distributed protocols. In this system, reliability is required to ensure the information delivery. The users send the social information to the server side, and when the server receives the information, it can store and process the information.

- **Web Service:**

  In order to provide a standard meaning of the interoperations between the mobile client applications that running on distributed devices, the server side of the system must be implemented in a RESTful way, which can be called RESTful API.

- **Consistency:**

  The consistency ensures the data from one valid state to another. Since the information will come from different clients, once the data is updated on the server side, the updated information should be disseminated to all corresponding client sides in time to keep the consistency of information.

- **Well Organized Information:**

  The entire information transferring on the networks must be in JSON format to keep the data organized, so the system does not need to transform the data format when processing it. It can reduce the time cost and increase the efficiency of the system.

- **Thin Clients:**

  A thin client means that the program depends on some other computers to fulfill its computational roles. In this system, the main functions are handled by the server side, and the front-end application

only needs to receive data, send data, and interact with the users. There is no complex computing logic that should be implemented on the client side.

- **Portability:**
  Portability refers to the usability of the same software in different environments. When this mobile application with the same functionalities is developed to run on various platforms, portability is the key issue on reducing development cost. In this research, the front-side client application is aiming to run on different mobile OSs such as Google's Android and Apple's iOS, so it needs to focus on the portability of the mobile application.

4.3 Kernel Description

This comparison is composed of six kernels, and each kernel involves a database operation whose performance is evaluated. The data used in the evaluation is generated by the mentioned algorithm above. The six kernels are described as follows:

4.3.1 Kernel One

The first kernel measures the performance of RDBs and GDBs on data insertion, and specifically, in this research, the data is the nodes and edges of the social graphs, and related properties. The data insertion operation is a very common in a database. For example, once a newcomer joins a social graph, the related personal information and the involving relationships should be inserted into the databases. The time spent to insert is the significant criterion of evaluation. Remarkably, the loading data should be generated before insertion, and the data generation time is not counted in this timing. In addition, the inserted data can be used for the following kernels to speed up the experiments.
The query used to insert personal information into MySQL could be:

Listing 4.1: Add people into MySQL

```
INSERT INTO People
(IdPeople, FirstName, LastName, Gender, Age, Email, Address)
VALUES
(1, 'Yaowen', 'Chen', 'Male', 27, 'test@email.com', 'test address');
```

and the query used to insert a relationship into MySQL could be:

Listing 4.2: Insert relationship into MySQL

```
INSERT INTO Relationships
(IdRelationship, FromPeople, ToPeople, Type, Since)
VALUES
```

```
(1, 1, 2,'knows', '1990-03-30');
```

The query used to create a node containing same data in Neo4j is:

Listing 4.3: Create nodes into Neo4j

```
CREATE (n:Person
{ id : 1, FirstName: 'Yaowen', LastName: 'Chen'', Age: 27,
Email: 'test@email.com', Address: 'test address'})
```

and the query used to insert the same relationship into Neo4j is:

Listing 4.4: Insert relationship into Neo4j

```
MATCH (a:Person),(b:Person)
WHERE a.id = x AND b.id = y
CREATE (a)-[r:Relationship{id:1, FromPeople:1, ToPeople: 2,
Type: 'knows', Since: '1990-03-30']->(b)
RETURN r
```

4.3.2 Kernel Two

This kernel measures the data searching performance of RDBs and GDBs. As mentioned above, the nodes and the edges of social graphs contain an amount of information. For example, the "hub" nodes are the nodes with high degrees, and searching such nodes may be a normal operation in lots of applications. This kernel is querying a set of edges that meet a condition; in this case, it tries to find all edges having types "know". The performance is evaluated based on the time spent to query.
The query used to find the requested edges from MySQL is:

Listing 4.5: Finding edges from MySQL

```
SELECT Relationship.FromPeople, Relationship.ToPeole
FROM Relationship
WHERE
Relationship.Type='knows';
```

The query used to find the same edges from Neo4j is:

Listing 4.6: Finding edges from Neo4j

```
MATCH (people1)-[:type:'knows']->(people2)
RETURN people1.id, people2.id
```

4.3.3 Kernel Three

This kernel measures the performance of the relational databases and the graph databases on graph traversal. According to the "six degrees of separation", everyone is six or fewer steps away from any other person in the world. People may be interested in how can they meet somehow based on the chain of "friend of a friend", and it can be very interesting in a social recommendation system. Simply, finding the shortest path between two people in a social graph can be a valuable strategy. In addition, breadth-first search is a basic algorithm for finding the shortest path in a graph. Therefore, this kernel is simply evaluating the performances on processing the breadth-first search in RDBs and GDBs. Remarkably, because recursion is difficult to handle in MySQL and it needs a sub-query to provide the related data to achieve breadth-first search, the lines of code increase exponentially (shown in the following queries). Therefore, in this test case, it will query data at level three and level four in the relationships from the MySQL database to simulate the graph traversal to simplify the evaluation. On the other hand, the similar graph traversal is processing in GDBs as well. Also, the spending time is used to evaluate the performance.

There are several queries needed to process BFS in MySQL, and the queries needed to BFS at depth three are listed as follows:

Listing 4.7: BFS at depth 1

```
SELECT Relationships.ToPeople AS depth1
FROM Relationships
WHERE Relationships.FromPeople = 1;
```

Listing 4.8: BFS at depth 2

```
SELECT DISTINCT Relationships.ToPeople AS depth2
FROM Relationships
WHERE Relationships.FromPeople IN
(SELECT Relationships.ToPeople AS depth1
FROM Relationships
WHERE Relationships.FromPeople = 1
)
```

Listing 4.9: BFS at depth 3

```
SELECT DISTINCT Relationships.ToPeople AS depth3
FROM Relationships
WHERE Relationships.FromPeople IN
(SELECT DISTINCT Relationships.ToPeople AS depth2
FROM Relationships
```

```
WHERE Relationships.FromPeople IN
(SELECT Relationships.ToPeople AS depth1
FROM Relationships
WHERE Relationships.FromPeople = 1
))
```

The query used to process BFS in Neo4j is much simpler:

Listing 4.10: BFS in Neo4j

```
START n=node(1)
MATCH p = n-[*1..]->m
RETURN p, length(p)
ORDER BY length(p) ASC
```

Meanwhile, in Neo4j, it can use the existing function shortestPath to find a single shortest path between two nodes:

Listing 4.11: Finding shortest path in Neo4j

```
MATCH (person1:Person { id:1}),(person2:Person { id:2}),
p = shortestPath((person1)-[*..15]-(person2))
RETURN p
```

4.3.4 Kernel Four

This kernel measures the performance of built-in functions of relational databases and graph databases. Both database system provides several built-in functions for querying data, and these functions offer users an easy way to operate the queries. In addition, the most functions that MySQL provides are different to the functions that Neo4j provides. In this research, two similar functions are chosen for comparing the performance. One is Ucase() (upper() in Neo4j), which returns a string or character converted to uppercase, and the other one is Lcase() (lower() in Neo4j), which returns a string or character converted to lowercase. The performance is evaluated based on the time spent to query.
The query uses Ucase() Function to convert the value of a field to uppercase in MySQL:

Listing 4.12: Using Ucase() in MySQL

```
SELECT UCASE(People.name)
FROM
People
```

and the query uses Lcase() Function to convert the value of a field to lowercase in MySQL:

Listing 4.13: Using Lcase() in MySQL

```
SELECT LCASE(People.name)
FROM
People
```

Similarly, the query uses UPPER() Function to convert the value of a field to uppercase in Neo4j:

Listing 4.14: Using upper() in Neo4j

```
RETURN upper( original )
```

and query uses lower() Function to convert the value of a field to lowercase in Neo4j:

Listing 4.15: Using lower() in Neo4j

```
RETURN lower( original )
```

4.3.5 Kernel Five

This kernel measures the performance of uniting two queries results from RDBs and GDBs to provide a complete query result. Everyone has its own social network, and it can be represented as a social graph. These graphs may contain lots of same nodes (social actors), so there is a possibility to unify the two small social graphs to generate a completed social graph. A graph union operation can achieve the target. In order to process graph union, another social graph will be loaded into the RDBs and GDBs. In addition, the time spent is measured to evaluate the performance.
The query used to unify query results in MySQL:

Listing 4.16: Union in MySQL

```
SELECT people1.FirstName
FROM people1
UNION
SELECT people2.FirstName
FROM people2
```

The query used to unify query results in Neo4j:

Listing 4.17: Union in Neo4j

```
MATCH (n:people)
RETURN n.FirstName AS name1
UNION
MATCH (n:people)
RETURN n.FirstName AS name2
```

4.3.6 Kernel Six

This kernel measures the capability of the relational databases and the graph databases on intersecting two graphs to find the same nodes between the graphs. Similar to the Kernel Five, this kernel is aiming to measure the performance of the relational databases and the graph databases on another common graph operation, graph intersection. People may have mutual friends, and they may appear in different individual social graphs. The graph intersection operation can query the same nodes, which are the mutual friends, from the two graphs. The loaded graphs in Kernel Five are used in this kernel as well, and the criterion of the performance is the execution time.
The query used to intersect in MySQL is:

Listing 4.18: Intersection in MySQL

```
SELECT people.id
FROM people1
WHERE people.id = 1
INTERSECT
SELECT people.id
FROM people1
WHERE people.id = 1;
```

The query used to interest in Neo4j is:

Listing 4.19: Intersection in Neo4j

```
MATCH (n {name:'a'}), (m {name:'b'})
RETURN FILTER(x IN n.knows WHERE x IN m.knows)
```

4.4 Summary

This chapter firstly introduces an algorithm called R-MAT to generate the social graphs that are used in the experiments. In order to generate the graphs matching the properties of the social graphs, the R-MAT algorithm is used to generate the highly right-skewed graphs.

Also, the architecture of the simple mobile social application is introduced as well. The application consists of three main components: mobile client application, HTTP server, and database backend. The functionalities of each component are explained with the design requirements.

Moreover, this chapter provides a detail explanation of the kernels of the database benchmark. The kernels contain six main operations: data insertion, data searching, graph traversal, database function, data union, and data intersection. Based on the performances of the relational databases and the graph databases

on such operations, it is easy to evaluate the capabilities of databases on storing and processing large-scale social graph data.

Chapter 5

Implementation

This chapter consists of three sections: workload characterization, hardware and software setting, and mobile application implementation. Workload characterization introduces the data schema of the relational databases and the corresponding graphs that are stored in the graph databases. In addition, the hardware and software setting section provides information about the hardware environment, and software tools that are used in the thesis research. Moreover, the mobile application implementation section describes the implementation of the cross-platform mobile social application. More information is in the following sections.

5.1 Workload Characterization

The generated social graphs that are used to be stored in the relational databases and the graph databases comprise many social actors, which are represented as the nodes, relationships, which are represented as the directed edges, and attributes, which are represented as the properties of nodes and edges. The relational database schema, which is representing social actors and social relationships, is showing in Figure 5.1. In order to simplify the relationships, in this thesis research, there are two tables that follow the 3NF storing in the relational databases. The People Table is used to store the data related to the social actors, and the Relationship Table is used to store the relationship data.

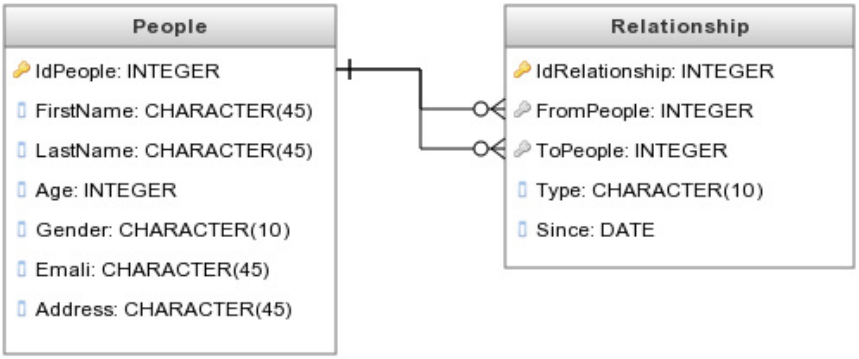

Figure 5.1: Data Schema

In the People Table, the IdPeople is the unique primary key of the People table; FirstName and LastName

are character strings that store the name of the social actors; there are two options, Male and Female, for users to store Gender; Age is an integer that stores the age of people. Email is a string to store the email address, and Address is a string to store the address of people.

