Data Relationship Query in Relational DB, NoSQL DB and Graph DB

Kay Thi Yar
University of Computer Studies, Yangon

Khin Mar Lar Tun
University of Computer Studies, Yangon

ABSTRACT
Every nation has vast amount of census data and analysis of these data is the value for nation as source citations, correlating and corroborating sources, relevance or findings contradictions. These census data may relate in any form based on family group records, friendship, co-worker and etc. In this paper, our nation, Myanmar's census data is used as source citations for searching relationship between two distinct persons based on unique National Registration Card (NRC) Number. The Myanmar census data involve person name, date of birth, gender, occupation, parent names, relationship with householder, NRC number and detail parent's family records including jobs and etc. NRC number is the unique identification number for every citizen in Myanmar. The aim of this paper is to observe the efficient data storage form for those related data among three types of database structure; relational DB, NoSQL DB, graph DB. The observation is done by retrieving data relationship from these databases using their query form. In relational database, personnel data is stored as table structure while in NoSQL databases like key-values store, column-family store and document store, personnel data is stored as key-values pair, column oriented and document oriented structure respectively. In graph database, personnel data is stored as graph structure with persons as nodes and relationship between them as edges. Then the query processing time is compared based on retrieving related data from those databases by using their relevant query system and find out which query process can produce the optimal running time. The experimental results show that graph database is more powerful in retrieving relationship over relational and NoSQL databases and it can provide better performance when handling in highly interconnected data compared to relational and NoSQL databases.

Keywords
Relational Database, NoSQL Databases, Graph Database, Data Model, Query Model.

1. INTRODUCTION
In today's world, the great majority of people around the globe are citizens of the information society. In real world, in addition to search personal information of each person, searching connections between them is interesting issue in our society to acquire related information and personnel history. Nowadays, social network like Facebook is the best medium for communication, sharing knowledge and information. For people, to contact with each other and to interchange information between them, searching relationship between persons is necessary and vital matter. Classic relational databases use two-dimensional table for data creation and their Structure Query Language have a limitation when it comes to data aggregation, which is used for business intelligence and data mining [2]. Moreover, multi-table queries are not effective for huge data queries. Data aggregation becomes impossible on very large volumes of data when it comes to memory and time consumption. The problem with relational model is that it has some scalability issues that is performance degrades rapidly as data volumes increases [4]. This led to the development of a new data model like NoSQL and graph databases. Though the concept of NOSQL was developed a long time ago, it was after the introduction of database as a service (DBaaS) that it gained a prominent recognition. Because of the high scalability provided by NOSQL, it was seen as a major competitor to the relational database model. Unlike RDBMS, NoSQL databases are designed to easily scale out and when they grow [3]. Most NoSQL systems have removed the multi-platform support and some extra unnecessary features of RDBMS, making them much more lightweight and efficient than their RDBMS counterparts. The NoSQL data model does not guarantee ACID properties (Atomicity, Consistency, Isolation and Durability) but instead it guarantees BASE properties (Basically Available, Soft state, Eventual consistency) [5]. It is in compliance with the CAP (Consistency, Availability, Partition tolerance) theorem. In recent years, new graph database technology emerged for the compliment of relational and NoSQL databases of storing and effectively retrieving huge volumes of relationship data. Graph database can store complex and dynamic relationships of highly connected data like person information. It can also make easier for developers to work with when navigating connected data [9]. Now, it is used in industries as diverse as healthcare, retail, oil and gas, media, and gaming. With pros and cons of each DB structure, the searching of data relation using each DB's standard query system is explored in this paper. Retrieving data from these databases perform differently. The query processing for data relation search makes the query statement complex and retrieving process lengthy in relational DB compared to graph DB. In this paper, query statements for data relationship searching and processing time are compared. The rest of this paper is organized as follows: Section 2 describes the census data scheme of every citizen in Myanmar. Section 3 express the overview of relational database and also presents its data model, query model and weakness of relationships. Section 4 describes the overview of NoSQL databases and also presents its data model and limitation of relationships. Section 5 describes the overview of graph database and also presents its data model, query model and strength of relationships. Section 6 shows the experimental results. Finally, conclusion and future work are stated in section 7.

