GRAPH DATA MANAGEMENT

Group ID: 43

Group Members:

Student Name	Student ID	Email
Shuai Wang	830166	shuaiw6@student.unimelb.edu.au
Shubham Bawa	936127	sbawa@student.unimelb.edu.au
Atheena Shakthidharan	924244	ashakthidha@student.unimelb.edu

Table of Contents

1. Abstract

Graph databases are mainly used to store data in the form of graphs and represent real world entities in a more natural form compared to any other database. In this report, we have tried to analyse graph database from many perspectives. Firstly, the report discuss about graph data modelling and explain the various components of graph databases such as the data structure, query manipulation languages involved and the integrity constraints. The report analysis and compares graph databases with one of the most prevalent database of today, relational database. The analysis is based on various features that have been used as measures of quality in the industry. The report then examines and analyses the researches and proposals made in the field of graph data management in aspects chosen that include distributed large scale graph, partitioning management, privacy management, clustering and also the real world application domains focusing on social network. As a conclusion to the report, based on the study, and with evidence, a booming future for the graph data is predicted and discussed.

2. Introduction

In a fast-paced technologically advanced world like todays, networks are ubiquitous. Networks are found in all fields of evolution. There are communication networks, sensor networks, financial transaction networks, ecological networks, real-time recommendation networks, disease transmission network, fraud detection networks, local area networks and much more. It is possible to build a network from a prosaic activity like that of a parcel shipment. The most effortless representation of a network is using a graph. The necessity for a simple depiction of the networks increased the need for a graph-structured data management system that is flexible for storing and retrieving such data. The graph database system allows representing entities or objects as nodes and the relationship and interaction between them as the edges. With the increasing size of the network, the necessity for larger graphs to manage voluminous data has grown.

According to the rank of databases by DB-Engines [10], the most popular and dominant databases are relational databases, such as Oracle, MySQL, Microsoft SQL Server and so on. Nowadays, graph databases are rapidly gaining popularity, such as Neo4j which came in No.22 place among all kinds of databases by May, 2018. Although the term of “Graph Database” has been proposed for many years, it recently became practical due to the development of relevant technologies, such as graph algorithms, high-speed storage devices, performance-enhanced CPU and GPU to process graph, powerful sensors used to directly acquire data and so forth [11]. Graph databases lead the great macroscopic business trends today. Those companies that can understand and analyze complex and dynamic relationships in highly connected data of users are able to gain competitive advantages. On the other hand, the volume of data is dramatically growing every day. Compared with other type of databases, graph databases are the best way to represent and query data.

In context of graph data, it is important to understand what graph data modelling refers to. The term of “data model” is widely used in the information management field, which has multiple different meanings. In general, it refers to a collection of conceptual model tools that are used to represent real world entities and relationships among them [13]. Any database system relies on a kind of data model.

2.1 Graph Data Model

A data model contains three fundamental components namely data structures, query and transformation language and integrity constraints [11], which shall be discussed in detail in the following section of the report. Based on data model, a graph database model can be defined as a model that data structures for schema and instances are represented as graphs, query and transformation language indicates graph-oriented operations, and suitable integrity constraints are applied on the graph structure. Some of the most representative graph database models are Logical Data Model (LDM), Hypernode Model, Hypergraph-Based Data Model (GROOVY), Sematic-XT, Graph Database System for Genomics (GGL), Graph Object-Oriented Data Model (GOOD), Graph-Oriented Object Manipulation (GMOD),

Object-Oriented Pattern Matching Language (PaMaL), Graph-Based Object and Association Language (GOAL), Graph Data Model (GDM), Gram, Related Data Model and so on [12]. For other type of database models, there are file system, relational model, hierarchical model, network model, semantic model, entity-relationship model, object-oriented model, XML, semi-structured model and so on [12]. This section of the report focuses on GDM.

GDM has been widely applied in applications that the relationship or topology among data is of great importance. This is due to several advantages of graph modelling. Firstly, graph is a natural way to model data. Compared with table used in relational database, graph allows users to clearly see the information stored and relationship among data. It makes data easy to understand. Secondly, querying data can be based on graph structure. For example, find shortest path from an entity to another, identify neighbors, generate sub-graphs, operate relationships and allow complicated operations allow to be done. In what follows, more details about GDM will be provided.