Additionally, in the Relationship Table, besides the primary key of the table, IdRelationship, there are four attributes, including FromPeople, ToPeople, Type and Since. FromPeople and ToPeople are two foreign keys in the table referring to the primary key of the People Table for identifying the direction of the relationship and people involved in the relationship. Type is used to show the relationship type, such as know, employers, and friends. Since is a date data field to show the beginning time of the relationship.

Although the data stored in the relational databases and the graph databases is the same, the representation is totally different. A typical representation of data storing in the graph databases is showing in Figure 5.2, and all of the data is represented as the nodes and edges.

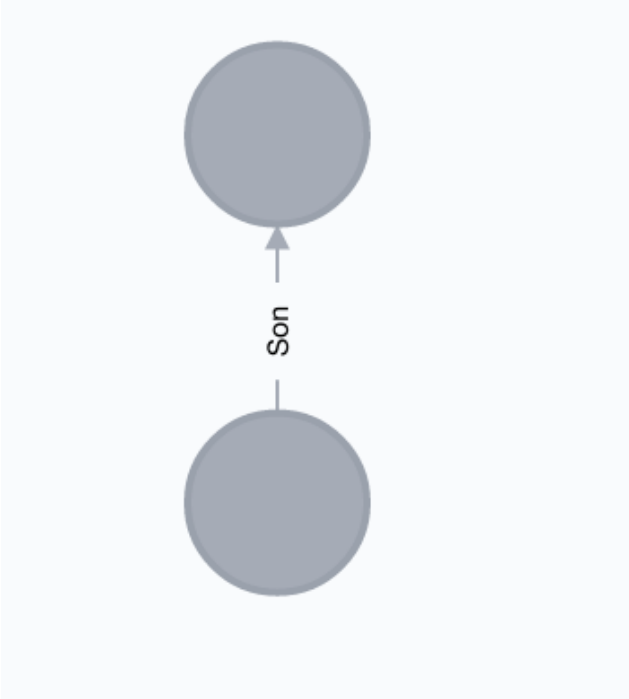

Figure 5.2: Graph data example

5.2 Hardware and Software Setting

In this thesis research, there are two devices, iPhone 6s running iOS 9.2.3 and Nexus 5 running Android 5.0, installing the cross-platform mobile social application, and being used in the experiments. The reason to use two devices is to test the performance of the mobile application on both mobile OS. Meanwhile, the HTTP server side is prepared as well. It runs on a Lenovo desktop with the following specifications:

Windows Edition:

Windows 10 Pro

System:

Processor: Inter(R) Core(TM) 2 Quad CUP Q9450 @ 2.66GHz
Installed memory (RAM): 4.0 GB
System type: 64-bit Operating System

Meanwhile, the graph databases (Neo4j) and the relational databases (MySQL) are running on another Lenovo desktop with the following specifications:

Windows Edition:

Windows 10 Pro

System:

Processor: Inter(R) Core(TM) i7-3770 CUP @ 3.40GHz
Installed memory (RAM): 16.0 GB
System type: 64-bit Operating System

5.2.1 Neo4j

The graph database, Neo4j, provides an Admin Web Console for users to accomplish the functions of databases, such as creating data, running queries and searching data graphically, to control the database. The main page of the Admin Web Console is showing in Figure 5.3. In addition, the Admin Web Console can provide some metadata about existing data that stored in the databases. Clearly, the number of nodes, relationships, properties and relationship types are representing directly. Also, the database disk usage and logical log disk usage are clear for users.

Neo4j also offers web browser to view the data. Users can query data and view the results in the browser directly. Fox example, when users try to find the people storing in the graph databases, the results are showing in Figure 5.4.

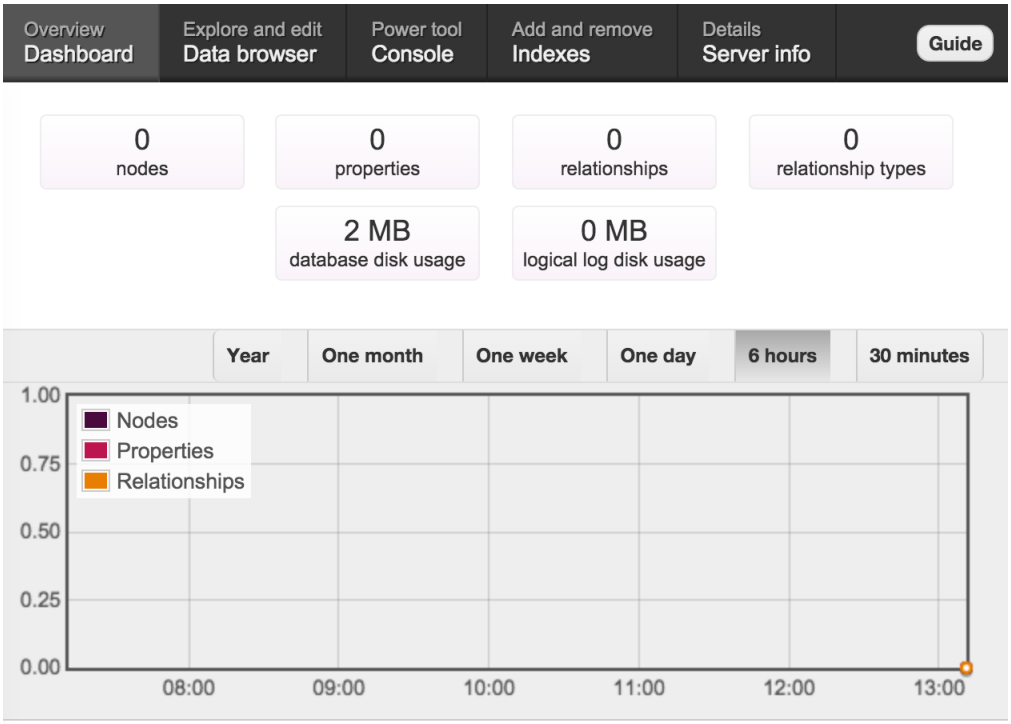

Figure 5.3: Neo4j Web Admin Consolo

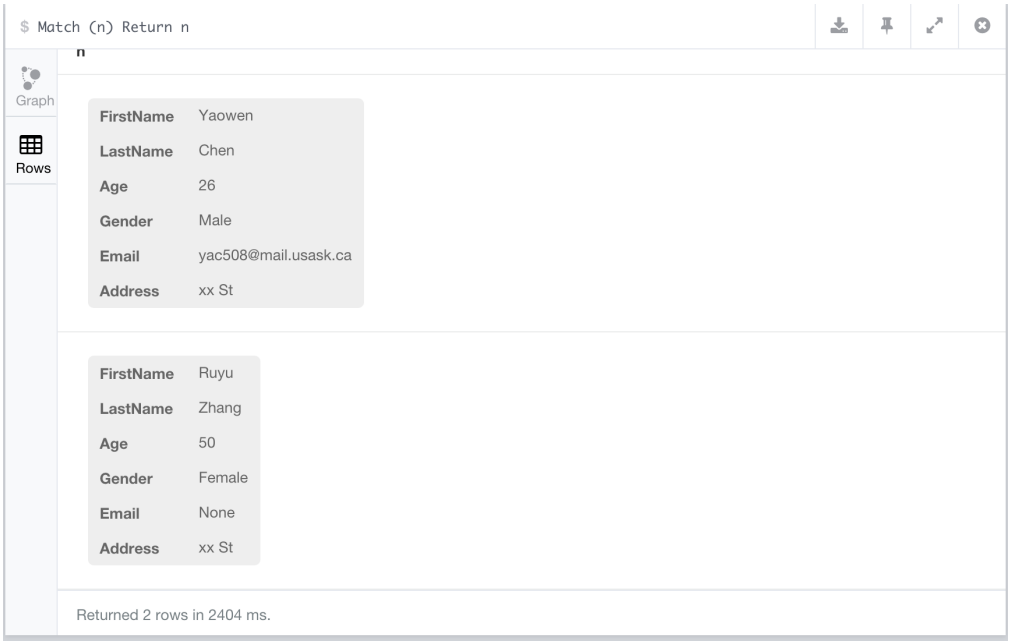

Figure 5.4: Neo4j Web Browser

5.2.2 MySQL

On the other hand, in order to manage the MySQL databases, this research applies MySQL Workbench, which is a visual database tool for integrating SQL development administration, database design, creation and maintenance of the MySQL database system into a single integrated development environment (IDE). Figure 5.5 shows the interface of the MySQL workbench.

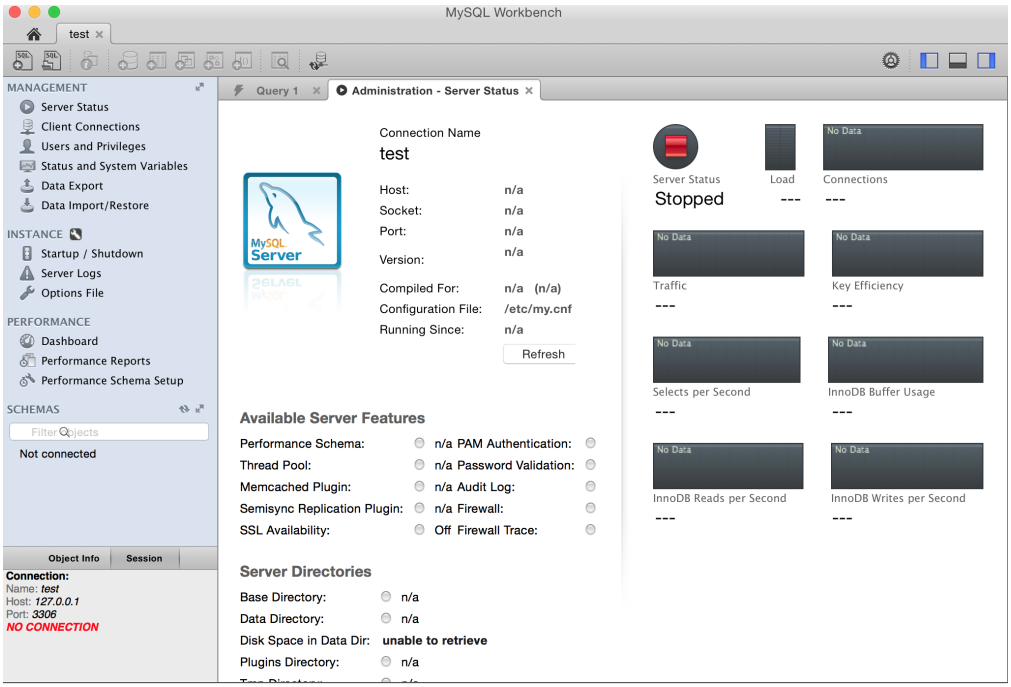

Figure 5.5: MySQL workbench

5.3 Mobile Application Implementation

In my research, a cross-platform mobile social application is developed by applying PHP+Database system+Apache, jQuery Mobile, and PhoneGap. First of all, PHP + Database system + Apache form the development environment for the mobile application. PHP is a widely-used open source general-purpose scripting language that is especially suited for Web development because it can produce dynamic Web pages and be embedded into HTML. Also, the database system is used to manage the data storage, and the Apache HTTP Server is a web server software playing a key role in the web application implementation. The mobile application is implemented by using PHP as the building language, and Apache as the web server.