2. THE CENSUS DATA SCHEME
The data scheme for the census data structure of a person in this paper includes as follows:

- Person Name, NRC No, Date of Birth, Place of Birth, Race, Nationality, Religion, Gender, Marital Status, Job Title, Department, Organization, Job Location, Phone No, Hobby, Favorite Music, Favorite Movie, Permanent Address, Current

Address, Native Town, Father Name, Father's NRC No, Father's Job, Mother Name, Mother's NRC No, Mother's Job, Name of Siblings, Name of Children, Wife/Husband's Name, Wife/Husband's NRC No, Wife/Husband's Job..

3. RELATIONAL DATABASE

Relational databases have been evolved for storing large amounts of structured data as tabular form where each column represents a field and each row represents a record. Tables can be related or linked with each other with the use of foreign keys or common columns. They are the database technology of choice for most traditional data-intensive storage and retrieval applications. Retrievals are usually accomplished using SQL, a declarative query language.

3.1 The Relational Database Design for The Census Data

The data scheme for the census data in relational database is as follows:

```
CREATE TABLE `person` (
  `person_id` decimal(15,0) NOT NULL,
  `name` varchar(30) DEFAULT NULL,
  `father_id` decimal(15,0) DEFAULT NULL,
  `mother_id` decimal(15,0) DEFAULT NULL,
  `husband_id` decimal(15,0) DEFAULT NULL,
  `wife_id` decimal(15,0) DEFAULT NULL,
  `gender` varchar(10) DEFAULT NULL,
  `marital_status` varchar(20) DEFAULT NULL,
  `nrc_no` varchar(30) DEFAULT NULL,
  `race` varchar(30) DEFAULT NULL,
  `nationality` varchar(30) DEFAULT NULL,
  `religion` varchar(30) DEFAULT NULL,
  `date_of_birth` date DEFAULT NULL,
  `permanent_address` varchar(100) DEFAULT NULL,
  `current_address` varchar(100) DEFAULT NULL,
  `native_town` varchar(30) DEFAULT NULL,
  `job_title` varchar(50) DEFAULT NULL,
  `department` varchar(50) DEFAULT NULL,
  `organization` varchar(100) DEFAULT NULL,
  `job_location` varchar(80) DEFAULT NULL,
  `phone_no` varchar(30) DEFAULT NULL,
  `hobby` varchar(30) DEFAULT NULL,
  `Alive_Death` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`person_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `relationship` (
  `relationship_id` decimal(15,0) NOT NULL,
  `primary_person_id` decimal(15,0) NOT NULL,
  `secondary_person_id` decimal(15,0) NOT NULL,
  `senior` varchar(30) DEFAULT NULL, PRIMARY KEY
(`relationship_id`,`primary_person_id`,`secondary_person_id`
)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `relationshiptype` (
  `relationship_id` decimal(15,0) NOT NULL,
  `relationship_Type` varchar(30) DEFAULT NULL,
  `remark` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`relationship_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `singertable` (
  `singerId` int(11) NOT NULL,
  `singerName` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`singerId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `songtable` (
  `songId` int(11) NOT NULL,
  `songName` varchar(100) DEFAULT NULL,
  `singerId` int(11) DEFAULT NULL,
  PRIMARY KEY (`songId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `movietable` (
  `movieId` int(11) NOT NULL,
  `movieName` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`movieId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `favoritemusictable` (
  `personId` int(11) DEFAULT NULL,
  `singerId` int(11) DEFAULT NULL,
```

```
  `songId` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `favoritemovietable` (

  `personId` int(11) DEFAULT NULL,

  `movieId` int(11) DEFAULT NULL,

  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

3.2 Relational Queries

Example Query 1 is the relational SQL query statement for retrieving the relationship between “U Zay Yar” and “Mg Zar Ni Linn”. These two persons' relationship is “father and eldest son”. To search for relationship, we need to create all necessary tables for storing personnel information. Firstly, respective person_ids of U Zay Yar and Mg Zar Ni Linn are retrieved from person table. And then, senior column is retrieved from relationship table that are matched with primary_person_id and secondary_person_id that are equal to above person_ids. Senior column indicates the position of the children and so may be eldest, first middle, second middle and youngest son/daughter and so on. Secondly, relationship_id is retrieved from relationship table by using the person_ids of U Zay Yar and Mg Zar Ni Linn. And then, relationship table and relationshipType table are joined based on the relationship_id of relationship table rs and relationshipType table rt to retrieve the relationship_Type of U Zay Yar and Mg Zar Ni Linn. Finally, senior Eldest and relationship type Son are obtained between U Zay Yar and Mg Zar Ni Linn. Example Query 2 and Example Query 3 are also like that the step by step of above query 1.

Example Query 1 : For finding relationship between Father "U Zay Yar" and his Eldest Son "Mg Zar Ni Linn"