2.1.1 Data Structure

GDM can model real world entities and relationships. An entity is something that exists as an object, such as a student. A relationship is the correlation between two or more entities, such as a student lives in Melbourne. GDM represents entities and relationships by graphs. A graph is comprised of nodes and edges. Data is stored in the form of nodes, each of them is inter-connected to others by edges and they are relationships. The figure below is a simple example:

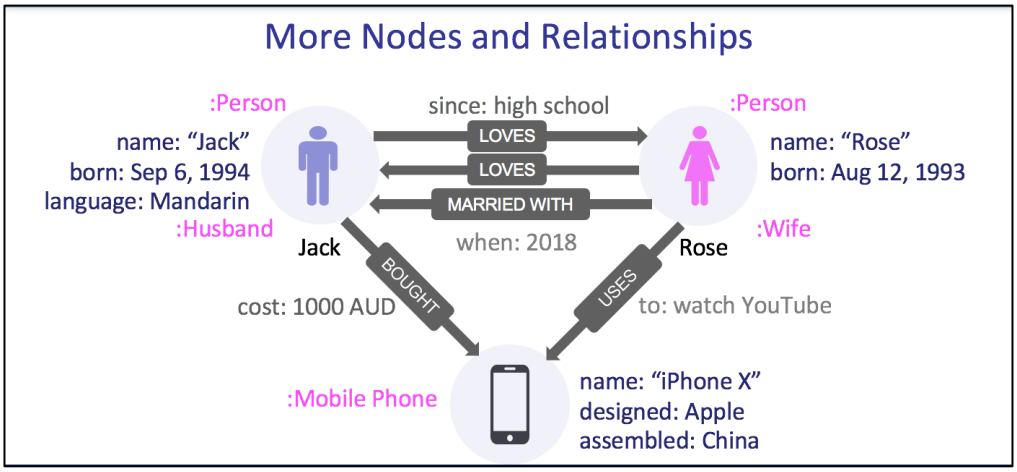

Figure 1: Graph data structure representation

There are three nodes: Jack, Rose and Phone. Each node represents an entity, two of them are human entities and one of them is mobile phone entity.

Information can be stored in a node in the form of property. Each property can be represented as a key-value pair. Here, Jack stored his name of "Jack", birth date on "Sep 6, 1994", language he can speak which is "Mandarin". Rose stored her name of "Rose", birth date on "Aug 12, 1993". Phone stored the name of "iPhone X", the company that design it, and location where it was assembled.

Each node has one or more label(s), which indicates what type the node is representing. Here, both Jack and Rose can be labeled with "Person". Jack can also be a "Husband" node while Rose can also be a "Wife" node. The phone can be labeled with "Mobile Phone".

Relationships are represented as labeled directed edges. Here are some relationships, such as Jack loves Rose since high school, Rose married with Jack in 2018, Jack spent 1000 AUD on this iPhone X. Rose use this phone to watch YouTube. Each relationship has a label, such as "LOVES", "MARRIED WITH", "BOUGHT", "USES". Moreover, each relationship has a direction or bi-direction.

In general, GDM contains two types of relationship: simple relations and complex relations. Simple relations connect entities under a simple property. Complex relations refer to hierarchical networks of relations under additional semantics.

2.1.2 Query and Manipulation Languages

A query language is a variety of operations that are applied to graph structure to manipulate and query data. Several query languages have been proposed and developed, such as G, G+, GraphLog, Cypher, SPARQL, Gremlin and so on [12]. What follows focuses on Cypher.

Cypher is an expressive graph database query language specific to Neo4j. It is designed to help developers, database specialists, users to easily understand and manipulate data. Cypher allows users to ask database to operate data that matches a kind of patterns. This pattern can be expressed by statements. For example,

(Jack)-[:LOVES]->(Rose)-[:USES]->(iPhoneX)

This pattern describes the relationship between Jack, Rose and iPhoneX. It shows a path that connect the three nodes. Like other query languages, Cypher consists of clauses. What follows discusses each clause in more detail.

Create:

Create a node example:

```
CREATE (Jack: Person { name: “Jack”, born:”Sep 6, 1994”, language: ”Mandarin”})
CREATE (Rose: Person { name: “Rose”, born:”Aug 12, 1993”})
CREATE (phone: MobilePhone { name: “iPhoneX”, designed:”Apple”, assembled: ”China”})
```

This clause is to create a node with none or multiple properties. “( )” parenthesis indicate a node. “Jack: Person” means a variable “Jack” and labeled with “Person”. “{ }” brackets add properties to the node. This clause is similar to “INSERT” in SQL, used to insert data.