Remarkably, because this research aims to compare the capability of the relational databases and the graph databases on storing and processing the large-scale social graph data, this mobile social application has two versions. One applies the relational database, MySQL, as the data storage, and the other one applies the graph database, Neo4j. Besides the backend storage, the other features of the two versions are almost

the same.

In the MySQL version, the application applies the following code to access the databases:

```
<?php
$host="localhost:3306"; // Host name
$username="root"; // Mysql username
$password=""; // Mysql password
$db_name="SocialJudgement"; // Database name
$tbl_name="UsersLogin"; // Table name
// Connect to server and select databse.
mysql_connect("$host", "$username", "$password")or die("cannot connect");
mysql_select_db("$db_name")or die("cannot select DB");
?>
```

Also, the system needs to use MySQL to query data from the relational databases. For example, the following code is used to check the login information:

```
<?php
// username and password sent from form
if($_POST['username']==""){
echo "<Script language='JavaScript'> alert('Please enter an username');
</Script>";
echo "<Script language='JavaScript'> history.go(-1); </Script>";
}else{
$username=$_POST['username'];
}
$password=$_POST['password'];
$sql="SELECT * FROM $tbl_name WHERE UserName='$username' and
Password='$password'";
$result=mysql_query($sql);
$db_field = mysql_fetch_assoc($result);
// Mysql_num_row is counting table row
$count=mysql_num_rows($result);
// If result matched $myusername and $mypassword, table row must be 1 row
if($count==1){
$_SESSION[?user?]=$db_field['idUsersLogin'];
$_SESSION[?name?]=$username;
$_SESSION[?log?]="true";
echo "<Script language='JavaScript'> alert('Login Successfully');
```

```
</Script>";
echo "<Script language='JavaScript'> window.location='main.php';
</Script>";
} else if ($password !=""){
echo "<Script language='JavaScript'> alert('Wrong Password');
</Script>";
echo "<Script language='JavaScript'> window.location='index.html';
</Script>";
} else if ($password ==""){
echo "<Script language='JavaScript'> alert('Please enter Password');
</Script>";
echo "<Script language='JavaScript'> window.location='index.html';
</Script>";
}
?>
```

On the other hand, the Neo4j version uses the following code to connect with the graph databases:

```
<?php
use Everyman\Neo4j\Client,
Everyman\Neo4j\Transport,
Everyman\Neo4j\Node,
Everyman\Neo4j\Relationship;
$host="localhost:"; // Host name
$port="7474"; // Port number
$username="root"; // Neo4j username
$password=""; // Neo4j password
$client = new Everyman\Neo4j\Client($host, $port);
$client->getTransport()
->setAuth($username, $password);
?>
```

In addition, the following code is using Cypher to query data to achieve the login function which is the same as the MySQL version:

```
<?php
// username and password sent from form
if($_POST['username']==""){
echo "<Script language='JavaScript'> alert('Please enter an username');
```

```
</Script>";
echo "<Script language='JavaScript'> history.go(-1); </Script>";
}else{
$username=$_POST['username'];
}
$password=$_POST['password'];
//Get password from Neo4j
$passwordInDB->getProperty($username)
if($password==$passwordInDB){
$_SESSION[?user?]=$db_field['idUsersLogin'];
$_SESSION[?name?]=$username;
$_SESSION[?log?]="true";
echo "<Script language='JavaScript'> alert('Login Successfully');
</Script>";
echo "<Script language='JavaScript'> window.location='main.php';
</Script>";
} else if ($password !=$passwordInDB){
echo "<Script language='JavaScript'> alert('Wrong Password');
</Script>";
echo "<Script language='JavaScript'> window.location='index.html';
</Script>";
} else if ($password ==""){
echo "<Script language='JavaScript'> alert('Please enter Password');
</Script>";
echo "<Script language='JavaScript'> window.location='index.html';
</Script>";}
?>
```

In addition, jQuery Mobile has been used to develop the mobile application running on the mobile devices. jQuery Mobile is a touch-optimized web framework that is compatible with a wide variety of smartphones and tablets[**?**]. Also, the jQuery Mobile framework is compatible with other mobile app frameworks and platforms such as PhoneGap. Thus, it provides the opportunity for applying both jQuery Mobile and PhoneGap in a project. The following Figure 5.6 is a screenshot of the login page.

Basically, the core of implementation the website using jQuery Mobile is linking jQuery Mobile Libraries and stylesheet (CSS file) on every page. Thus, when applying JQuery Mobile, it needs to include the related lib in the file. The following codes of login page include jquery.mobile-1.4.5.min.css, jquery-1.11.1.min.js and jquery.mobile-1.4.5.min.js in the head.

