International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
International Conference on Industrial Automation And Computing (ICIAC- 12th & 13th April 2014)

RESEARCH ARTICLE OPEN ACCESS

Modern Graph Databases Models

Kalyani N.Satone
M.E (C.S.E)
S.D.C.E Wardha
kalsatone@gmail.com

Abstract:
Graph Database Models is increasingly a topic of interest the database area. The representation of data in the form of a graph lends itself well to structured data with a dynamic schema. This paper goes over current applications and implementations of graph databases, giving an overview of the different types available and their application. Due wide spread of graph algorithms and models, no standard system or query language has been defined for graph databases. Research and industry adoption will determine the future direction of graph databases.
Keywords: Database Models, Graph Data, NOSQL, Graph Traversal.

I. INTRODUCTION

One of the most general and powerful data structures useful in a variety of applications are graphs. In the past few years there has been a repetition of interest in storing and managing graph data. In academia and research, we see many new attempts at providing a database model for large graph data, particularly social graphs and the Web graph. While, more and more commercial applications are looking towards graph databases for their dynamic schema and ease of use in storing more complex data. This paper will go through many of the current database models giving a comparison of the different design implementations and trades.

Historically the birth of graph theory is attributed to the Swiss mathematician Leonhard Euler, who first solved the Seven Bridges of Konigsberg problem in 1736. This problem introduced the concept of representing data in the form of a graph (a set ofvertices or nodes with edges connecting them) and determining the traversal of the graph that results in every edge being crossed only once. Graph theory translates to today's work in computational biology and social graphs with shortest path queries, clustering, community detection, and other graph algorithms. The optimization of these queries separates graph databases from the rest.

The research of graph databases was popular in the early 1990s with database models like LDM, GOOD, O2, and GraphDB. However, this interest died with the uprising of XML and the Internet. Not until recently have graph databases again become a topic of interest. This re-emergence is due in part to the large amounts of graph data introduced by the Web. The first graph conference Graph Connect 2012[7] - was held focusing only on graph databases and the adoption of such models.

The recent trend, following the NOSQL movement, has moved away from relational databases to ones better suited for a given application. While much of this movement is focused around the horizontal scalability of data with column-store and key-value-stores, the graph data model provides a greater level of data complexity in comparison. Figure 1 shows a categorization of NOSQL data models, comparing data complexity versus data size. Graph data models provide a higher level of data complexity in return for being able to handle less data.

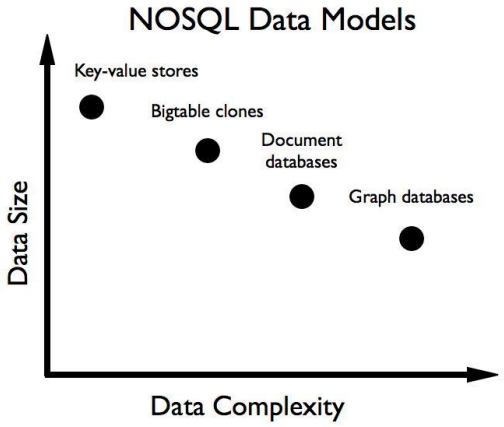

Figure1 : NO SQL DataModels

II RELATED WORK

In particular Renzo Angles[19] presents a well-rounded survey of graph data models and their features. The paper has given multiple comparisons of graph data models with respect to data storing, data structure, query languages, and integrity

constraints. For a survey of earlier work (pre-NOSQL) in graph databases Angles and Gutierrez[20] present a survey of graph database models prior to 2002, particularly geographical, spatial and semi-structured database models. It is important to notice the shift of focus between the two papers, with respect to data structure. Older data models focused heavily on semi-structured and XML data in a traditional database.There is also a trend of abstraction by database models only providing API's for operation and manipulation.

As many of the graph databases remain unchanged from these surveys, this paper will instead focus more on the application of each database model and categorize them into different types.

III GRAPH APPLICATIONS

A Graphsnot only store the data dynamic schema,but also provides representations of data which is not previously possible. The ability overlay different graphs (Ex. social, temporal, and special) on data extends the functionality of querying data. In Managing and Mining Graph Data[18] introduced to a variety of applications for graph data, focusing on three major groups: chemical and biological data, social networks, the Web,Data Mining, Enterprise Data with the addition of Security and access rights.

3.1 Chemical and Biological

Chemical data is modeled as a graph by assigning atoms as nodes and bonds the edges between them. Biological data is represented the same way, only with amino acids as the nodes and links between them as the edges. This graph data is important for such operations as drug discovery and analysis.

The data has many repeating node labels, so graph operations are focused at pattern recognition. Pattern recognition is done by finding frequent sub-graphs of a given graph. Modeled with a traditional database model, these operations would take a great deal longer, due to the recursive nature of traversing a graph.