```
SELECT (SELECT senior FROM relationship WHERE
primary_person_id = (SELECT person_id FROM person
WHERE NAME LIKE 'U Zay Yar')
AND
secondary_person_id = (SELECT person_id FROM person
WHERE NAME LIKE 'Mg Zar Ni Linn'))AS senior,
rt.relationship_Type
FROM
relationshipType rt JOIN relationship rs ON rs.relationship_id
= rt.relationship_id
WHERE rt.relationship_id = (SELECT relationship_id FROM
relationship WHERE primary_person_id = (SELECT
person_id FROM person WHERE NAME LIKE 'U Zay Yar')
AND
secondary_person_id = (SELECT person_id FROM person
WHERE NAME LIKE 'Mg Zar Ni Linn')) GROUP BY
rt.relationship_Type;
```

Example Query 2 : For finding relationship between Husband " Mg Zar Ni Linn " and his Wife "Ma Khin Me Me"

```
SELECT (SELECT senior FROM relationship WHERE
primary_person_id = (SELECT person_id FROM person
WHERE NAME LIKE 'Mg Zar Ni Linn')
AND
secondary_person_id = (SELECT person_id FROM person
WHERE NAME LIKE 'Ma Khin Me Me'))AS senior,
rt.relationship_Type
FROM
relationshipType rt JOIN relationship rs ON rs.relationship_id
= rt.relationship_id
WHERE rt.relationship_id = (SELECT relationship_id FROM
relationship WHERE primary_person_id = (SELECT
person_id FROM person WHERE NAME LIKE 'Mg Zar Ni
Linn')
AND
secondary_person_id = (SELECT person_id FROM person
WHERE NAME LIKE 'Ma Khin Me Me')) GROUP BY
rt.relationship_Type;
```

Example Query 3 : For finding relationship between Father-in-Law "U Zay Yar" and his Eldest Daughter-in-Law "Ma Khin Me Me"