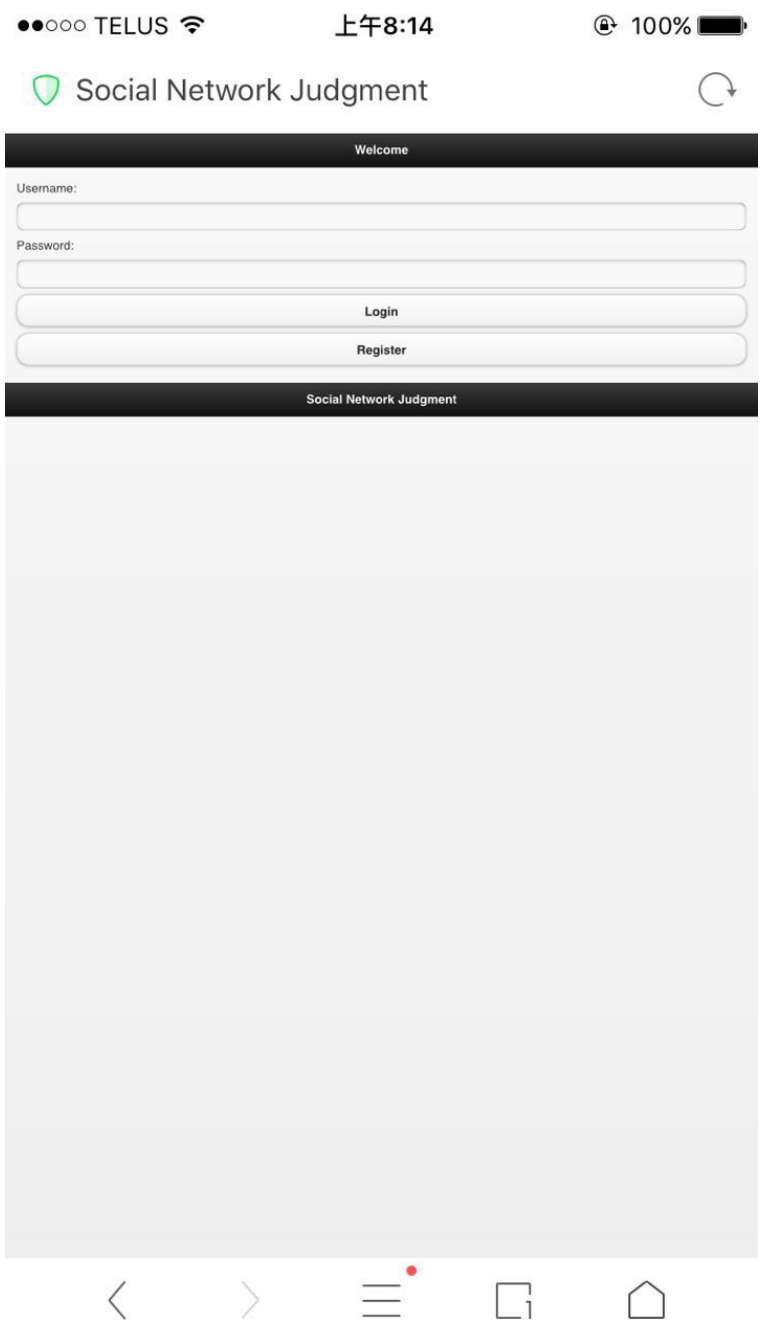

Figure 5.6: Login Page

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Social Network Judgment</title>
<link rel="stylesheet"
href="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css" />
<script src="http://code.jquery.com/jquery-1.11.1.min.js">
</script>
<script src="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js">
</script>
</head>
<body>
<div data-role="page" id="login">
<div data-role="header">
<h1>Welcome</h1>
</div>
<div data-role="content" data-inset="true">
<form id="loginForm" action="checklogin.php" method="post">
<fieldset>
<label for="username">Username:</label>
<input type="text" name="username" id="username" value="" />
<label for="password">Password:</label>
<input type="password" name="password" id="password" value="" />
<input type="submit" value="Login" id="submitButton">
<input type="button" value="Register" id="register"
onclick="window.location='registerPage.html'">
</fieldset>
</form>
</div>
<div data-role="footer">
<h4>Social Network Judgment</h4>
</div>
</div>
</body>
</html>
```

Moreover, in order to enable the social application running on the mobile devices, I apply PhoneGap, which is an open source mobile development framework, in the cross-platform mobile social application. PhoneGap enables software programers to build applications for mobile devices by using JavaScript, HTML5, and CSS3, instead of device-specific languages such as Object-C for the iOS system and Java for the Android system, and the resulting applications are hybrid. Therefore, the resulting apps are not truly native, because all layout rendering is done via web views instead of the platform's native User Interface framework. Also, they are not purely web-based applications, because they are not just web apps, but are packaged as apps for distribution and have access to native device APIs. PhoneGap enables the application running on lots of the mobile operating systems, such as Apple iOS, Google Android, and Microsoft Windows Phone. In this project, I apply Adobe PhoneGap Build, which packages mobile apps with PhoneGap in the cloud.

The application workflow is basically shown in the following graph.

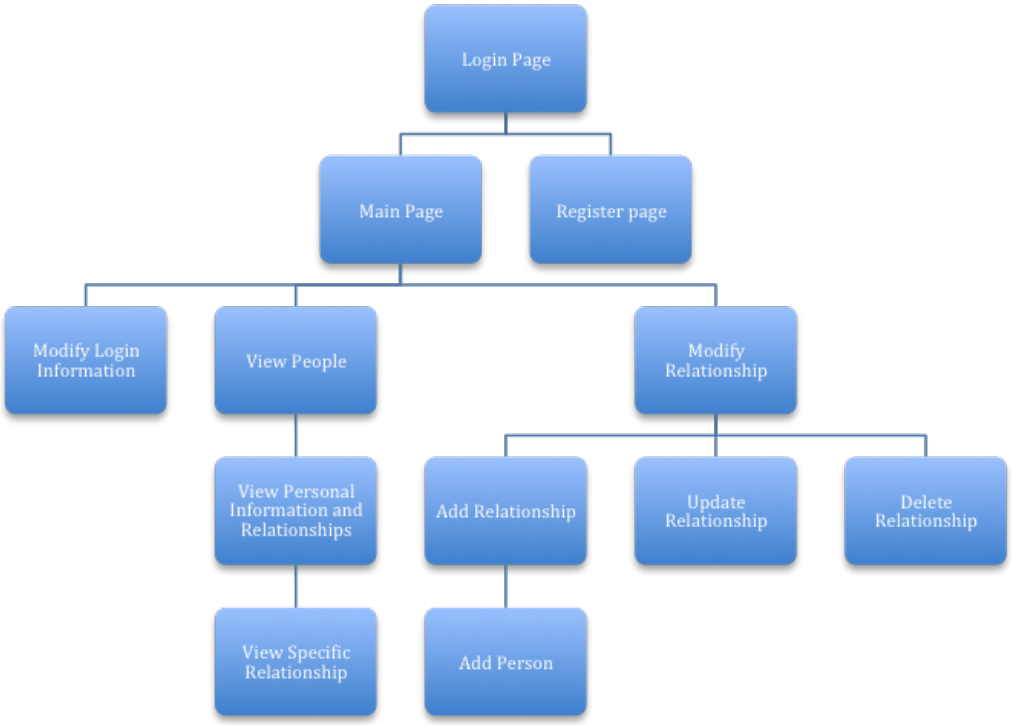

Figure 5.7: Application Workflow

More details about each page is shown as follows:

- **Login Page:**

  This page is used for users to login with their username and password. After users login, the page will jump to the main page. If users have no account, they could go to the register page to register.

- **Register page:**

  This page is used for users to register. Users need to provide their age, Email and gender when they

register. After users register successfully, the system will jump to the login page.

- **Main Page:**

  This is the main page of the application, and there are three options, Modify Login Information, View People, and Modify Relationship, for users to do.

- **Modify Login Information:**

  In this page, users could change their login information, such as the password.

- **View People:**

  This page provides a list of people having stored in the databases. Users could click one to view more specific information.

- **Modify Relationship:**

  In this page, users could manage the relationships they have added. There are three options, add, update, and delete.

- **View Personal Information and Relationship:**

  In this page, users could view the personal information and the involving relationships.

- **Add Relationship:**

  If users want to add a new relationship with others, they will use this page.

- **Update Relationship:**

  If users make any mistake when they add a relationship, they could fix the mistake in this page. Also, once the relationships are changing, users can update as well.

- **Delete Relationship:**

  Also, users can delete the relationships, which they added in the system.

- **View Specific Relationship:**

  In this page, users could view the judgments and votes from others.

- **Add Person:**

  If the subject person is not in the system, users could add them by providing the information of the subject person.

Chapter 6

Experiments

In order to evaluate the performance of the relational databases and the graph databases for handling the large-scale social graph data, two kinds of analytics are taken into the comparison to provide a complete view of the capabilities. Firstly, the quantitative analysis is applied to analyze the measurable and verifiable data and statistics, like storage cost and execution time, which are collected from the experiments, to develop and employ theories about the storage approaches. Secondly, the qualitative analysis is used to provide the empirical support for the performance evaluation based on the analysis of the performance of the relational databases and the graph databases on five fields, including maturity and level of support, ease of programming, flexibility, security, and data visualization.

6.1 Quantitative Analysis

Two kinds of measurable and verifiable data, the storage cost and the execution time, are collected from the experiments for quantitatively evaluating the performance of the relational databases and the graph databases when processing large-scale social data. Firstly, the storage cost should be considered as an important criterion in evaluating the performance of a storage approach, since it can influence query performance. Also, the execution time is another important metric in the evaluation, and a storage approach has better performance with shorter execution time.

6.1.1 Storage Cost

Firstly, the storage cost can impact the database performance, so the storage usage should be an essential metric on the database performance. Each time users need to read a piece of data from a database, it also needs to retrieve the data from disk, which will cause a disk I/O operation. Also, the data storing in the database is located in a number of different physical pages, which are the basic internal structure that organizes the data in the database files, and an I/O operation is required to retrieve the data for every page. Therefore, minimizing the record size storing in the database can maximize the number of records that each data page can store, and then improve the retrieving efficiency to increase the query performance.

For example, the default single physical page is 4k bytes long in a MySQL server, so when it needs to retrieve 10,000,000 records from a table in a MySQL Server and each record is 400 bytes long, the entire

10,000,000 records require 1,000,000 data pages, and each data page can store ten different records. Because an I/O operation is required for every data page, it needs 1,000,000 I/O operations to read every record in the large table. When the data size become smaller, for example, saving 40 bytes for every record to make each record 360 bytes long, each data page can store 11 different records to reduce the number of I/O operation to be 909,090. Clearly, saving the data size of data from each record stored in the databases to reduce the storage size can improve the database performance. The more data storage saving, the more the performance can improve.

Table 6.1: Databases with Size

Nodes Number	Edges Number	MySQL Size	Neo4j Size
1000	3000	0.32 MB	1.08 MB
5000	15000	1.61 MB	2.73 MB
10000	30000	3.37 MB	3.90 MB
50000	150000	15.87 MB	16.3 MB
100000	300000	31.87 MB	31.55 MB
500000	1500000	153.37 MB	148.45 MB
1000000	3000000	305.47 MB	291.26 MB

In this case, the size of seven databases is listed in Table 6.1. The size of the databases is measured in MB, and seven databases store different size social graphs. Remarkably, there are three edges for each node, since normally, in some social networks, the scale is between 2 to 5, for example, in YouTube, the scale is about 3.5[5], in this research, the scale is set as 3. Since Clearly, when the social network is small, Neo4j needs more space. When the nodes and edges in the graph increase to 100000 and 300000, Neo4j costs 31.55 MB, and is less then MySQL, which costs 31.87 MB. With more nodes and edges stored in the database, Neo4j uses much less space, so it should has better performance when storing the large-scale social graph.

Remarkably, the size of MySQL database is almost linearly related to the number of nodes and edges storing in the databases. This is because the structure of the relational database is fixed, and when defining the tables, the data type of each attribute is defined as well. Consequently, when inserting data into the tables, the increment of database size should be roughly based on the defined data type and the number of nodes and edges that inserting in the database.

On the other hand, although Neo4j requires more storage when the social network is small when there are more nodes and edges stored in the database, the graph database is beneficial from the flexible database schema. The graph database can easily add nodes and edges into the database without restructuring the database. The high flexibility enables the graph database having a high ability to adapt to the dynamic social graphs. The increment of database size should be mainly related to how much new data is inserted only. Thus, we can say for large-scale social data, graph database, like Neo4j, may cost less storage than the relational database, due to its flexible database structures. Although the gap between the relational database and graph database is relatively small in this case, but based on the analysis above, when the graphs become

huge big, the gap will increase as well, and have serious effects on the performance of the database system.

6.1.2 Execution Time

Clearly, the execution time for querying data from a database is a very basic criterion for evaluating the performance of databases quantitatively. If the database takes shorter time to query the data, it should be considered as more efficient and has better performance. Based on the six kernels mentioned in the above chapter, there are six experiments to process related operations to collect data about the execution time for evaluation. In every experiment, the related queries are processed one hundred times to get the average time for analysis. In addition, in order to simulate the operation environment of the mobile social applications, the related query codes are embedded into the mobile social application. Once the social application is running, the queries are executed as well, and the data is collected for analyzing, and the time is recorded in milliseconds (ms). The longest time and the shortest time are dropped to ensure that the results are not affected by any caching or system process activities. The data used for insertion and searching is generated randomly, and used for both MySQL and Neo4j databases.

Firstly, data insertion is a basic data transaction for a database. Once new data is collected, the data insertion is required for inserting the new data into the database. The time spent for data insertion is listed in Table 6.2. Clearly, when the index of table increases, the inserting time increases as well for MySQL; on the other hand, Neo4j has more stable performance than MySQL, even if the number of nodes and edges are increasing. Remarkably, in this experiment, the time spent for inserting data into the relational database is still short than the graph database, but there is an obvious trend that the graph databases would take shorter than the relational databases. Thus, we may conclude that MySQL performs slightly better than Neo4j at small scale, but when there is more data stored in the database, the inserting time tends to shift in favor of Neo4j.

Table 6.2: Data Insertion in ms

Nodes Number	Edges Number	MySQL	Neo4j
1000	3000	0.2	29.2
5000	15000	0.3	30.3
10000	30000	0.8	32.5
50000	150000	2.6	34.2
100000	300000	5.6	37.3
500000	1500000	20.3	34.5
1000000	3000000	36.1	36.2

In the second experiment, the comparison works on assessing the database performance for searching data from RDBs and GDBs. Data searching is a typical data manipulation operation, and is commonly used in social applications. The experiment results are shown in Table 6.3. Similarly, data searching is much faster in MySQL when the database is small, but if the database size increases, MySQL would take more time for

searching. Thus, the graph database Neo4j, which has more constant performance in data searching, works massively better in bigger graphs than the relational database, and should be considered as the more efficient database system when dealing with the large-scale social graphs.

Table 6.3: Data Searching in ms

Nodes Number	Edges Number	MySQL	Neo4j
1000	3000	0.2	32.2
5000	15000	5.4	31.1
10000	30000	10.8	33.4
50000	150000	23.6	32.5
100000	300000	76.8	35.2
500000	1500000	255.7	36.1
1000000	3000000	316.1	36.8