3.2 Social Networks

Social Networks is a very popular topic not just in society, but in graph research. Social networks, not only introduce a profound amount of data, but presentlarge graph data problems for the research community. These graphs, not only store nodes of people but also link nodes of multimedia, relationships, and messaging. For large social graphs we are most interested shortest path queries and clustering. These graph algorithms provide analysis of relations of two nodes and determination of communities or social networks. Currently social networking sites like Facebook do not use a graph databases. Instead they use keyvalue stores or column stores in associated with Cassandra (a column store similar to BigTable). Definitely it handles large amounts of data across many commodity server with no single point of failure.

3.3 The Web

The Web, in its entirety, is essentially a graph of data and information linked together. Cudre-Mauroux et al.[23] defined the Web in terms of the Linked Data movement which supports the rapid dissemination of largescale structured data through three principles: i) Unique Resource Identifiers (URIs) establish the creation of distinct data anywhere on the web. ii) Structured data, usually in the form of Resource Description Framework (RDF) triplets, provides a standard structure for data to be linked by. iii) Links to similar online resources connect the data to form communities or clusters of data. This massive graph of data presents applications in web search and data collection. PageRank, possibly on of the most well known secrets of Google. Other important graph algorithms include webdocument clustering and keyword search. Both of these algorithms aid in the searching and narrowing of data sets. Applications that deal with web data, if on a smaller scale can efficiently provide on-line querying of this graphlike data. For applications that focus on largerscale graph data, online querying provides analytics and aggregates of graph data.

3.4 Enterprise Data

Graph databases are not limited to academia or largedata graphs. Enterprise data provides perhaps the largest uptake of applications for graph database models. Modeling of data as a graph is not limited to scientific c or web data; rather you can model most anything as a graph. The advantage of using graphs is the ability to represent more complex data models and support a dynamic schema. In particular, graph databases have been successful for companies that store hierarchies of product and financial and industry data[7]. The ordnance of modeling data with relationships, allows for efficient restructuring as well as multiperspective querying. Graph algorithms are utilized the most, with applications such as these where data analytics are a large part of business. Another possible application worth mentioning is the use of graph databases for bug localization[18]. Overall there is a wide range of areas where graph data models are applicable.

3.5 Mining Data

You can traverse the relationships in a graph database to investigate the direct and indirect interactions between nodes. These interactions can help you to spot trends in your data and enable you to act on these trends.

3.6 Determining Security and Access Rights

Many distributed systems need to be able to query and track the access rights that have been granted to a large number of users over an equally large volume of resources. When an application requests access to a resource on behalf of a user, it is important to be able to resolve this request quickly and accurately. A simple and efficient solution is to create graph database that stores information about users and resources, and implements access rights as the relationships between these entities.

IV GRAPH DATABASE MODELS

A graph database is a database that uses graph structures with nodes, edges, and properties to represent and store data. A graph database is any storage system that provides indexfree adjacency. This means that every element contains a direct pointer to its adjacent elements and no index lookups are necessary.

There is a wide range of graph database models that have been introduced throughout the past few years. From implementations on top existing nonrelational database models to graph database models build from the ground up, there is no standard graph database model on which graph algorithms are developed[30]. Rather, each graph database is optimized for a specific set of task or queries. The problem resides in the multiple divisions of graph databases. Graph databases can focus on graph algorithms like shortest path queries and subgraph matching which require the whole graph to reside in memory and make distributed systems very difficult. On the other side of the spectrum, a graph database can focus on handling large graphs by scaling horizontally. This however makes many graph algorithms extremely inefficient or even impossible. With respect to the recent developments in the area, the different graph database models published a comparison matrix that included information like software features, schema features, query features, general database features, database operation utilities, language bindings and operating systems.

The following sections organize the different graph database models into categories corresponding to the data model type.

4.1 Graph Databases

The main-stream graph databases provide an object model for nodes and relationships. These graph databases focus on either RDF triplets, linked data, or relationships for storage. These databases often use direct memory links to adjacent nodes rather than requiring joins or keys lookups.

AllegroGraph[1] is a high-performance, software oriented database model that came as a precursor to the current generation of graph databases. It is implemented as an RDF databases, and serves as a reference implementation for the SPARQL query language. Implementations of geotemporal reasoning and social network analysis extend the functionality of the database as well as a prolog extension. AllegroGraph also partially enforces ACID while remaining scalable.

DEX[3, 26] is a very efficient, bitmaps-based graph database model written in C++. The focus of DEX is performance in the management of very large graphs, and even allows for the integration of various data sources. In addition to the large data capacity, DEX has a good integrity model for management of persistent and temporary graphs. Operation or core functionality like link analysis, social network analysis, pattern recognition and keyword search is done through their Java API. These core functionalities lend themselves well to applications like IMDb, on which experiments were done[26].