Create a relationship example:

```
CREATE (Jack)-[:LOVES {since: “high school”}]->(Rose)
CREATE (Rose)-[:LOVES]->(Jack)
CREATE (Rose)-[:MARRIEDWITH]->(Jack)
CREATE (Jack)-[:BOUGHT {cost: “1000 AUD”}]->(phone)
CREATE (Rose)-[:USES {to: “watch YouTube”}]->(phone)
```

This clause is to create a relationship between two nodes. “(A)-[:R]->(B)” represents a directed relationship R from variable A to variable B. “{}” can add properties to relationship.

Match:

Match example:

```
MATCH (a: Person)-[:LOVES]->(b)-[:USES]->(c), (a)-[:BOUGHT]->(c)
WHERE a.name = ‘Jack’
RETURN b, c
```

The MATCH clause is to specify a pattern of nodes and relationships. “MATCH” clause to describe the pattern from known nodes to found nodes. “(a: Person)” represent a single node pattern with label “Person” which will assign matches to the variable “a”. “-[:LOVES]-” matches “LOVES” relationship. “WHERE” clause to constrain the results. “a.name=’Jack’” compare name property with the value “Jack”. “RETURN” clause used to request results. The above query is to find variables b and c that match those patterns.

Match example 2:

```
MATCH (a: Person)-[:LOVES]->()-[:USES]->(product)
WHERE a.name = ‘Jack’ AND product.name=’iPhoneX’
```

RETURN DISTINCT product

"AND" clause is to combine multiple conditions. "DISTINCT" clause means more than one path will match the pattern.

Delete:

Delete a node example:

MATCH (n: Person {name: 'Jack'})
DELETE n

This clause will delete a person node whose name property is 'Jack'. It is used to delete data.

Delete everything example:

MATCH (n)
DETACH DELETE (n)

This clause will delete a data set.

Delete a node and its relationships example:

MATCH (n {name: 'Jack'})
DETACH DELETE (n)

This clause will delete a person node whose name property is 'Jack' and any relationship going to or from it.

Delete a relationship:

MATCH (n {name: 'Jack'})-[r: BOUGHT]->()
DELETE (r)

This clause will delete outgoing "BOUGHT" relationships from the node with the name 'Jack'.

Set:

Set a property on a node example:

MATCH (n {name: 'Jack'})
SET n.birthdate = "Oct 10, 1994"
RETURN n.name, n.birthdate

This clause will update the birth date of Jack.

Set a property on a relationship example:

MATCH (a: Husband {name: 'Jack'})-[r: BOUGHT]->(b: MobilePhone {name: 'iPhoneX'})
SET r.cost = "2000 AUD"

This clause will update the cost property of relationship "BOUGHT".

In Cypher, there are many other clauses, such as MERGE, FOREACH, UNION, WITH, START and so on. The widely-used clauses are MATCH, CREATE, SET and DELETE. They are very similar with SELECT, INSERT, UPDATE and DELETE in SQL. In Neo4j, querying and manipulating data can be done using those clauses.

2.1.3 Integrity Constraints

Codd [14] indicates that integrity constraints are general statements and rules that define the set of consistent database states, or changes of state, or both. In graph database model, integrity constraints include schema-instance consistency, identity and referential integrity constraints, and functional and inclusion dependencies.

In general schema-instance consistency contains two parts: the instance should contain only concrete entities and relations from entity types and relations that were defined in the schema; An entity in the instance may only have those relations or properties defined for its entity type. Thus, all node and edge labels occurring in the super-type in the instance must also occur in the scheme. In this way, incomplete or non-existing information can be incorporated in the database. Another aspect to consider is the degree to which schema and instance are different objects in the database.

Entity integrity assures that each hyper-node is a unique real world entity identified by its content; Referential integrity requires that only existing entities be referenced.

The concept of hyper-node functional dependency denoted by A->B, where A and B are sets of attributes, let us express that A determines the value of B in all hyper-nodes of the database.

In the end, let's remark that the notion of integrity constraint in graph database models is concentrated in the creation of consistent instances and the correct identification and reference of entities. The notion of functional dependency is a feature taken from the relational model, which cannot be easily represented in all graph database models.

Analysis and evaluation of the advantages and disadvantages of graph data model compared with other database models concludes the following. The data structure provided by physical database model is not flexible and difficult to separate from actual implementation. The main differences between graph database model and relational database model are manifold. The data structure of the later is known in advance, and the schema is fixed. It's hard to make change to the structure. Semantic database model is relevant to graph database model as it uses graph-like structure. Object-oriented database model view the world as a set of complex objects while graph database model view the world as a network of relations. Semi-structured database model uses a tree-like structure to represent data.

This report is based on the study of existing graph database systems and is structured to explain how a graph is able to store data and the different ways to store it, a comparison between relational databases and graph databases, various features of management of graph data like clustering, partitioning, privacy preservation and some real time applications of graph and the future trends of the graph databases.