In addition, the time spent for graph traversal is measured in the third experiment. Basically, applying recursive searching to process graph traversal to collect social data should be very useful in a social application. The experimental results are shown in Figure 6.1. Clearly, the graph database, Neo4j in this case, has much better performance than the relational database (MySQL) for graph traversal for the given social graph. Since the relational database requires sub-queries for processing BFS and when the BFS going deeper, the more sub-queries is required for graph traversal, the query should more complex. In addition, due to the relational model, the relational database has to execute several sub-queries in the query, which is equivalent to execute exponentially more queries than executing a sign query in a graph datbase, so the execution time is much longer than the graph database. Therefore, given a traversal of an artificial graph, the graph database can be viewed as more optimal than the relational database MySQL.

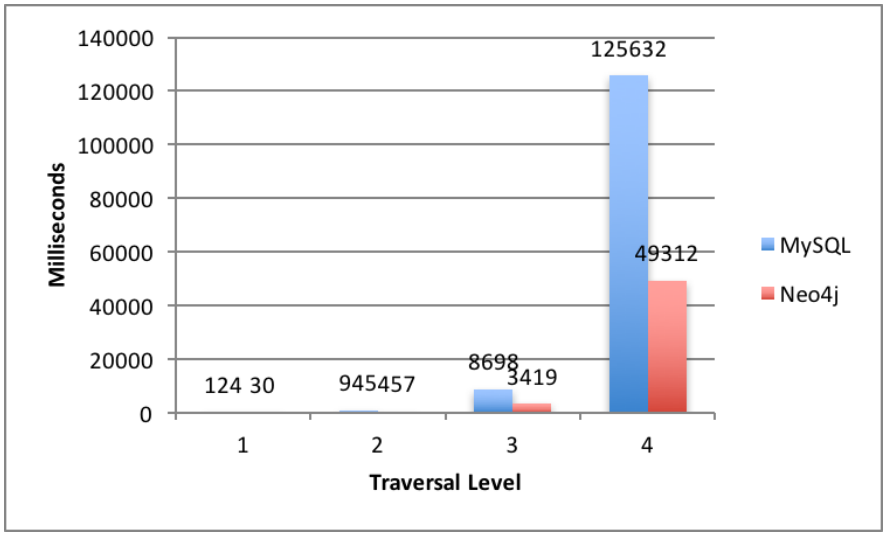

Figure 6.1: Graph Traversal Time in ms

Furthermore, both the graph database and the relational database provides built-in functions for querying data. In this experiment, four common built-in functions are tested, including Max(), Min(), lower() and upper(), and a large-scale social graph containing 1000000 nodes and 3000000 edges is stored in both a relational database a the graph database for the experiments. As Figure 6.2 shown, the relational database and graph database have similar performance for using built-in functions. Both the relational database and the graph database take about 200 to 300 ms to operate the built-in functions for the data.

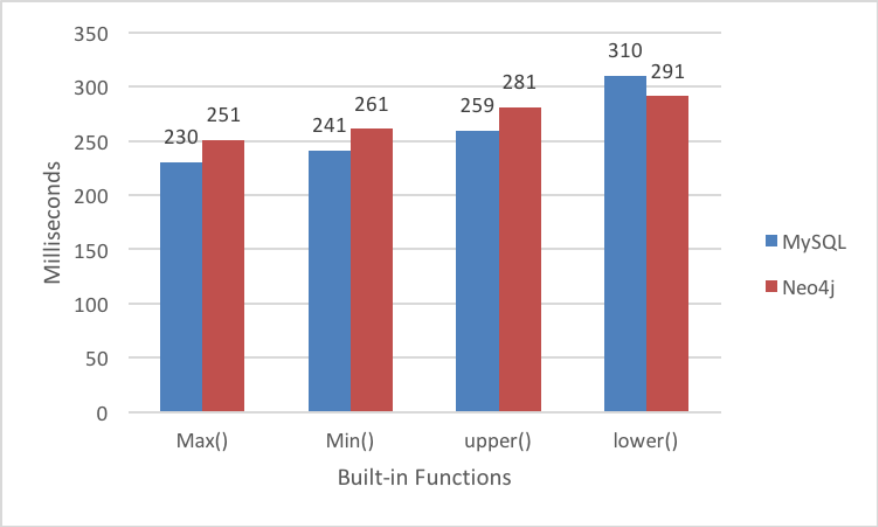

Figure 6.2: Built-in Function in ms

Finally, Experiment Five and Experiment Six test the performance of the union and intersection operations in the relational database and the graph database. These two operations are very common when processing the graph-like data. In a social network, the union and intersection operations can generate lots of social information for users. The two experiments are using a large-scale social graph that contains 1000000 nodes and 3000000 edges as the target, and the experimental results are shown in Figure 6.3. Clearly, in the experiments, the graph database Neo4j has much better performance than the relational database MySQL. Neo4j only takes 680 ms and 710 ms for processing a union operation and an intersection operation, but MySQL requires 1328 ms and 1432 ms for same operations. In both experiments, MySQL almost takes double time more than the Neo4j, this may be because in MySQL, a Union or Intersection operation needs two sub-queries to get the required data for processing, but in Neo4j, a single query can complete the operations.

The experimental results show that MySQL has better performance for data insertion and data searching when the database size is small, but once the database size increases, Neo4j becomes more efficient. In addition, when considering the operation related to graphs, graph traversal, the graph database is more useful than the relational database, especially when traversing the graph with a high degree. Meanwhile, Neo4j and MySQL have the equivalent performance for using built-in functions at an acceptable level. Moreover, when

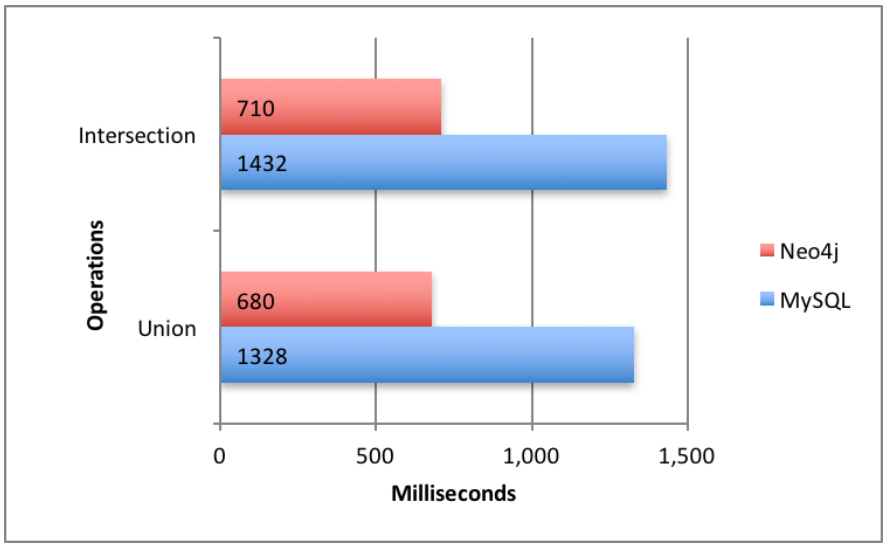

Figure 6.3: Data Union and Intersection in ms

comparing the performance of union and intersection operation, Neo4j has slightly better performance than MySQL. Thus, the graph database, Neo4j, should be considered as the more suitable database system for storing and processing large-scale social graph, according to the query performance.

6.2 Qualitative Analysis

In this section, five criteria, including maturity and level of support, ease of programming, flexibility, security, and data visualization, are applied to the comparison to evaluate the performance of the relational database systems and the graph database systems, in this case, MySQL and Neo4j. These criteria are significant for developers to choose which type of databases to adopt for implementation, although they may be difficult to quantify. Thus, the qualitative analysis is applied to provide insight about the capability based on the subjective judgments.

These criteria are chosen for a reason. Firstly, maturity and level of support mean how stable the database systems are, and the performance on the maturity and level of support shows the reliability of the databases. Also, ease of programming shows how easy to use the database system in practice, or saying the coding efficiency. Moreover, flexibility represents the capability of a database system to adjust to the different situation. Furthermore, security is another important issue for data storage because of the increasing value and importance of data.Finally, the performance on data visualization, which should be a common feature in various applications, especially in social applications, is evaluated to show the practicability of the databases.

6.2.1 Maturity/Level of Support

Maturity refers to how well the particular system has been tested. Usually, a system can be viewed as a stable system if it has higher maturity, which means the system has been tested lots of times, because the

more tests the system has processed, the more bugs will have been identified[74]. Also, the maturity of a system is usually proportional to the level of support. A mature system can be used by more users and in more fields, and thus, it would be more tested to ensure stability.

Firstly, as the traditional data storage approach, the relational database systems have been widely used in several fields for decades. Thus, it can be viewed as a more stable system with higher maturity. The relational model, which organizes data in terms of tables of columns and rows with a unique key to identify the rows, was proposed by E. F. Codd in 1970[19], and various relation database management system (RDBMS) are used to maintain the relational databases. In addition, the high market share percentage of the relational database systems enforces the financial motivation for the corporations, such as Oracle and Microsoft, to ensure the continued performance of their relational database products.

On the other hand, the graph database systems, currently maintain less market percentage, and have smaller commercial value usually. Thus, it may lack the financial motivation for individuals or organizations to enforce the performance. In addition, although the concept of graph databases may appear earlier than the relational databases, the commercial transactional graph databases, for instance, Neo4j, became available in the late 2000s[24]. Also, because the relational database systems have a unified language, SQL, to query data, the support for one relational database implementation can be applied to others, which enriches the supports for the relational databases. Conversely, the most graph database systems developed their own query languages. Therefore, support for one graph database may not work for another one, and this might impact the level of support negatively.

In this case, MySQL and Neo4j are two good examples to show the maturity of the relational database system and the graph database system. Firstly, MySQL has been released for 21 years since May 23, 1995, and the last stable version was released on 11 April, 2016. Clearly, MySQL has processed many versions, and during the version updating, the bugs in the system were fixed to ensure the system stability. Thus, it could be viewed as a database system with high stability and maturity. Meanwhile, in the last 21 years, MySQL and its parent company, Sun, are continuing to provide the extensive support for the commercial products.

On the other hand, Neo4j initially released in 2007, and the latest stable version was released on May 6, 2016. Currently, its commercial venture is in rapidly growing, but is still relatively small. Although the Neo4j website: www.neo4j.org offers a reasonable amount of support for users, there is limited support from outside of the Neo4j site. Thus, if the parent company of Neo4j collapses, the majority of support for Neo4j would be a serious problem. However, remarkably, when considering the graph issues, Neo4j has rich support for users to dealing with graphs. For example, it provides a specific guide for the user to convey the content of a graph to achieve the graph visualization, which is useful in a social application. In addition, there are three query languages supported by Neo4j, including SPARQL, Gremlin and Cypher Query, and this makes the support for one implementation will not work for another one in Neo4j.

Therefore, it may conclude that as the time goes by, the graph databases, Neo4j in this case, are growing and maturing, and it may be as mature as the relational database systems, for example, MySQL, in the

future. However, currently, the relational databases are more stable database systems; on the other hand, the graph database systems provide more support for graph-like data storage and processing.

6.2.2 Ease of Programming

When applying a database system into an application during the application development process, ease of programming should influence the efficiency of programming for developers. Developers can work more efficiently with the database system that can be programmed in an easy way. Thus, ease of programming should be another subjective criterion for evaluating the database systems. In this case, the application programming interface (API), the query language and the performance on data traversal of relational databases and graph databases are compared to evaluate the ease of programming. API is a set of routines, protocols and tools for building a program, which is an important component of implementation. Also, when applying the database system in an application, it needs to use a query language to query data, so the query language should be considered as well. Moreover, graph traversal, which is a process to visit each vertex in a graph, is a common operation when dealing with the graph-like data, and how easy to achieve graph traversal can be a good example of the ease of programming.

Generally, since a relational database uses Structured Query Language (SQL), which is designed for managing data held in the relational database, as the default query language, the common language can make the transitioning between implementations becoming easier than the graph database system. For example, a company may need to change their relational database management system from Oracle to MySQL, and because they both use SQL as the query language, there is no need to rewrite queries for the new database management system. Clearly, it reduces the work for developers.

On the other hand, graph databases are query language-specific, and different graph database systems have different APIs and query languages to query the relevant data for the factual questions from the database. Table 6.4 lists ten popular graph database systems with their own APIs and query languages. For example, AllegroGraph, an RDF and graph database, applies SPARQL, which is a query language for RDF graphs specifically, to query data, and provides a Java Client API. OrientDB, which is open source NoSQL database system supporting multi-model, including graph database, supports SQL queries with extensions to handle the graphs of connected documents via the Java API. DEX is another graph database management system, and it has its own syntax (DEX query) to query data and own API (DEX API). Moreover, ArangoDB, another distributed open-source database with a flexible data model for documents, graphs, and key-values, provides an SQL-like query language called ArangoDB query language via the Java RESTful HTTP API. Clearly, since the graph database systems have different structures and standards, they use different query languages. Thus, developers need to be familiar with the specific languages for developing when applying a graph database management system, and it definitely will increase the difficulty of programming for developers. Consequently, it may make applying a graph database more difficult than applying a relational database.

Table 6.4: Graph Databases with Query Languages and API

Graph Database	Query Language	API
Neo4j	Cypher, Gremlin, SPASQL	Native Java API
AllegroGraph	SPARQL	Java API
Oracle Spatial and Graph	SPARQL	Java API
ArangoDB	ArangoDB query language	Java RESTful HTTP API
VertexDB	JSON	AJAX API
InfiniteGraph	Gremlin	Java API
OrientDB	Own SQL-like Query Language, Gremlin	Java API
Bitsy	Gremlin, Pixy	Blueprints API
DEX	DEX query	DEX API
Titan	Gremlin, SPARQL	Titan Graph API

Actually, the ease of programming depends on the tasks. In this case, Neo4j and MySQL are chosen to evaluate. Neo4j applies Cypher, which is an SQL-inspired language for describing patterns in graphs visually, to select, insert, update or delete from the graph databases. The general structure to query nodes is like:

Listing 6.1: Query in Neo4j