Neo4j[11] is a disk-based transactional graph database advertised as “The world leading graph database". It works on a network oriented model with relations as first class objects. The API is in Java, and supports Java object storage. The systemis very efficient in graph traversals, however currently requires the full dataset on each node (work is being done on transparent partitioning). Neo4j also has partial ACID support and lends itself well to transactional enterprise solutions.

HyperGraphDB[9, 24] is an opensource database focused on supporting generalized hypergraphs. Hypergraphs differ from normal graphs in their ability for edges to point to other edges. This representation is useful in the modeling of graph data for artificial intelligence, bioinformatics, and other knowledge representations. Hypergraph supports online querying with a Java API.

Sones[15] is an object-oriented database written in C#. The graph database model provides its own query language based on SQL and supports a higher level of abstraction for graph queries. The model is based on weighted graphs and also has sup-port for hypergraphs. Sones runs on a distributed system to support scalability.

4.2 Distributed Graph Databases

Distributed Graph databases focus on distributing large graphs across a framework. Partitioning graph data is a non-trivial problem, optimal division of graphs requires findingsubgraphs of a graph. For most data, the number of links or relationships is too large to efficiently compute an optimal partition, therefore most databases userandom partitioning.

Horton[8, 28] is a transactional graph processing framework created by Microsoft. Horton makes use of the Orleans cloud framework in order to query large distributed graphs. Instead of adopting a map/reduce architecture, Horton works with a distributed graph, passing a state machine across nodes. This allows for better ad-hoc querying in comparison to map/reduce systems.

InfiniteGraph[10] is a distributed-oriented system that supports large-scale graphs and efficient graph analysis. Rather than in-memory graphs, this system supports efficient traversal of graphs across distributed data stores. This works by creating a federation of compute nodes operated through their java API.

4.3 Key-Value Graph Databases

Key-value graph databases simplify the object-related model of graph databases to allow for greater horizontal scalability. These models build on, or on top of, existing keyvalue stores allowing for greater scalability and partitioning of graph nodes.

VertexDB[17] is a keyvalue disk store that makes use of TokyoCabinet. The graph database focuses on a vertex graph with added support for automatic garbage collection.

CloudGraph[2] is an in-development, fully transactional graph database written in C#. It takes advantage of key/value pairs to store data both in-memory and on-disk. CloudGraph has also created its own graph query language (GQL).

RedisGraph[14] is an implementation of a graph database in python using redis. Redis is a modern key-value store; the python implementation is minimalistic, creating an API in only forty lines of code.

Trinity[16, 29] is a RAM-based key value store under development by Microsoft Research. It uses message passing over a distributed system, achieving low latency queries on large distributed graphs. The benefit of in-memory key value storage can be seen with increased performance.

4.4 Document Graph Databases

Like key-value stores, document based graph databases introduce a higher level of data complexity for a given node.

Orientdb[12] is a high-performance documentgraph database. They make use of a novel distributed hash table algorithm in order to get greater parallelism.

Another example of a document-store in graph databases is an implementation on CouchDB[27]. This implementation makes use of the document store, in order to serve low latency queries for large graph databases. Document stores, much like key-value stores provide quick data retrieval for structured data.

4.5 SQL Graph Databases

Filament[4] is a graph persistence library built on top of PostgreSQL. It allows SQL querying through JDBC with navigational queries for querying the graph data.GStore[5] is a prototype query language and storage manager for large graphs. It is also build on top of PostgreSQL.

These implementations of graph databases are often referred to as Graph stores, for the implementation only concerns storage and retrieval of a graph data from the database, not how the data is stored.

4.6 Map/Reduce Graph Databases

To handle very large graphs, one can implement Map/Reduce functionality, in order to achieve the maximum amount of parallelism. Partitioning nodes of a graph across many machines will result in only a sizable amount of computation to be one on each machine.

Pregel[25] is a vertex-based infrastructure for graphs built on top ofHadoop.Hadoop, a Map/Reduce framework provides batch jobs for processing the distributed vertices with message passing. These approach only accords doing online queries of the graph data.

Phoebus[13] is another implementation ofPregel, again building on top of Hadoop in order to benefit from the Map/Reduce framework.

Giraph[6] also builds of Pregel with the addition of fault tolerance. If the application coordinator has a fault, one of the available nodes will automatically become the new coordinator.

V.COMPARISONS

Multiple studies have been done comparing the performance of graph databases and relational ones. Graph databases like Neo4j[11] optimize for adjacency queries and graph traversal. While some operations may not be as fast as the indexing provided in a SQL database, the overall performance when doing graphlike queries will be much improved. Things to look for in graphlike queries are, lots of many tomany relationships, having tree like characteristics, or requiring frequent schema changes. In one comparison Neo4j and MySQL[22], the

authors found that graph databases did perform better than the relational model on the objective queries. However they noted that Neo4j is not yet mature, and because there is no standard query language available it only added to this. Another paper looked at how graph databases like Neo4j performed on special data[21]. The paper found that relational databases still performed better, in all spatial queries but the ones that involved hierarchical traversal. The fact that a relational database can quickly index a coordinate location gives an advantage to relational databases.