```
SELECT (SELECT senior FROM relationship WHERE
primary_person_id = (SELECT person_id FROM person
WHERE NAME LIKE 'U Zay Yar')
AND
secondary_person_id = (SELECT person_id FROM person
WHERE NAME LIKE 'Ma Khin Me Me'))AS senior,rt.
relationship_Type
FROM
relationshipType rt JOIN relationship rs ON rs.relationship_id
= rt.relationship_id
WHERE rt.relationship_id = (SELECT relationship_id FROM
relationship WHERE primary_person_id = (SELECT
person_id FROM person WHERE NAME LIKE 'U Zay Yar')
AND
secondary_person_id = (SELECT person_id FROM person
WHERE NAME LIKE 'Ma Khin Me Me')) GROUP BY
rt.relationship_Type;.
```

3.3 Relational Database Lacks In Relationship

Relational databases are schema based, define the structure in advance and deal poorly with connected data. Relational database systems are generally efficient unless the data contains many relationships requiring joins of large tables [1]. Recursive query such as “which customers bought this product who also bought that product?” become more and more expensive as the degree of recursion increases. Reciprocal queries such as “What products did a customer buy?” and “Which customers bought this product?” are so complex to manipulate. Example reciprocal query : “Who are Bob’s friends?” and “Who is friends with Bob?”. It’s possible to get an answer for the query : “Who are my friends-of-friends-of-friends?” in a reasonable period of time. If this query extend to four, five, or six degrees of friendship, performance decline significantly due to the computational and space complexity of recursively joining tables. In this paper, person table is used to store persons record and to describe relationship between persons such as grand father, mother, daughter, son-in-law, father-in-law, uncle nephew and so on, relationship table and relationship type table are created. When the size of the data is more and more large, the huge amount of personnel data and relationship are needed to store in every table. Besides, queries that are used to retrieve relationships between persons are so complex and not flexible to write. And then, query processing time of these complex queries will high and decline performance corresponding to the data size. With requirements changing due to transformation of the IT world, several types of NoSQL databases have emerged and are gaining popularity. Moreover, within computer science, there currently exist several options for storing data outside of a traditional relational model and there has been increased interest in

graphs to represent social networks, web site link structures, chemical structure graphs and others [7].

4. NoSQL DATABASES

NoSQL databases provide a mechanism for storage and retrieval of data that is modeled other than the tabular relations used in relational databases. This systems are also referred to as "Not only SQL" to emphasize that they may allow SQL-like query languages to be used. In the context of the CAP Theorem, NoSQL stores often compromise consistency in favor of availability and partition tolerance [5]. NoSQL provides the flexibility to store entire data in terms of documents instead of conventional method of table-row-column. It is extensively useful when huge amounts of unstructured data or data that's stored remotely on multiple virtual servers are needed to access and analyze. An important aspect of NoSQL databases is that they have no predefined schema, records can have different fields as necessary, this may be referred to as a dynamic schema [6]. Also an important difference between relational databases and NoSQL databases is that they do not fully guarantee ACID properties.

4.1 Types of NoSQL Databases

1. Key-values stores: key-values store is a system that stores values indexed for retrieval by keys. They have a simple data model such as a map or dictionary, allowing clients to put and request values per key. These systems can hold structured or unstructured data and can easily be distributed to a cluster or a collection of nodes as in Amazon's DynamoDB and Project Voldemort.