```
MATCH (node:Label)
RETURN node.property
```

And similarly, the general syntax to query relationships is like:

Listing 6.2: Query in Neo4j

```
MATCH (n1:Label1)-[rel:TYPE]->(n2:Label2)
WHERE rel.property > {value}
RETURN rel.property, type(rel)
```

On the other hand, MySQL uses SQL, which is a common query language for relational databases. There is no difference between nodes and relationships stored in relational databases. To query an attribute from a table in a relational database by using SQL is like:

Listing 6.3: Query in Neo4j

```
SELECT * FROM Table
WHERE (condition)
```

Basically, SQL offers a simpler query language for querying data from the databases, and provides more functions and operations for querying data than Cypher. However, when querying data from a social graph, Cypher is more suitable to query the target data, which can be the nodes, relationships and properties. In this case, the performance on graph traversal is compared for evaluation.

Normally, it is very difficult to traverse a graph to query the data when using relational databases, since it is hard to represent the graph structure, and it needs to use some complex sub-queries to traverse the

graphs. On the other hand, a graph database can easily process graph traversals due to the features of the graph databases. Because the data stored in a graph database is stored in terms of a graph, users can take advantage of the graph structure in a graph traversal. Specifically, in a social recommendation system, which is very common in a social application for users to find some new friends, the friend-of-a-friend query is a particular query that needs to traverse the graph to query the relationships with two or more degrees. For example, A is friends with B and C. B is friends with D and E. D is friends with F and H. C is friends with G. The distance of the relationships can be:

A = 0 (the origin)

B = 1

C = 1

D = 2

E = 2

G = 2

F = 3

H = 3

When querying the data of H, who is three degrees away from A, if applying the relational database, for example, MySQL, the SQL query needs to use a sub-query to traverse one degree deeper in a graph. In this case, it requires three sub-queries to query the data of H, and it is like:

Listing 6.4: Querying data with degree 3 in MySQL