3. Graph Database vs. Relational Database

Whenever it comes to choosing a database technology, Relational Databases always come to mind mainly because of it being around for so many years. Relational Databases' queries are implemented using the SQL query language which is really efficient till the time there are less number of self-joins and relationships between tables, which is when Graph Databases comes into the picture which queries through Cypher query language.

The comparisons between the two databases have been an important area of research for the past few years. Researchers like "Batra S. & Tyagi C in [15]" and "Vicknair C., Macias M, Zhao Z., Nan X., Chen Y. & Wilkins D. in [16]' have analysed the two databases on different criteria and in different perspective.

In [15], the author have made the comparison between the databases on the basis of the execution time of queries and concludes by saying that graph databases take less time than relational databases as Relational databases searches all the data to find the data that meets the search criteria. Whereas, graph databases do not scan the entire graph to find the nodes that meet the search criteria and only looks at the records that are directly connected to other records.

On the other hand, in [16] the authors have made the comparison between the databases objectively and subjectively. Objectively, MySQL performed much slower than graph databases. But at a small scale, MySQL performed slightly better than Neo4j, scaling up resulted in Graph databases performing better. Subjectively, the authors have made the comparison on the level of support, ease of programming, flexibility and security.

Analysing these researches and few others in the industry, it can be noted that the main features of comparison used by the authors are the following.

- **Stability**

The authors in [16] studied about stability in databases and made the comparison. According to them, a system is considered to be stable if it has been well tested over the years and bugs have been fixed it in as and when they were found i.e. constant support is provided to the system. In the case of Relational Databases this support has been provided for decades now which makes it more stable and mature. Whereas, for Graph databases, which are fairly new to the market can be considered less stable and mature as it is still in its growing phase and has not gone through the extensive performance testing like relational databases.

- **Security**

In their paper [15], Batra, S., & Tyagi, C. provides the evidence to say that MySQL has wide-ranging user support for security in the system. Whereas, in Neo4j there are not many mechanisms for security as it presumes a trusted environment.

- **Flexibility**

In [15] a comparison has been made on the level of flexibility and it provides the necessary evidence to say that Relational databases are more stable and secure, but their schema is fixed and difficult to extend which makes unsuitable for managing schemas which evolve over time. Graph databases, on the other hand, have a flexible schema which is easy to extend and is able to handle many relationships between nodes as its focus is on the interconnectivity of data.

- **Scalability**

The author of [17] studied the limitations of Relational databases and says that in order to expand a relational database, one may try to run the database on a more powerful and expensive system. For this, the database must be distributed across more than servers. But relational databases are not able to work in this type of environment because joining the tables the tables is not easy in a distributed system.

4. Management of Graph Data

With the increasing demand for graph-structured data and their application in the real world, there is demanding need to inject, manage and query large volume of such data. Management of these data are extremely important as data in these graph are the most valuable resource of the system. Numerous aspects that need to be considered while dealing with large chunks of data include efficient management of distributed dynamic graphs, partitioning mechanism, clustering approaches, protection of privacy and usage in application domains. Extensive researches have been carried out in these areas and many solutions have been proposed.

4.1 Distributed Large-Scale Dynamic Graph Management

In today's world, robust management of dynamic graph is been studied enormously. In this regard, Mondal, J., & Deshpande, A. [1] designed an 'in-memory distributed graph data management' system. The authors aimed at managing dynamically-changing large scale graphs. An important characteristic of the proposed system is the ability for low latency query processing. Another system named G* was introduced [2]. The authors of [2] concentrated on the efficient storage and querying of graph snapshots and built a distributed system based on it. Graph snapshots are the intermittent snapshots of dynamically changing graphs in a real world application like social network or an enterprise network. On a common note, the authors of [1] and [2] investigated on the drawbacks of partitioning a graph across the networks, the distributed nature of the graph however their focal points were different.

In [1], Mondal, J., & Deshpande A. concentrated on the large computation time taken to answer even a simple query by such a distributed graph system. They emphasised on examining the approach based on

replication of the graph data for building the system. The key idea was to minimise the distributed traversal counts by replicating the data nodes in a memory efficient manner. Based on their study, they built a system that support management of large scale dynamically changing graph that is completely 'in-memory' and uses disk space only as a backing storage. The proposed system implements three main components. First, the system discusses a hybrid, adaptive replication policy by monitoring a node's read-write frequencies to determine dynamically which data needs to be replicated and the type of replication that needs to be used. The second component is a clustering-based approach for the evasion of the decision making costs. Lastly, they suggested how replication decisions should be made based on some fairness criteria. The authors of the paper have done a brief theoretical analysis of the system and also put forward optimization algorithms for problems that they faced in the process.