2. Document-based stores: These databases store data and organize them as document collections, instead of structured tables with uniform sized fields for each record. With this database, users can add any number of fields of any length to a document as implemented in Couch DB and Mongo DB [3] Document stores can be considered to be next step to the key-value stores because they store more complex data than the key-value stores. They store "documents" which allow values to be nested documents or lists as well as scalar values, and the attribute names are dynamically defined for each document at runtime.
3. Column-oriented databases: Column-oriented database is a system that stores data in whole column instead of a row, which minimizes disk access compared to a heavily structured table of columns and rows with uniform sized fields for each record as in HBase and Cassandra. All data stored in a column family is usually of the same type for compressing data in the same column family together.

4.2 NoSQL Database Structure for The Census Data

In key-values store, each personal information such as person name, father name, mother name, NRC No, Date of Birth, Place of Birth, Race, Nationality, Religion, Gender, Marital Status, Job Title, etc... is stored as values in list or map data structure. These values are indexed with their respective keys. In column-family store, each personal information is stored as column oriented structure. In document store, each personal information is stored in each document.

Personnel Information	
Key	**Values**
1	**Name:"U Zay Yar", Father_Name:"U Thein Aung", Mother_Name:"Daw Khin Kyi" , Job_Title ...**
2	**Name:"Mg Zar Ni Linn", Father_Name:" U Zay Yar ", Mother_Name:"Daw Khin Kyawe" , Job_Title ...**
.....	**.....**

Fig 1: Structure of Key-Values Store

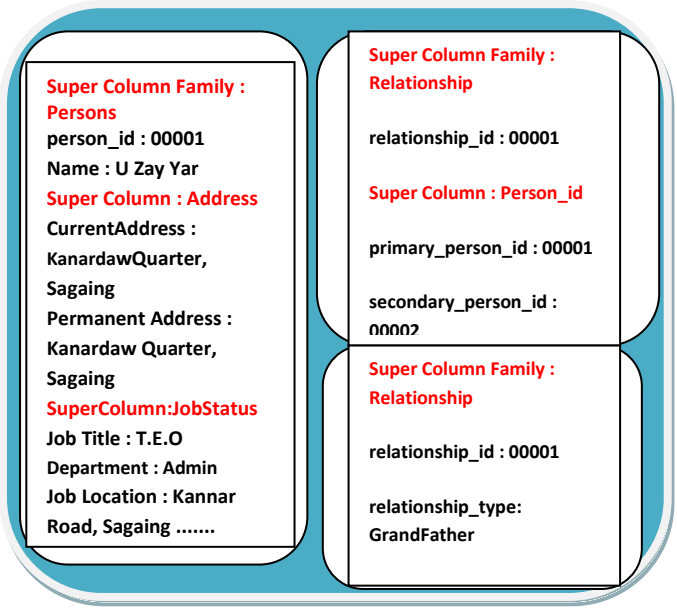

Fig 2 : Structure of Column-Family Store

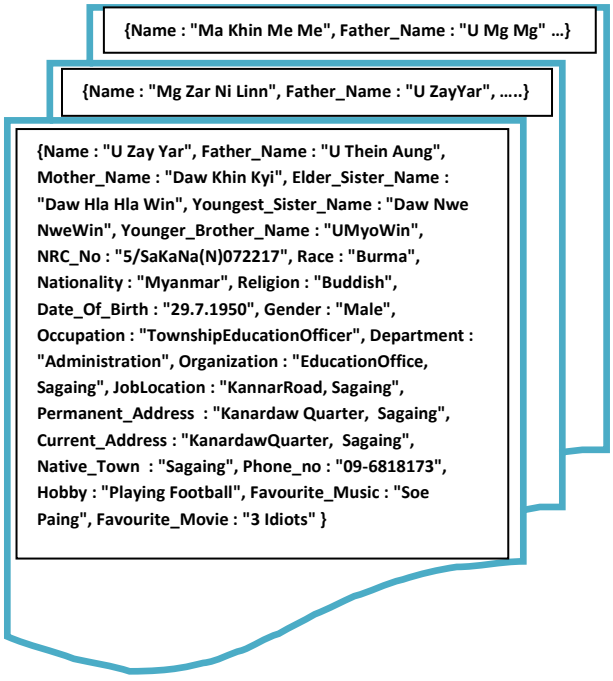

Fig 3 : Structure of Document Store

4.3 NoSQL Databases Also Lack In Relationship

NoSQL databases like key-values store, column-family store and document store consist of disconnected values, columns and documents. According to these stores' structure, information of each person is stored incoherently as shown in above figures. And so, finding relationship between disjointed personnel information is encountered troubles in NoSQL databases. This makes it difficult and so complicated to use query for retrieving relationships among connected personnel

data since these databases also lack relationships. Well-known strategy for adding relationships to such stores is to embed an aggregate's identifier inside the field belonging to another aggregate such as foreign keys. However, this requires joining aggregates at the application level, which quickly becomes prohibitively expensive and decreases performance. Therefore, NoSQL database management systems are useful when working with a huge quantity of data when the data's nature does not require a relational model and retrieving related data [8]. Due to the necessity of relationship nature, graph database technology is emerged to handle this problem.

5. GRAPH DATABASE

Graph databases are databases which store data in the form of a graph. The graph consists of nodes and edges, where nodes act as the objects and edges act as the relationship between the objects. Graph Database Management Systems provide an effective and efficient solution to data storage where data are more and more connected, graph models are widely used, and systems need to scale to large data sets [11]. Graphs are widely used to model social networks and the use of graphs to solve real world problems is becoming necessary. There are several application domains in which the data have a natural representation as a graph. For instance, this happens in the social networking applications, recommendation software, bioinformatics, content management, security and access control, geographic applications, network and cloud management etc... [10]. In these contexts, relational systems are usually unsuitable to store data since they hardly capture their inherent graph structure. Moreover, graph traversals over highly connected data require complex join operations, which can make typical relational operations on this kind of data inefficient and applications hard to scale. For these reasons, a new brand category of data stores, called Graph Database Management Systems (GDBMSs), is emerging [13]. In GDBMSs, data are natively stored as graphs and queries are expressed in terms of graph traversal operations. This allows applications to scale to very large graph-based data sets. In addition, since GDBMSs do not rely on a rigid schema, they provide a more flexible solution in scenarios where the organization of data evolves rapidly.