```
SELECT node
FROM graph
Where connectedTo IN (
SELECT node
FROM graph
Where connectedTo IN (
SELECT node
FROM graph
Where connectedTo IN (
SELECT node
FROM graph
WHERE connectedTo = ''A'')
)
)
```

When the graph is larger and more complex, it needs more sub-queries to define the condition, and make the query more difficult. However, it is easy to traverse graph to query the friend of a friend relationship

from a graph database, in this case, Neo4j. The query is like:

Listing 6.5: Querying data with degree 3 in Neo4j

```
// Query friends and friends of friends, order by depth of the relationship

start n=node(*), person=node({userNode})
MATCH p = (person)-[:FRIEND*1..3]-(friend)
return distinct p order by length(p);
```

Clearly, the graph database, in this case Neo4j, offers a simple query to achieve graph traversal to query the nodes with high degrees, but it will be more complex in a relational database. MySQL requires complex sub-queries in the query, which will increase the difficulty of programming. Thus, when handling large-social social graph data, developing with the graph databases can enable developers simplifying programming than applying the relational databases.

In conclusion, we can say applying the relational database can take advantage of the unified query language and API to simplify programming, but when dealing with the graph-related operations, the graph databases can enable developers programming more efficient.

6.2.3 Flexibility

Although the relational database system can be viewed as more mature and secure when compared with the graph database, its flexibility may be less than the graph database. Since its database schema is fixed, it is hard to represent semi-structure or unstructured data, and may be less suitable for storing the social graph that may be extending to be a larger graph dynamically. For example, suppose that in a social network, originally, there are two people A and B. They have their social attributes, such as name, gender, age and address, and they are good friends; also, they post five personal statements: A posts statement a and b, and B posts statement c, d and e. To store such information needs three tables: one for storing people information, one for storing the relationship information, and the third one for storing statement information. When using the relational database, for example, MySQL, the tables should be like:

Table 6.5: People Table

ID	Name	Gender	Age	Address
1	A	Male	27	Address A
2	B	Male	28	Address B

Table 6.6: Relationship Table

ID	PeopleA	PeopleB	Type
1	A	B	Friend

Table 6.7: Statement Table

ID	Who	When	What
1	A	Time 1	Statement a
2	A	Time 2	Statement b
3	B	Time 3	Statement c

For the graph database, Neo4j in this case, the information is stored in the nodes or edges, and the same information should be represented like:

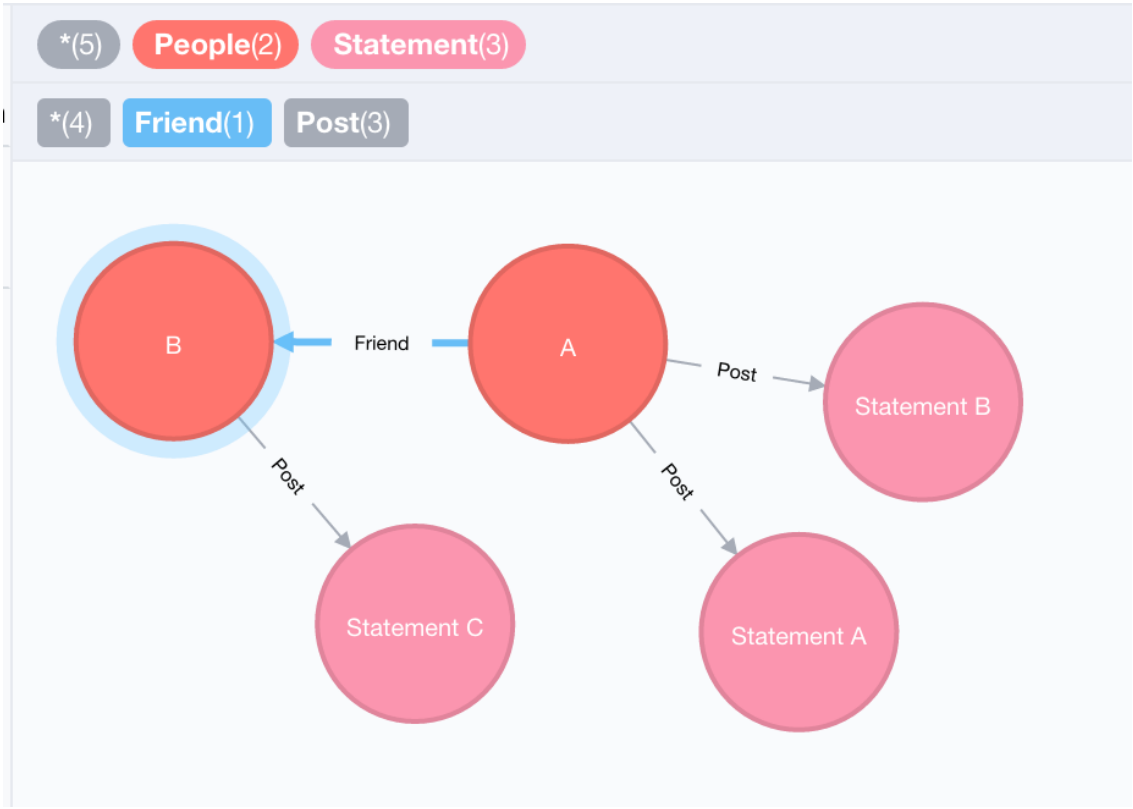

Figure 6.4: Relationship of A and B

Furthermore, the social network can extend over time, for example, a new friend, C, joins to this group, and he has a cat, named Sweet. Since the relational database schema is fixed, when storing such information, it needs to restructure the entire relational database schema to add a new table to store the cat information.

After restructuring the database scheme, there should be one new table in the database, and remarkably, the relationship between C and Sweet is stored in this table, as in Table 6.8:

Table 6.8: Pet Table

ID	Name	Species	Owner
1	Sweet	Cat	C

On the other hand, when using graph database to store such data, the system just needs to add a few edges and nodes into the graph to represent the new information, without restructuring the original graph. In this case, it only needs to add a node for C and a node for Sweet, and some edges to link them with other nodes in the graph to represent the new relationships:

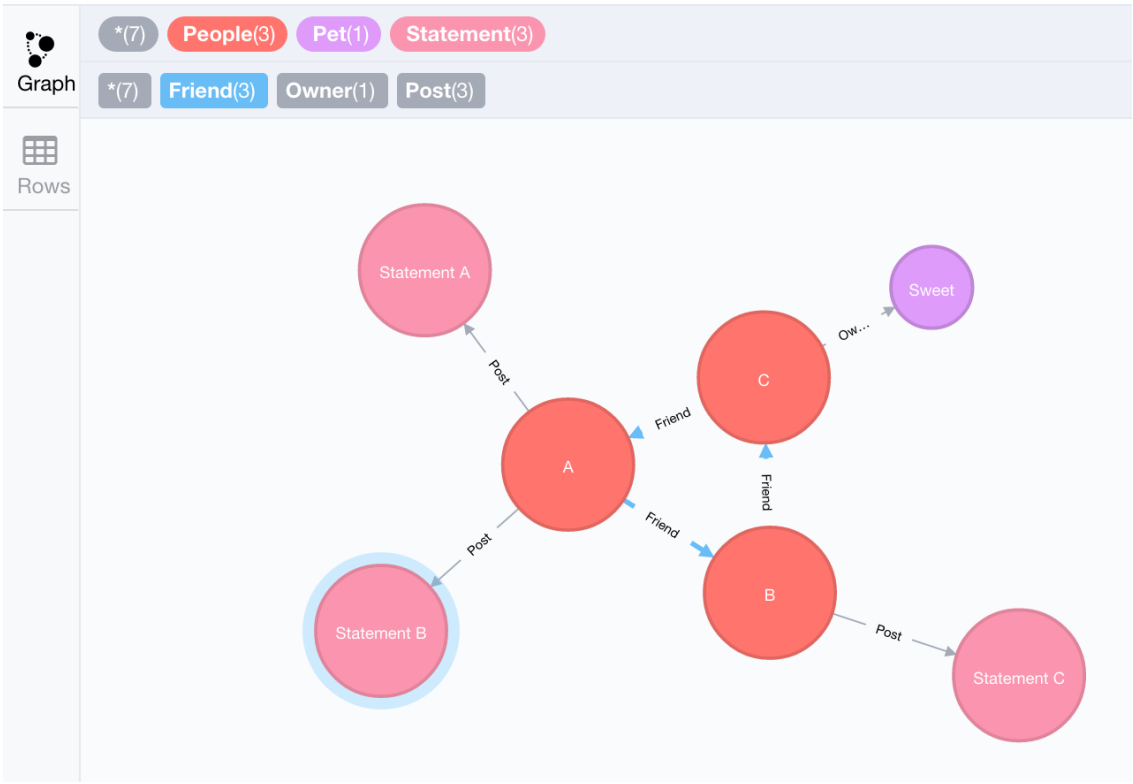

Figure 6.5: Relationship of A , B and C

In this case, database scheme restructuring may be simple, but when more people join the network and data sets become larger, restructuring would be very difficult and costly. Conversely, a graph database does not need to restructure the original database schema, and it can fulfill the requirements for the network exploration by adding the nodes and edges simply. Generally, for the most applications, a fixed database may increase the database performance, because the designed database schema should fulfill the most requirements. However, for the social applications, a flexible database is more suitable, since the social graph

may change dynamically, and it is hard to design a database schema for the dynamical situation. Hence, we may conclude that the graph database has a higher flexibility when compared with the relational database for storing social graph data, which can be suitably stored in a flexible database.

6.2.4 Security

Since data is a valuable kind of resource, it is more necessary to ensure the security of the data. Generally, protecting the sensitive data from unwanted actions of unauthorized users should be important for users, especially for social networks, which store valuable social information for users. For example, in 2011, 77 million Sony PlayStation Network accounts were hacked, and the site was down for a month, and the leaked information was used for credit fraud. Thus, data security is another important measure to evaluate a storage system.

Generally, relational database systems can be considered to be providing more mature support for enforcing data security. Although relational database systems may face several security threats, such as SQL injection, cross-site scripting, rootkits, and weak communication protocols, a relational database system can adopt lots of mature mechanisms to secure the data. However, although the graph database system provides a solution to solve problem of big data storing with the high database performance, it needs to pay more attention to the security issue.

In this case, Table 6.9 shows the comparison between MySQL and Neo4j in some security services, including authentication, data integrity, confidentiality and database logs. Basically, authentication is the process of identifying an individual to control the access. Data integrity refers to maintaining and assuring the accuracy and consistency of data over the entire life-cycle. Confidentiality is about whether the private information that should not be publically available will be leaked or not. Quality logs are a key element for monitoring for malicious activity and analyzing discovered irregularities.

As a typical relational database system, MySQL has good support for data security. Based on Table 6.9, it may conclude the relational database has good performance in authentication, data integrity, confidentiality and database logs to enforce the data security. On the other hand, although Neo4j can ensure the data integrity and has completed log files, its authentication may have problems in some cases, and it has no method to ensure the confidentiality, which may be very important in the social network. Therefore, we may say, relational database systems have a mature mechanism in data security, and although graph database is in rapid development, it still needs to enforce the data security.

6.2.5 Data Visualization

This research is focusing on the social data, and data visualization can communicate the social information clearly and efficiently via information graphics, which should be very useful for a social network. Thus, the capability to achieve data visualization is compared to show the usability of the relational databases and the graph databases in the practice.

Table 6.9: Security Services Comparison between MySQL and Neo4j

Security Services	MySQL	Neo4j
Authentication	MySQL applies Old Password Authentication and Secure Password Authentication to authenticate the connections. The user-password scheme prevents access from unwanted or untrusted users.	In Neo4j, SSL protocol is used to enable clients to authenticate with the database, but there is no method to secure the operating environment to ensure the server authentication.
Data Integrity	As a relation database system, MySQL uses ACID properties to ensure the database transactions are processed reliably, and then the data integrate is ensured.	Neo4j is a transactional database so that operations can be performed within transaction boundaries only. Also, Neo4j fully support ACID properties to maintain data integrity and ensure reliable transactional behavior.
Confidentiality	MySQL offers a number of cryptographic and hashing functions, such as AES_ENCRYPT(), AES_-DESCRYPT(), DES_ENCRYPT(), DES_DECRYPT(), and SHA(), to encrypt or decrypt data directly in queries. Therefore, confidential data can be stored or returned in a secure way, and data confidentiality is ensured.	In Neo4j, data is stored clearly so that data confidentiality may be not achieved.
Database Logs	MySQL offers two types of logs: the general log and error log. The general log contains connection attempts with the basic details, queries and other miscellaneous operations performed by clients. Also, the error log contains the problem notifications. In MySQL, the most activities can be logged for monitoring.	There are five log files in Neo4j, including neo4j.log storing general information about Neo4j, debug.log storing useful information for debugging problems with Neo4j, http.log storing request log for the HTTP API, gc.log storing the Garbage Collection logging provided by the JVM, and query.log storing information about executed quires. There log files can record most information about Neo4j.

Although a textual display of data may be a lot clearer in some cases, data visualization can enable users to view data directly and have a clear view of the data. Users can communicate information clearly and efficiently via the graphs, especially for data of the social network, in which data can be represented as graphs. For example, The graph in Figure 6.6 represents the data stored in Table 6.10, 6.11, 6.12 and 6.13. Once the social network is visually represented, users can easily know the relationships, which should be a very convenient way for users have a clear view about the social network. Thus, when handling the social graph data, visualizing data should be better than displaying the data textually.

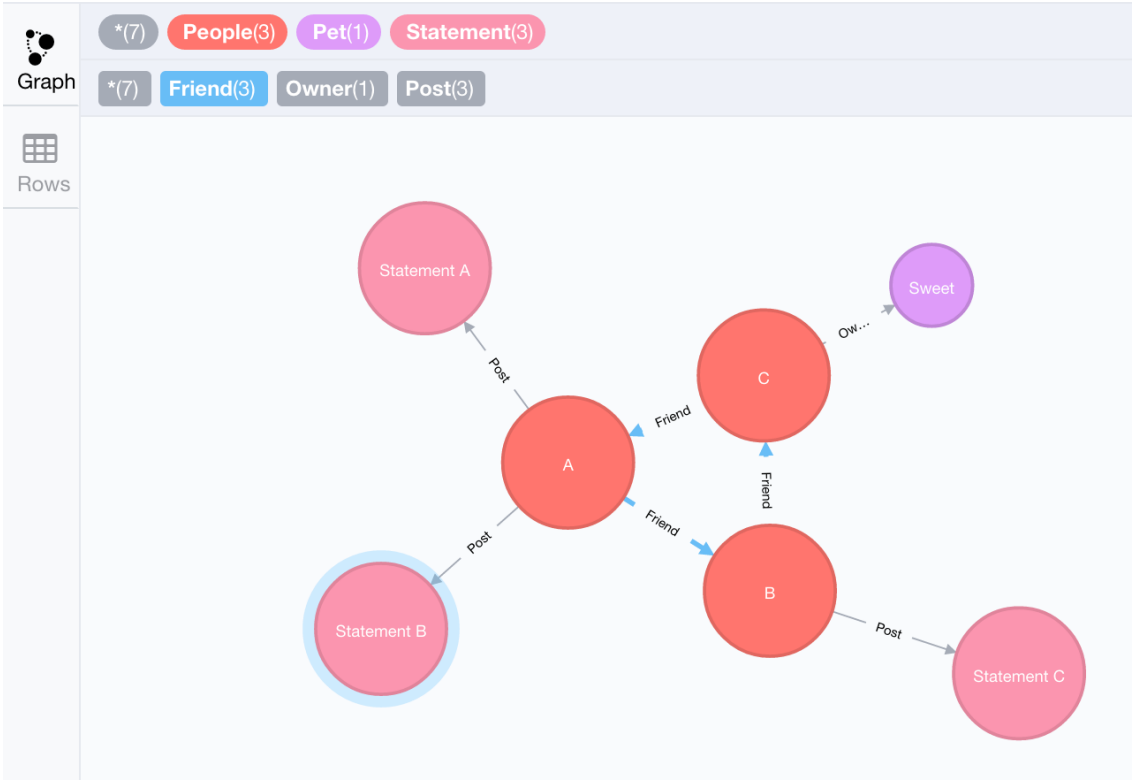

Figure 6.6: Data visualization example

Table 6.10: People Table

ID	Name	Gender	Age	Address
1	A	Male	27	Address A
2	B	Male	28	Address B

Table 6.11: Relationship Table

ID	PeopleA	PeopleB	Type
1	A	B	Friend

Table 6.12: Statement Table

ID	Who	When	What
1	A	Time 1	Statement a
2	A	Time 2	Statement b
3	B	Time 3	Statement c

Table 6.13: Pet Table

ID	Name	Species	Owner
1	Sweet	Cat	C

The most graph database systems provide support for data visualization. Since in the graph database, data is stored in term of the graph, the database system can easily convey the content of graphs. In this case, Neo4j offers a customizable data visualization tool based on the built-in D3.js library, called screencast. It can demonstrate how to style nodes and relationships in the Neo4j Brower visualization, and set the colors, sizes and titles for the data in the graph.

However, the relational database has little support for data visualization. Because the relational database structure, which is representing data in term of tables, is fixed, it is hard to translate the structure of the relational database into other forms, like graph or plots, for visualization. Thus, we may conclude that graph database has more ability to provide support to visualize data for database users than the relational database.

6.3 Summary

In this chapter, two kinds of analysis, the quantitative analysis and the qualitative analysis, are used to evaluate the performance of the graph database, Neo4j, and the relational database, MySQL, when handling large-scale social graph-like data. Firstly, Neo4j should be viewed as a more useful database system based on the quantitative analysis. Basically, due to the flexible database schema, a graph database can take advantage to reduce the storage cost for storing large-scale social graphs, although it may require more space when the size of the social graph is small. In addition, I performed six experiments about execution time, which were very informative for evaluating the performance of the databases, based on the mentioned six kernels. According to the experimental results, we can say that normally, when the size of the database increases to a large-scale, a relational database takes more time to insert or search data than a graph database. Also, a graph database has absolute better performance in using built-in functions and union and intersection operations. Moreover, when traversing the graph, if the depth of traversing increases, a graph database spends much shorter time to query the data. Thus, generally, a graph database has better performance

objectively when handling large-scale social graph data.

On the other hand, the qualitative analysis shows the graph database is better in some areas than the relational database, although the relational database is better in maturity, ease of programming and security. Basically, a relational database should be considered as a more mature way to store data with more support than a graph database. Also, since it applies SQL as the uniform query language, the relational database makes it friendlier to the developers, although a graph database may be a better choice for handling graph-related applications. In addition, a relational database is securer than a graph database to ensure the data security for users. On the other hand, since the database schema of the relational database is fixed, the graph database comparatively has more flexibility for storing unstructured or semi-structured data, especially the social graph data. Finally, a graph database can help users to handle data visualization to get better performance, whereas a relational database lacks such support.

Chapter 7

Conclusion and Future Work

All in all, after comparing the performance of the graph database and the relational database when handling large-scale social data, we can answer the key research problems defined in the second chapter. Firstly, unlike the relational database that has fixed database schema, which causes the database size to have a linear relationship with the number of nodes and edges stored in the tables, a graph database, which is more flexible, can reduce the storage cost to increase the performance, especially for large-scale social graph data. Also, according to the experimental results, a graph database, basically, has better performance in execution time for querying data from the database. Thus, a graph database should be considered having better performance objectively in storing and process large-scale social data.

Secondly, based on the performance on maturity and level of support and security, the relational database system should be viewed as the highly reliable database system, compared with the graph database system. First, the relational databases are more mature than the graph databases. Because of the longer development history and more released versions, the relational databases can provide more support for users, although the graph databases are rapidly developing. Also, a relational database is a securer storage approach than a graph database. A relational database provides more support to ensure data security, whereas a graph database lacks such support; therefore, the users may be more confident with the relational database in data security.