On the other hand, in [2], the authors studied the distribution and the replication of the data in graph snapshots. Graph snapshots allowed them to explore the differences in the evolution of the graph over time based on various features like network density, clustering coefficient, shortest distance between pairs and cluster size. The drift patterns identified in these analyses are critical and decisive. For example, in a graph representing people, credit card and goods as nodes or vertices and the edges between them represents ownership and purchase relations between them, a dramatic change in the distribution would help identify a case of fraud or malicious activity. The authors of G*studied some of the prevailing single graph systems and multiple graph systems in a distributed environment. They explained the different drawbacks and their motivation to build a system where each graph snapshot is stored in few worker servers rather than all of them, so that the overall queried data is balanced on all the workers. They assumed that the multiple snapshots are queried together.

In [2], Alan.G et al. presented three main idea. They first discussed the challenges in distributing graph snapshots and provided a solution for it which was built on ideas of load balancing, updating the nodes and edges, splitting a complete segment and support of graph processing operations. They also proposed an approach to adaptively identify the placement of replicas with an aim of increasing efficiency. The G* system maintains a 'r' replicas of each graph snapshot on numerous workers in an optimised manner making different types of query traversal efficiently available. This allows the data to be available even in a worst case of 'r-1' worker failures. They evaluated the effectiveness of the system based on certain metrics.

4.2 Partitioning Management

Another important feature of graph data management that is extensively researched about is the partitioning and clustering approach. Effective processing of large graphs require an efficient method of partitioning and clustering data that the graphs represent. Among the various journals and researches in this regard, in this report we choose to analyse three important proposals. First is a 'Self Evolving Distributed Graph Management Environment' (Sedge) proposed by Yang,S.et al.[3], second is an analysis of the balanced graph partitioning which is helpful in parallel computations, and third is min-max cut algorithm that follows the min-max clustering principle.

Sedge aimed to maximize query response time and throughput by reducing the inter-machine communications that are incurred in the process. The authors of [3] investigated the challenge faced in managing large scale graphs into clusters and focused on local graph query attributes like random walk and breadth-first search. Based on their study, they designed Sedge, a two-level partition management architecture system that consists of a primary and a dynamic secondary partition. They elucidated that graphs queries that deal on local and skewed workload, static partitioning techniques are not effective. They also introduced two partitioning mechanisms- complementary partitioning and on-demand partition based on the workload on the graphs for a query processing. They evaluated Sedge and proved that the system could efficiently reduce the inter-machine communications during a distributed graph query processing.

In [4], the authors provided for a thorough analysis of the balanced graph edge partition problem against the alternative native graph vertex partition. For the graph partition issue of assigning edges of vertices to various partitions, the authors characterised the expected computational cost for it. They also proposed approximation algorithms for balanced edge partition issue with and without message aggregation. They were able to deduce the idea that both the issue use balanced vertex partition as a subprogram. This would allow using already known algorithms and techniques for the graph partition issue.

Ding, C. H., et al, introduced a new graph partition algorithm that follows the min-max clustering algorithm. The min-max clustering principle states that the association or similarity between two sub-graphs should be minimised while the summation of the associations within a sub-graph should be maximised. The new algorithm approaches this principle using a simple min-max cut function. The authors have theoretical analysed that the min-max cut function always provides a much better balanced cut than the ratio cut and normalized cut. The linkage-based refinements in the algorithm increased the efficiency and quality of clustering considerably. They also determined another effective partition approach based on the linearized search order based on linkage differential as they were able to explain that it performed better than that based on Feidler vector.

4.3 Privacy Management

Preserving the privacy of sensitive relations in a graph is crucial and is a widely studied area. Elena Z. and Lise G. focussed on the problem of link re-identification and introduced five different strategies to fight it. Link re-identification is the term used for the problem of inferring sensitive relations from anonymised graph data [5]. In this paper, the authors have introduced the strategies based on the amount of data that need to be removed or amount of privacy preserved. The authors have clearly explained the problem and have deduced the strategies with a good explanation. The strategies proposed are node anonymisation, edge anonymisation, Partial-edge anonymisation, Cluster-edge anonymisation and cluster-edge anonymisation with constraints. The authors have concentrated on the problems of link re-identification and illustrated the proposal with clear examples. They have also done an initial evaluation on the effectiveness of the models, with numerous assumptions made with a hope to extend the research further at a later stage

In another research in the area of privacy preservation studied, Brickell,J. and Shmatikov, V [9] introduced new algorithms for privacy preserving computation of ‘all pairs shortest distance’ (APSD) and ‘single source shortest distance’ (SSSD) and another two for privacy in set union. The illustration of the research was on the basis of a scenario where two disrespectful parties possess a graph and would like to join their graphs without leaking of any data and preserving the privacy of the inputs to the join. In the research, they dealt with challenges like graph privacy comparison protocol and graph isomorphism. The journal is well-structured and gives brief description of the definitions, algorithms and its proof. The journal explains why privacy preservation in graph will be a good topic in the future.