5.1 Graph Database for The Census Data

For creating person nodes and personnel relationship, CREATE clauses can create many nodes and relationships at once as follows:

Create (n1 : PersonnalInfo {Name : "U Zay Yar", Father_Name : "U Thein Aung", Mother_Name : "Daw Khin Kyi", Elder_Sister_Name : "Daw Hla Hla Win", Youngest_Sister_Name : "Daw New New Win", Wife_Name : "Daw Khin Kyawe", Eldest_Son_Name : "Mg Zar Ni Linn", Middle_Son_Name : "Mg Zar Ni Toe", Youngest_Daughter_Name : "Ma Kay Thi", NRC_No : "5/SaKaNa(N)072217", Race : "Burma", Nationality : "Myanmar", Religion : "Buddish", Date_Of_Birth : "29.7.1950", Gender : "Male", Marital_Status : "Married", Occupation : "Township Education Officer", Department : "Administration", Organization : "Education Office, Sagaing", JobLocation : "Kannar Road, Sagaing",Permanent_Address : "Kanardaw Quarter, Sagaing", Current_Address : "Kanardaw Quarter, Sagaing", Native_Town : "Sagaing", Phone_no : "09-6818173", Hobby : "Playing Football", Favourite_Song : "Ah Thi Ta Yar Ah Nyar Ta Khu", Favourite_Movie : "The Forbidden Kingdom", Alive_Death : "Alive" }),

(n2 : PersonnalInfo {Name : "Daw Khin Kyawe", Father_Name : "U Kyin Sein", Mother_Name : "Daw San Hmi", Elder_Brother_Name : "U Yu", Younger_Brother_Name : "U Tint Wai", Husband_Name : "U Zay Yar", Eldest_Son_Name : "Mg Zar Ni Linn", Middle_Son_Name : "Mg Zar Ni Toe", Youngest_Daughter_Name : "Ma Kay Thi", NRC_No : "5/SaKaNa(N)072232", Race : "Burma", Nationality : "Myanmar", Religion : "Buddish", Date_Of_Birth : "20.7.1952", Gender : "Female", Marital_Status : "Married", Occupation : "Teacher", Department : "Mathametics", Organization : "B.E.H.E (1), Sagaing", JobLocation : " Kannar Road, Sagaing ", Permanent_Address : "Kanardaw Quarter, Sagaing", Current_Address : " Kanardaw Quarter, Sagaing", Native_Town : "Sagaing", Phone_no : "09-33089548", Hobby : "Cooking", Favourite_Song : "Tu Po Tu Po", Favourite_Movie : "Round the World in 80 Days", Alive_Death : "Alive" }),

(n3 : PersonnalInfo {Name : "Mg Zar Ni Linn", Father_Name : "U Zay Yar", Mother_Name : "Daw Khin Kyawe", Younger_Brother_Name : "Mg Zar Ni Toe", Youngest_Sister_Name : "Ma Kay Thi", Wife_Name : "Ma Khin Me Me", NRC_No : "5/SaKaNa(N)155472", Race : "Burma", Nationality : "Myanmar", Religion : "Buddish", Date_Of_Birth : "29.7.1980", Gender : "Male", Marital_Status : "Married", Occupation : "Inspection Technican", Department : "Inspection", Organization : "Total EMP, Myanmar", JobLocation : " KabarAye Pagoda Road, Yankin",Permanent_Address : "Kanardaw Quarter, Sagaing", Current_Address :" No.2 Electric, Minbu", Native_Town : "Sagaing", Phone_no : "09-450002857", Hobby : "Reading", Favourite_Song : "Light", Favourite_Movie : "The Karate Kid", Alive_Death : "Alive" }),

(n4 : PersonnalInfo {Name : "Mg Zar Ni Toe", Father_Name : "U Zay Yar", Mother_Name : "Daw Khin Kyawe", Elder_Brother_Name : "Mg Zar Ni Linn", Younger_Sister_Name : "Ma Kay Thi", NRC_No : "5/SaKaNa(N)155856", Race : "Burma", Nationality : "Myanmar", Religion : "Buddish", Date_Of_Birth : "13.9.1983", Gender : "Male", Marital_Status : "Single", Occupation : "Senior Assistant Engineer", Department : "Mechanical", Organization : "Construction Office, Sagaing", JobLocation : " HtuParYoune Pagoda Road, Sagaing ",Permanent_Address : "Kanardaw Quarter, Sagaing", Current_Address :" Kanardaw Quarter, Sagaing ", Native_Town : "Sagaing", Phone_no : "09-440227457", Hobby : "Playing Football", Favourite_Song : "Believe for one step", Favourite_Movie : "The Myth", Alive_Death : "Alive" }),

……

……

(n3)-[:Father{type: 'Father'}]->(n1),

(n3)-[:Mother{type: 'Mother'}]->(n2),

(n4)-[:Father{type: 'Father'}]->(n1),

(n4)-[:Mother{type: 'Mother'}]->(n2),

(n1)-[:Eldest_Son{type: 'Eldest_Son'}]->(n3),

(n2)-[:Eldest_Son{type: 'Eldest_Son'}]->(n3),

(n1)-[:Younger_Son{type: 'Younger _Son'}]->(n4),

(n2)-[: Younger _Son{type: 'Younger _Son'}]->(n4),

(n4)-[:Elder_Brother{type: 'Elder_Brother'}]->(n3),

(n3)-[:Younger_Brother{type: 'Younger_Brother'}]->(n4),

…….

…….

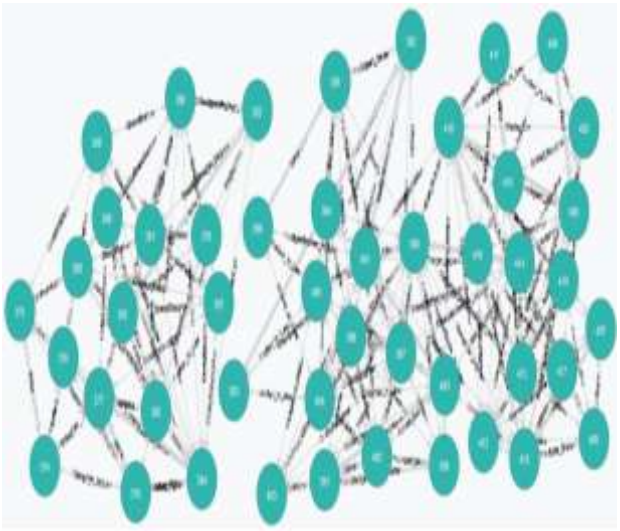

Fig 4 : Structure of Sample Personnel Information Graph