Thirdly, a graph database has advantage in flexibility and data visualization, so it is a storage approach with high practicability compared with a relational database, although a relational database may be easy to program in some cases. First, it may be easier for developers to use a relational database than a graph database, although a graph database may be more useful to handle graph-related operation, such as graph traversal. The relational databases apply a uniform query language, SQL, but each graph database may have its own query language and API, which makes programming more difficult. Second, since the database schema of a relational database is fixed, it may be unsuitable to store unstructured or semi-structured data; on the other hand, a graph database offers more flexibility to handle such data, especially the graph data. Thus, the graph database can be more useful to handle the large-scale social graph data. Third, a graph database can enable data visualization to provide a clearer way to view the data, but a relational database lacks such support on it. Thus, a graph database can offer more features when applying the data visualization in a social application.

Therefore, in conclusion, a graph database could be considered as a better choice for handling social graph data, especially when the dataset size is very large. Developers can take advantage of using the graph databases in several areas, such as storage cost, query time, flexibility, and data visualization, although the relational database is still a good storage approach and has its own advantage in ease of programming and data security. A graph database enables users to handle large-scale social graphs in an easier way and provides more useful functions to use the graph data.

7.1 Future Work

7.1.1 Database Benchmark

Although there are several database benchmarks, there is no benchmark for evaluating the performance of databases to store and process large-scale social graph data. This work designs several queries to evaluate the capabilities on processing graph data, and these queries can be used to measure the performance of the databases. As social applications, such as Facebook, Twitter, and LinkedIn are becoming more common in daily social lives, there is an increasing need to handle large-scale social graph data, so a reliable, repeatable benchmark test suite is needed for evaluating the databases. The designed queries in this research can be implemented as a benchmark to test the main capabilities of RDBs and GDBs. However, the work on this is still incomplete, and for future work, I will implement more queries to complete this benchmark to offer an open source database benchmark for evaluating the performance of databases on storing and processing large-scale social graph data.

7.1.2 Comparing More Database Systems

This thesis research measures and compares the performance of relational databases (RDBs) and graph databases (GDBs) on storing and processing large-scale social graph data, and I choose MySQL as the typical RDB and Neo4j as the typical GDB. Currently, with the development of the graph database technology, there are several graph database systems, such as AllegroGraph, DEX, InfoGrid, and FlookDB, and unlike relational database systems, which have the similar basic structure, many graph databases have their own structures and different standards, so Neo4j may not be representing all graph databases. Therefore, in future work, more graph database platforms should be compared to offer a more complete view of the capabilities of the graph database on handling large-scale social graph data.

7.1.3 Storing and Processing Practical Social Graph Data

In order to store and process large-scale graph data, this work applies the R-MAT algorithm to generate the social graphs for evaluation. Although the generated graphs may be large enough and meet the properties for a scale-free network, they are very simple social graphs with only a few of the attributes or properties of the

social actors (nodes). In practical applications, the social graphs would be more complex with more data types and properties. Meanwhile, lots of social network companies such as Facebook and Twitter provide several datasets for research purposes. The next step of my research may incorporate such datasets into evaluations. But, the existing graph datasets are normally only for GDBs, rather than RDBs. How to transform such data into relational database form for evaluation may be an issue in future studies. As analyzed in Chapter 6, the graph databases have a highly flexible database schema, so transforming a relational database to a graph database needs to organize the unstructured or semi-structured data structurally. A designed transformation for a particular database may work, but a general solution for transforming the database is hard to develop. By doing such a transformation, we can have the same, real world and thus realistic, databases to compare. This can improve the validity of the evaluation, and benefit the other related work.

References

[1] Data modeling - database normalization. `http://gerardnico.com/wiki/data_modeling/normalization/`. Online; accessed 19-June-2016].

[2] Sematic web. `https://www.w3.org/standards/semanticweb/`. [Online; accessed 19-June-2016].

[3] Graph theory and terminology. `https://leanjavaengineering.wordpress.com/2013/08/28/graph-gremlins/`, 2013. [Online; accessed 19-June-2016].

[4] Graph databases. `https://en.wikipedia.org/wiki/Graph_database`, 2016. [Online; accessed 19-June-2016].

[5] Nadeem Akhtar, Hira Javed, and Geetanjali Sengar. Analysis of facebook social network. In *Computational Intelligence and Communication Networks (CICN), 2013 5th International Conference on*, pages 451–454. IEEE, 2013.

[6] Sarah Allen, Vidal Graupera, and Lee Lundrigan. *Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile and Android Development and Distribution*. Apress, 2010.

[7] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1–39, 2008.

[8] Grigoris Antoniou. *A semantic web primer*. the MIT Press, 2004.

[9] Timothy G Armstrong, Vamsi Ponnekanti, Dhruba Borthakur, and Mark Callaghan. Linkbench: a database benchmark based on the facebook social graph. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 1185–1196. ACM, 2013.

[10] Sumita Barahmand and Shahram Ghandeharizadeh. Bg: A benchmark to evaluate interactive social networking actions. In *CIDR*, pages 1–28. Citeseer, 2013.

[11] Carlo Beltrame. Key-value stores. *Algorithms for Database Systems*, pages 1–12, 2013.

[12] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific American*, 284(5):28–37, 2001.

[13] Peter J Carrington, John Scott, and Stanley Wasserman. *Models and Methods in Social Network Analysis*. Cambridge University Press, 2005.

[14] Dorwin Cartwright and Frank Harary. Structural balance: a generalization of heider's theory. *Psychological review*, 63(5):277, 1956.

[15] Rick Cattell. Scalable sql and nosql data stores. *Acm Sigmod Record*, 39(4):12–27, 2011.

[16] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-mat: A recursive model for graph mining. In *SDM*, volume 4, pages 442–446. SIAM, 2004.

[17] Dan Chalmers and Morris Sloman. A survey of quality of service in mobile computing environments. *Communications Surveys & Tutorials, IEEE*, 2(2):2–10, 1999.

[18] Andre Charland and Brian Leroux. Mobile application development: web vs. native. *Communications of the ACM*, 54(5):49–53, 2011.

[19] Edgar F Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[20] Edgar F Codd. Further normalization of the data base relational model. *Data Base Systems*, pages 33–64, 1972.

[21] Edgar F Codd. *The Relational Model for Database Management: Version 2*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[22] David Coupier. Workshop on random graphs. `http://math.univ-lille1.fr/~tran/journeesgraphesaleatoires.html`, 2011. [Online; accessed 19-June-2016].

[23] Bipin C Desai. *An Introduction to Database Systems*. West Publishing Co., 1990.

[24] Neo4J Developers. Neo4j. *Graph NoSQL Database [online]*, 2012.

[25] Boonsri Dickinson. Social media marketing. `http://www.businessinsider.com/explainer-what-exactly-is-the-social-graph-2012-3`. Online; accessed 19-June-2016].

[26] David Dominguez-Sal, P Urbón-Bayes, Aleix Giménez-Vanó, Sergio Gómez-Villamor, Norbert Martínez-Bazan, and Josep-Lluis Larriba-Pey. Survey of graph database performance on the hpc scalable graph analysis benchmark. In *International Conference on Web-Age Information Management*, pages 37–48. Springer, 2010.

[27] Ramez Elmasri. *Fundamentals of Database Systems*. Pearson Education India, 2008.

[28] Brian Fling. *Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps*. O'Reilly Media, Inc., 2009.

[29] George H Forman and John Zahorjan. The challenges of mobile computing. *Computer*, 27(4):38–47, 1994.

[30] Linton C Freeman. *The Development of Social Network Analysis*. Empirical Press Vancouver, 2004.

[31] Mark Granovetter. The strength of weak ties: A network theory revisited. *Sociological theory*, 1(1):201–233, 1983.

[32] Mark S Granovetter. The strength of weak ties. *American Journal of Sociology*, pages 1360–1380, 1973.

[33] Mark Graves, Ellen R Bergeman, and Charles B Lawrence. Graph database systems. *IEEE Engineering in Medicine and Biology Magazine*, 14(6):737–745, 1995.

[34] IT Policy Compliance Group. Managing the benefits and risks of mobile computing. Technical report, New York, Tech. rep. 2011.[Online]. http://www.isaca.org/Knowledge-Center/Documents/Managing-the-Benefits-and-Risks-of-Mobile-Computing-ITPCG-Dec2011.pdf, 2011.

[35] John Guare. *Six Degrees of Separation*. Dramatists Play Service, 1992.

[36] Theo Haerder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Computing Surveys (CSUR)*, 15(4):287–317, 1983.

[37] Allan Hammershoj, Antonio Sapuppo, and Reza Tadayoni. Challenges for mobile application development. In *Intelligence in Next Generation Networks (ICIN), 2010 14th International Conference on*, pages 1–8. IEEE, 2010.

[38] Jing Han, E Haihong, Guan Le, and Jian Du. Survey on nosql database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, pages 363–366. IEEE, 2011.

[39] Robert A Hanneman and Mark Riddle. Social network analysis. *Riverside: University of California*, 2001.

[40] Robert A Hanneman and Mark Riddle. Introduction to social network methods. 2005.

[41] Frank Harary. Structural models: An introduction to the theory of directed graphs. 2005.

[42] Robin Hecht and Stefan Jablonski. Nosql evaluation. In *International Conference on Cloud and Service Computing*, pages 336–41. IEEE, 2011.

[43] Florian Holzschuher and René Peinl. Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 195–204. ACM, 2013.

[44] Tomasz Imielinski and Henry F Korth. *Mobile Computing*, volume 353. Springer, 1996.

[45] Borislav Iordanov. Hypergraphdb: a generalized graph database. In *International Conference on Web-Age Information Management*, pages 25–36. Springer, 2010.

[46] Hoe Jin Jeong and Sang Ho Lee. An integrated database benchmark suite. In *2005 First International Conference on Semantics, Knowledge and Grid*, pages 60–60. IEEE, 2005.

[47] Hans K Klein and Daniel Lee Kleinman. The social construction of technology: Structural considerations. *Science, Technology & Human Values*, 27(1):28–52, 2002.

[48] Graham Klyne, Jeremy J Carroll, and Brian McBride. Resource description framework (rdf): Concepts and abstract syntax. *W3C Recommendation*, 10, 2004.

[49] Jeff Korhan. What your business needs to know about social graphs. `http://www.socialmediaexaminer.com/what-your-business-needs-to-know-about-social-graphs/`, 2011. [Online; accessed 19-June-2016].

[50] David Krackhardt. The strength of strong ties: The importance of philos in organizations. *Networks and Organizations: Structure, Form, and Action*, 216:239, 1992.

[51] Valentino Lee, Heather Schneider, and Robbie Schell. *Mobile Applications: Architecture, Design, and Development*. Prentice Hall PTR, 2004.

[52] AR Mashaghi, Abolfazl Ramezanpour, and V Karimipour. Investigation of a protein complex network. *The European Physical Journal B-Condensed Matter and Complex Systems*, 41(1):113–121, 2004.

[53] Dan McCreary and Ann Kelly. Making sense of nosql. *Shelter Island: Manning*, pages 19–20, 2014.

[54] Stanley Milgram. The small world problem. *Psychology Today*, 2(1):60–67, 1967.

[55] Vision Mobile. Cross-platform developer tools 2012. Technical report, London, Tech. rep. 2012.[Online]. http://www. visionmobile. com/product/cross-platformdeveloper-tools-2012, 2012.

[56] ABM Moniruzzaman and Syed Akhter Hossain. Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. *arXiv preprint arXiv:1307.0191*, 2013.

[57] Reed E Nelson. The strength of strong ties: Social networks and intergroup conflict in organizations. *Academy of Management Journal*, 32(2):377–401, 1989.

[58] Nicholas Palmer, Roelof Kemp, Thilo Kielmann, and Henri Bal. Ibis for mobility: solving challenges of mobile computing using grid techniques. In *Proceedings of the 10th Workshop on Mobile Computing Systems and Applications*, page 17. ACM, 2009.

[59] Jeff Z Pan. Resource description framework. In *Handbook on Ontologies*, pages 71–90. Springer, 2009.

[60] Jan Paredaens, Paul De Bra, Marc Gyssens, and Dirk Van Gucht. *The structure of the relational database model*, volume 17. Springer Science & Business Media, 2012.

[61] Dan RK Ports, Austin T Clements, Irene Zhang, Samuel Madden, and Barbara Liskov. Transactional consistency and automatic management in an application data cache. In *OSDI*, volume 10, pages 1–15, 2010.

[62] Shelley Powers. *Practical RDF*. O'Reilly, 2009.

[63] Anatol Rapoport. Contribution to the theory of random and biased nets. *The Bulletin of Mathematical Biophysics*, 19(4):257–277, 1957.

[64] Andre Ribeiro and Alberto Rodrigues da Silva. Survey on cross-platforms and languages for mobile apps. In *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*, pages 255–260. IEEE, 2012.

[65] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph Databases: New Opportunities for Connected Data*. " O'Reilly Media, Inc.", 2015.

[66] Martin Ruef. Strong ties, weak ties and islands: structural and cultural predictors of organizational innovation. *Industrial and Corporate Change*, 11(3):427–449, 2002.

[67] Bryce Merkl Sasaki and Aspiring Graphista. Graph databases for beginners: Why we need nosql databases. `https://neo4j.com/blog/why-nosql-databases/`, 2015. [Online; accessed 19-June-2016].

[68] John Scott. Social network analysis. *Sociology*, 22(1):109–127, 1988.

[69] John Scott and Peter J Carrington. *The SAGE Handbook of Social Network Analysis*. SAGE publications, 2011.

[70] Seung Kyoon Shin and G Lawrence Sanders. Denormalization strategies for data retrieval from data warehouses. *Decision Support Systems*, 42(1):267–282, 2006.

[71] Inderjeet Singh and Manuel Palmieri. Comparison of cross-platform mobile development tools. In *2012 16th International Conference on Intelligence in Next Generation Networks*, 2012.

[72] IBM Software. Native, web or hybrid mobile-app development. Technical report, New York, Tech. rep. 2013.[Online]. http://public.dhe.ibm.com/common/ssi/ecm/en/wsw14182usen/WSW14182USEN.PDF, 2013.

[73] Antonio Tapiador, Diego Carrera, and Joaquin Salvachua. Social stream, a social network framework. In *Future Generation Communication Technology (FGCT), 2012 International Conference on*, pages 52–57. IEEE, 2012.

[74] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th Annual Southeast Regional Conference*, page 42. ACM, 2010.

[75] Stanley Wasserman. *Social Network Analysis: Methods and Applications*, volume 8. Cambridge university press, 1994.

[76] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *nature*, 393(6684):440–442, 1998.

[77] Douglas Brent West et al. *Introduction to Graph Theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[78] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna PN Puttaswamy, and Ben Y Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 205–218. Acm, 2009.

[79] Paul Zaich. Dev bootcamp day 17: Relational databases and deconstructing the learning process. `http://www.paulzaich.com/2012/07/03/blog/ruby-rails/dev-bootcamp-day-17-relational-databases-deconstructing-learning-process/`, 2012. Online; accessed 19-June-2016].

[80] James Bryan Zimmerman. Mobile computing: Characteristics, business benefits, and the mobile framework. *University of Maryland European Division-Bowie State*, 1999.