4.4 Clustering

Clustering refers to dividing a set of objects into groups, where each group has similar objects and the type of similarity is defined with the help of an objective function. Clustering on graphs is done with the help of some algorithms, which are typically of two types. The first one being a node clustering algorithm which finds the dense regions in a graph on the basis of how the edges behave. The second one being, structural clustering algorithms, which consists of algorithms which cluster different graphs on the basis of the overall structural behavior [20].

Song, S., & Zhao, J. have evaluated 3 graph clustering algorithms and their variations in [21]. They have done the evaluation on data acquired from Amazon review data, which consisted 10 million nodes. They used SNAP’s Clauset-Newman-Moore (CNN), Spectral Clustering and Max-flow/Min-cut.

The Spectral clustering method and Max-flow/Min-cut method come under Node clustering algorithms where a numerical value is associated with each edge in the graph which will be used in order to create clusters. These values can either be a similarity value or it can be a distance value associated to a edge. The partitioning of the graph in these algorithms can be minimized or it can be maximized.

In CNN algorithm, to store the current cluster labels and max-heap of nodes a sparse matrix is used. This allows the algorithm to have a run time of O(mdlog(n)), where n is the number of vertices, m is the number of edges and d is the depth of the largest dendrogram. Dendrogram refers to a tree that is generated using hierarchical agglomerative clustering.

In Spectral Clustering method, the authors describe a basic version of the method that uses the Laplacian matrix, defined as L = D – A, where A is an adjacency matrix and D is the degree of the matrix. The main idea behind the method that if for two points the rows of eigenvectors of the Laplacian matrix are similar then those points belong to the same cluster.

In the Max-flow/Min-Cut method, the graph is divided into disconnected components on the basis of the minimum weight. However, this division results into a problem where singular node is isolated from the rest of the graph. The authors suggest a solution to this problem as well, being Normalized cut, where the weight of the graph is minimized as the optimization of each round which in turn, optimizes the cut's influence over the separated components.

Whereas, Aggarwal, C. C., & Wang, H. in [10], suggested using the maximum flow problem in order to solve this, where the weight is the capacity of the edge and each edge is allowed to have a flow which is at most equal to the capacity.

4.5 Application Domain

The various limitations of the relational database and the advantages of graph database has led towards a renewed interest in graph data management. This has also been motivated because of a great increase in application domains in graph databases such as Social Networks and Web of data, both of which focus on distinct graph models and characteristics and tackle some of the fundamental problems of graph databases.

Social Networks

One of the most predominant and seen-around application of graph databases in today's world is in social networks. The authors of [18] have studied the use of graphs in social networks in a detailed manner. According to them and other researches in the field, it is evident that these days a number of social networking websites such as Facebook are using social graphs in order to manage their networking data, as graphs provide the necessary mechanisms for identifying and creating relationships among nodes and allow for ease in sharing information. These social graphs consist of several properties and nodes, wherein Nodes represent entities such as people, while the edges between these nodes represent relationships among these nodes such as friends, parents or spouses. In the case of Facebook, these social graphs are likely to have the number of nodes in billions.

There are number of queries which are issued on graphs in general, but the authors in [18] have mainly focused on two types of queries that are issued on these social graphs by social networking websites, reading and updating queries wherein, the update queries are mostly about adding a relationship between two people such as friendship or about adding a new friend to a person's friend list and the read queries are mostly about looking up information in the graph such as searching for a node.

The authors have also covered some of the most relevant graph queries and manipulation techniques, such as:

- Updating an individual node (e.g. updating personal details of a person)

- Inserting/updating edges between nodes (e.g. adding friendships)
- Searching (e.g. finding nodes for a user which are directly related to him/her)
- Finding shortest paths (e.g. to know the type of relationship between two nodes)

Web of Data

The authors in [18], have also discussed the use of graph data management for the Web of data and have presented some of their implication in terms of graph models and query execution. They explain how there is a web of data which is growing over the internet, which is creating a huge graph of data items. In this graph, data which is structured is being created, archived and sent over the Web.