5.2 Graph Queries for Data Searching

In graph database, storing personnel information and relationship is easily done because of the schema free nature and flexible data model such as no need to declare data types for vertices or edges as opposed to the more constrained table-oriented model of a relational database. Example Query 1 is the cypher query statement for retrieving the relationship between "U Zay Yar" and "Mg Zar Ni Linn". These two persons' relationship is "father and eldest son". The relationship between two persons can be easily retrieved in graph database by defining the person names that we want to search in where clause and the relationship finding pattern between two persons in match clause. However, complex SQL query statements are written for finding relationship among two persons in relational database. One of the weaknesses of the relational model is its limited ability to explicitly capture requirement semantics [14]. Big data problems involving complex interconnected information have become increasingly common in the sciences. Storing, retrieving, and manipulating such complex data becomes onerous when using traditional RDBMS approaches. Schema based data models also limits on how information will be stored. There is an involved manual process to redesign the schema in order to adapt to new data. So, while the RDBMS is optimized for aggregated data, graph databases such as Neo4j are optimized for highly connected data. In general, graph databases are useful when we are more interested in relationships between data than in the data itself: for example, in representing and traversing social networks, generating recommendations and conducting forensic investigations.

Example Query 1 : For finding relationship between Father "U Zay Yar" and his Eldest Son "Mg Zar Ni Linn"

START n=node(*) MATCH n-[r]->m WHERE n.name="U Zay Yar" AND m.name="Mg Zar Ni Linn" RETURN n,m,r;

Example Query 2 : For finding relationship between Husband " Mg Zar Ni Linn " and his Wife "Ma Khin Me Me"

START n=node(*) MATCH n-[r]->m WHERE n.name="Mg Zar Ni Linn" AND m.name="Ma Khin Me Me" RETURN n,m,r;

Example Query 3 : For finding relationship between Father-in-Law "U Zay Yar" and his Eldest Daughter-in-Law " Ma Khin Me Me"

START n=node(*) MATCH n-[r]->m WHERE n.name="U Zay Yar" AND m.name="Ma Khin Me Me" RETURN n,m,r.

5.3 Graph Database Embraces Relationship

Graph databases are occurrence based, store data and relationships as they are encountered. Traversing connected data is extremely rapid because it use index-free adjacency. This means that every element contains a direct pointer to its adjacent element and no global index lookups are necessary. In this paper, persons' information is stored as graph structure with persons as nodes and relationships between them as edges by using Neo4j graph database. They can store complex and dynamic relationships of highly connected data. According to the database schema nature, multiple tables are created for searching personnel relationship in relational database but only one personnel information graph is necessary to create in graph database. Furthermore, the queries used to search personnel relationship are so simple and easy to write rather than relational SQL queries. For data of any significant size or value, graph databases are the most excellent way to represent and query connected data within the best performance. Moreover, pattern-matching queries are very difficult to write in SQL, laborious to write in aggregate stores, and in both cases they tend to perform very poorly. Graph databases are optimized for these types of queries, and providing in millisecond responses [12]. Therefore, graph database is the best solution if there is a need for a dynamic data model that represents highly connected data.

6. EXPERIMENTAL RESULTS

This section discusses the experimental results for the query processing time comparison for finding relationship between two persons. In RDBMS, the more bigger the size of the data, the more decline the performance. In GDBMS, although the data size is so large, it can maintain the high performance.

Table 1. Query Processing Time of Relationship in Personnel Information Database

Relationship	RDBMS Execution Time (sec)	GDBMS Execution Time (sec)	No. of Records