In contrast test run with a directed acyclical graph on Neo4j and MySQL[31], showed a clear advantage of graph databases for structural queries. When comparisons focus on structured data with graphs that are fairly dense, relational indexing performance with joins can no longer keep up with the linked data representation in graph databases.

VI. CONCLUSIONS

This paper gave an overall summary of the current state of graph databases. Much of the current research in the application driven. However, in turn, the various applications have made a wide assortment of graph databases. In order to possibly enumerate all the different categories of graph databases, this paper went over many of the current graph database models being used today. Graph database models are divided bya number algorithms and paradigms which databases wish to optimize. There still does not exist a standard query language for graph databases, leading many implementations to be API only. The future of graph databases resides in the prevalence of one database over another, most likely determined by the enterprise industry and their adoption. Overall graph databases provide a much needed structure for storing data and incorporating a dynamic schema, however the research topic itself needs more structure before it can fully be adopted by industry.

References

[1] Allegrograph.http://www.franz.com/agraph/ allegrograph/.
[2] Cloudgraph. http://www.cloudgraph.com/.
[3] Dex. http://www.sparsity-technologies. com/dex.
[4] Filament. http://filament.sourceforge. net/.
[5] G-store. http://g-store.sourceforge.net/.
[6] Giraph. https://github.com/apache/giraph.
[7] Graph connect. http://www.graphconnect. com/.
[8] Horton. http://research.microsoft.com/ en-us/projects/ldg/.
[9] Hypergraphdb. http://www.hypergraphdb. org/.
[10] Infinitegraph. http://infinitegraph.com/. Neo4j. http://www.neo4j.org/.
[11] Orientdb. http://www.orientdb.org/.
[12] Phoebus. https://github.com/xslogic/ phoebus.
[13] redis graph. https://github.com/tblobaum/ redisgraph.
[14] Sones.http://www.dekorte.com/projects/open source/vertexdb/.
[15] Trinity. http://research.microsoft.com/ enus/projects/ trinity/
[16] Vertexdbhttp://www.dekorte.com/ projects/opensource/vertexdb/.
[17] C.C. Aggarwal and H. Wang. Managing and mining graph data, volume 40. Springer, 2010. Renzo Angles. A comparison of current graph database models. In Proceedings of the 2012 IEEE 28th International Conference on Data Engineering Workshops, ICDEW '12, pages 171{177, Washington, DC, USA, 2012. IEEE Computer Society.
[19] Renzo Angles and Claudio Gutierrez. Survey of graph database models. ACM Comput. Surv., 40(1):1:1{1:39, February 2008.
[20] BLP Baas. Nosqlspatial{neo4j versus postgis. 2012.
[21] S. Batra and C. Tyagi. Comparative analysis of relational and graph databases. International Journal of Soft Computing, 2.
[22] P. CudreMauroux and S. Elnikety. Graph data management systems for new application do-mains. In International Conference on Very Large Data Bases (VLDB), 2011.
[23] B. Iordanov. Hypergraphdb: a generalized graph database. Web-Age Information Management, pages 25{36, 2010.
[24] GrzegorzMalewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, NatyLeiser, and GrzegorzCzajkowski. Pregel: a system for largescale graph processing. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, SIG-MOD '10, pages 135{146, New York, NY, USA, 2010.ACM.
[25] Norbert Mart nez-Bazan, Victor Muntes-Mulero, Sergio Gomez-Villamor, Jordi Nin, Mario-A. Sanchez-Mart nez, and JosepL. LarribaPey. Dex: high-performance exploration on large graphs for information retrieval. In Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, CIKM '07, pages 573{582, New York, NY, USA, 2007. ACM.

[26] JayantaMondal and AmolDeshpande. Managing large dynamic graphs efficiently. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, SIGMOD '12, pages 145{156, New York, NY, USA, 2012. ACM.

[28] M. Sarwat, S. Elnikety, Y. He, and G. Kliot. Horton: Online query execution engine for large distributed graphs. In Data Engineering (ICDE), 2012 IEEE 28th International Conference on, pages 1289{1292. IEEE, 2012.

[29] B. Shao, H. Wang, and Y. Li. The trinity graph engine. Technical report, Technical Report 161291, Microsoft Research, 2012.

[30] Bin Shao, Haixun Wang, and Yanghua Xiao. Managing and mining large graphs: systems and implementations. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, SIGMOD '12, pages 589{592, New York, NY, USA, 2012. ACM.

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 South-east Regional Conference, ACM SE '10, pages 42:1{42:6, New York, NY, USA, 2010. ACM