The reason for this growth of Web of data relates to the linked data movement, which involves:

- Creating a Unique resource identifier for every published resource over the Web.
- For each resource, publishing the structured data.
- Including links relating to each resource to resources similar to it.

They list some of operations that are performed on web data graphs which are as follows:

- Querying paths (e.g. finding all pair of nodes that are connected to each other by an edge in the graph)
- Querying interference (e.g. finding the nodes on the basis of the properties of their edges)
- Inserting in bulk (e.g. inserting data using a relational database)
- Look-ups on the basis of patterns (e.g. identifying on the basis of constraints on its edges and on its neighbors)
- Querying molecules (e.g. finding nodes or edges or both which are directly or indirectly linked to a given node)
- Retrieving distributed entities (e.g. finding information from entities which are interlinked and stored in separate databases)

The authors in [19], have touched on some of the other application domains of graph data such as Chemical data and Biological data.

- **Chemical data**

Graphs are used in chemistry as well, where the chemical data is represented using graphs with nodes representing as atoms and edges between the atoms represents the bonds between them

- **Biological data**

These graphs are similar to how graphs are in chemical data. But here, the graphs are much larger, e.g. one biological network can consist of hundreds or thousands of nodes and the total size of the databases is large enough making the primary graphs disk-resident.

5. Conclusion and Future Prediction

Based on our study of graph database and graph data management, it is certainly clear that graph database is a trending approach in the industry. The real world applications of graphs have been on a rise and in beginning to dominate the world. Though the future of graph databases may take a bumpy road, it shows an evidence of thrive.

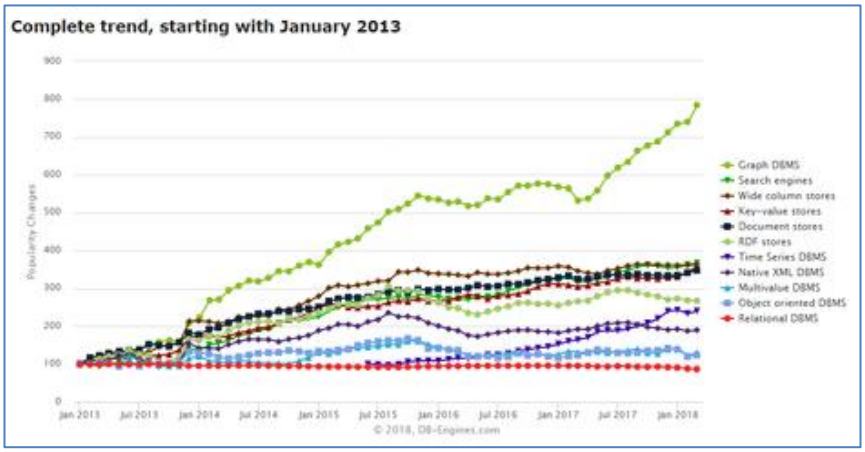

Figure 2: Graph database trend [6]

Figure 1 [6], depicts the trend followed by various database systems from Jan 2013 to Jan 2018. It is perceptible from the graph that there has been a spike in graph data. The main reasons quoted for this trend is the entering of Microsoft and AWS into the field with CosmoDB and Neptune. Some other reasons of its boom is the industry support for the Cypher query language by some predominant enterprises like SAP. The SAP HANA supports a subset of Cypher, mainly for the query processing, making it simple for transforming existing apps into HANA.

IBM's report "The State of Graph Database Worldwide" [7] released in 2017 explained usage and adoption of graph database in its different sectors and industries (Figure 2).The report is based on a survey conducted among nearly 1400 developers and business executives across 74 countries. The figures describes a proof that graph data is being used in multiple industries like network and IT operations, master data management, forensic and fraud investigation, personalization and resource optimization in today's world. It also shows that the company plans to increase it usage of graph database in these sector. This proves the effectiveness and usefulness of the database system in real world applications. In all , the figures explains that there are indeed good number of reasons for people to consider graph database in the applications.