person-person	0.016	0.01	~2500
person-person	30.267	0.168	~110000
person-person	1543.505	1.359	~600000

7. CONCLUSION AND FUTURE WORK

This paper gives an overview about the relational database, NoSQL database and graph database by comparing with data model and query model. Myanmar census data, including facts of creating National Registration Card (NRC), personnel profile and other additional information are used to find related information between persons. Relational database

system use two-dimensional table for data creation, with properties like transactions, complex SQL queries, and multi-table related query. Moreover, multi-table queries are not effective for huge data queries when data size is getting bigger and bigger. Scalability in relational databases requires powerful servers that are both expensive and difficult to handle. So, they lack in relationships for finding related data because of joining multiple tables and writing complicated queries. And then, NoSQL databases are effective when we need to process large amount of data with high scalability and to use data among many servers compared to the conventional relational database systems. NoSQL is extensively useful to analyze huge amounts of unstructured data or data that's stored remotely on multiple virtual servers. However, these databases also lack relationships for retrieving connected data because of their disconnected storage structure. In contrast to these NoSQL and relational databases, the relations between the objects are of primary importance and so powerful in graph database. Graph databases support a graph model which allows for a direct persistent storing of the particular objects together with the relations between them. In addition, they should provide an access to query methods that not only deal with the stored objects, but also with the graph structure itself. They are the best technology for dealing with complex, semi-structured, densely connected data and substantially quicker than relational and NoSQL data stores. This paper only analyzes with the query model for finding relationships between persons. For the future work, this paper can be extended to search personnel relationships by utilizing different algorithms for three databases.

8. REFERENCES

[1] R.D.Virgilio, et al. 2013. Converting Relational to Graph Database. In *Proceedings of the First International Workshop on Graph Data Management Experience and Systems* (GRADES 2013), June 23, 2013 - New York, NY, USA.

[2] R. Cattell, 2010. Relational Databases, Object Databases, Key-Value Stores, Document Stores, and Extensible Record Stores: A Comparison.

[3] A Nayak, et al, 2013. Type of NOSQL Databases and its Comparison with Relational Databases. In International Journal of Applied Information Systems (IJAIS) – ISSN: 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 5– No.4, March 2013 – www.ijais.org.

[4] C. Hadjigeorgiou, et al, 2013. RDBMS vs NoSQL: Performance and Scaling Comparison.

[5] S. K. Gajendran. A Survey on NoSQL Databases.

[6] C. J. M. Tauro, et al. A Comparative Analysis of Different NoSQL Databases on Data Model, Query Model and Replication Model. In *Proceedings of the International Conference on "Emerging Research in Computing, Information, Communication and Applications" ERCICA 2013, ISBN: 9789351071020.*

[7] C. Vicknair, et al. 2010. A Comparison of a Graph Database and a Relational Database. ACMSE '10, April 15-17, 2010, Oxford, MS, USA

[8] A. B. M. Moniruzzaman, 2013. *NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison*. International Journal of Database Theory and Application, Vol. 6, No. 4. 2013.

[9] I. Robinson, et al. 2013. Graph Databases. June 2013: First Edition.

[10] M. Buerli, 2012. *The Current State of Graph Databases*. Department of Computer Science, Cal Poly San Luis Obispo, mbuerli@calpoly.edu.

[11] R. Angles, et al. 2008. Survey of Graph Database Models. *ACM Computing Surveys, Vol. 40, No. 1, Article 1, Publication date: February 2008.*

[12] F. Holzschuher, 2013. *Performance of Graph Query Languages*. EDBT/ICDT '13 March 18 - 22 2013, Genoa, Italy.

[13] S. Jouili, et al. An empirical comparison of graph databases.

[14] Hull, R. AND King, R. 1987. Semantic database modeling: Survey, applications, and research issues. *ACM Comput. Surv. 19*, 3, 201-260.