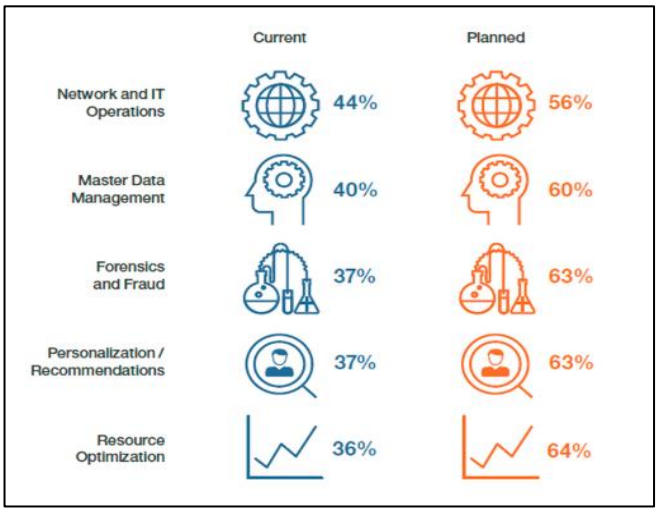

Figure 3: IBM's graph adoption trend

The impressive growth of graph database can be awarded to the belief that it emphasises on the relationships between entities and allows for a flexible schema [8]. This has become crucial in the world today, especially in the world of industries and technologies, as the value of connected data has been

increasing drastically; indeed proving that graph database and its management is one of the top studied approaches today and in future.

6. References

[1] Mondal, J., & Deshpande, A. (2012, May). Managing large dynamic graphs efficiently. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*(pp. 145-156). ACM.

[2] Labouseur, A. G., Olsen, P. W., & Hwang, J. H. (2013). Scalable and Robust Management of Dynamic Graph Data. *BD3@ VLDB*, *1018*, 43-48.

[3] Yang, S., Yan, X., Zong, B., & Khan, A. (2012, May). Towards effective partition management for large graphs. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (pp. 517-528). ACM.

[4] Bourse, F., Lelarge, M., & Vojnovic, M. (2014, August). Balanced graph edge partition. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1456-1465). ACM.

[5] Ding, C. H., He, X., Zha, H., Gu, M., & Simon, H. D. (2001). A min-max cut algorithm for graph partitioning and data clustering. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on* (pp. 107-114). IEEE.

[6] Anadiotis, G. (2018). Back to the future: *Does graph database success hang on query language?* | ZDNet. [online] ZDNet. Available at: https://www.zdnet.com/article/back-to-the-future-does-graph-database-success-hang-on-query-language/ [Accessed 9 May 2018].

[7] Anadiotis, G. (2018). *The continuing rise of graph databases* | ZDNet. [online] ZDNet. Available at: https://www.zdnet.com/article/the-rise-and-rise-of-graph-databases/ [Accessed 5 May 2018].

[8] Computer Business Review. (2018). *Graph Technology - A Data Standby By For Every Fortune 500 Company - Computer Business Review. [online]* Available at: https://www.cbronline.com/enterprise-it/software/graph-technology-data-standby-every-fortune-500-company/ [Accessed 6 May 2018].

[9] *Brickell, J., & Shmatikov, V. (2005, December). Privacy-preserving graph algorithms in the semi-honest model. In International Conference on the Theory and Application of Cryptology and Information Security (pp. 236-252). Springer, Berlin, Heidelberg.*

[10] https://db-engines.com/en/ranking.

[11] Angles, R., & Gutierrez, C. (2017). An introduction to Graph Data Management. arXiv preprint arXiv:1801.00036.

[12] Angles, R., & Gutierrez, C. (2008). Survey of graph *database models. ACM Computing Surveys* (CSUR), 40(1), 1.

[13] Silberschatz, A., Korth, H. F., & Sudarshan, S. (1996). Data models. ACM Computing Surveys (CSUR), 28(1), 105-108.

[14] Codd, E. F. (1988). Relational database: a practical foundation for productivity. In Readings in Artificial Intelligence and Databases (pp. 60-68).

[15] Batra, S., & Tyagi, C. (2012). Comparative analysis of relational and graph databases. International Journal of Soft Computing and Engineering (IJSCE), 2(2), 509-512.

[16] Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., & Wilkins, D. (2010, April). A comparison of a graph database and a relational database: a data provenance perspective. In Proceedings of the 48th annual Southeast regional conference(p. 42). ACM.

[17] Leavitt, N. (2010). Will NoSQL databases live up to their promise?. Computer, 43(2).

[18] Cudré-Mauroux, P., & Elnikety, S. (2011). Graph data management systems for new application domains. Proceedings of the VLDB Endowment, 4(12).

[19] Aggarwal, C. C., & Wang, H. (2010). Graph data management and mining: A survey of algorithms and applications. In Managing and mining graph data (pp. 13-68). Springer, Boston, MA.

[20] Aggarwal, C. C., & Wang, H. (2010). A survey of clustering algorithms for graph data. In Managing and mining graph data (pp. 275-301). Springer, Boston, MA.

[21] Song, S., & Zhao, J. Survey of graph clustering algorithms using amazon reviews.