ISSN:0975-9646

Graph Databases- An Overview

ShefaliPatil[1] , GauravVaswani[2] , Anuradha Bhatia[3]

[1] *Student, ME Computers, Terna College of Engg, Navi Mumbai*
[2] *Student, , Computer Technology, VESIT, Mumbai*
[3] *Computer Technology Department, VES Polytechnic,Mumbai*

Abstract: For a long time, data has been typically stored in tabular form so as to increase the indexing and readability. Nowadays, the trends are changing as Graph databases are quickly gaining popularity. In fact, it would not be wrong to call them "the future of DBMS". 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 denied for graph databases. Research and industry adoption will determine the future direction of graph databases.

Keywords: Graph databases, Application areas, graph models, distributed databases, key value databases.

I. INTRODUCTION - GRAPHS

Graphs are the most generic form of storing data in a visual manner in the world of data structures. Graphs store data in the form of nodes (data blocks) where one node points to another. We can reach any data block from another.

II. WHAT ARE GRAPH DATABASES?

Technically, Graph Databases are a way of storing data in the form of **nodes, edges and relationships** which provide index-free adjacency.

Data is stored in the form of *nodes*, every node (or data block) is connected to another one and this connection is called an *edge*. A few words are also mentioned on these edges to further define the connection between one node and the other – this description is called relationship. Since each node can directly look-up the node it is connected to (they are all connected through edges, remember?), this eliminates the need of searching a data block by its 'index', hence the term 'index-free adjacency'.

Today, most of the social networking sites like Facebook use graph databases to store their massive amount of data.Usage of Graph databases

Graph databases find their usage when data to be stored is associative, meaning when the relationship between two data blocks matters a lot. Relational databases (tabular form) are not that good when a relationship exists between two data blocks, especially if it's the relationship that's more important than the actual data blocks. Graph databases are a very intuitive and expressive way to describe any form of data – as if we're writing something on whiteboards. They let you represent related data as it is – as a set of objects connected by a set of relationships each with its own set of descriptive properties.

III. THE PROPERTY GRAPH MODEL

- A property graph is made up of nodes, relationships and properties.
- Nodes contain properties. Think of nodes as documents that store properties in theform of arbitrary key-value pairs. The keys are strings and the values arbitrary datatypes.
- Relationships connect and structure nodes. A relationship always has a direction,a label, and a *start node* and an *end node*--there are no dangling relationships.Together, a relationship's direction and label add semantic clarity to the structuringof nodes.
- Like nodes, relationships can also have properties. The ability to add properties to relationships is particularly useful for providing additional metadata for graph algorithms, adding additional semantics to relationships (including quality and weight), and for constraining queries at run time.These simple primitives are all we need to create sophisticated and semantically rich models. So far, all our models have been in the form of diagrams.
- Diagrams are great for describing graphs outside of any technology context, but when it comes to using a database, we need some other mechanism for creating, manipulating and querying data.

Example

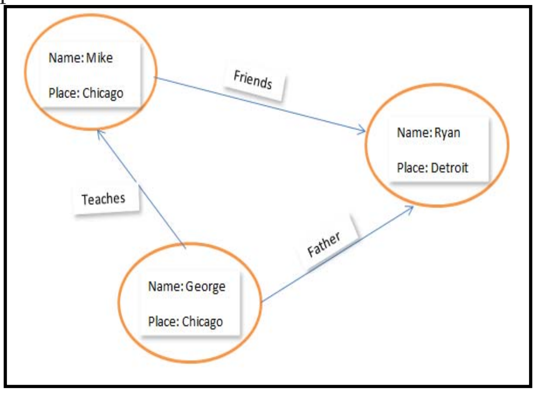


Figure 1

In this example, Mike has a teacher named George, whose son Ryan is his friend. This has been represented by writing relationships on the edges and properties in the circle. This is very close to how the data is actually stored in a graph database. It would not have been that easy if we were using a table to depict such a relationship.

IV. The Power Of Graph Databases

Performance: One compelling reason, then, for choosing a graph database is the sheer performance increase when dealing with connected data versus relational databases and NOSQL stores. In contrast to relational databases, where join-intensive query performance deteriorates as the dataset gets bigger, with a graph database performance tends to remain relatively constant, even as the dataset grows. This is because queries are localized to a portion of the graph. As a result, the execution time for each query is proportional only to the size of the part of the graph traversed to satisfy that query, rather than the size of the overall graph.

Flexibility: As developers and data architects we want to connect data as the domain dictates, thereby allowing structure and schema to emerge in tandem with our growing understanding of the problem space, rather than being imposed upfront, when we know least about the real shape and intricacies of the data. Graph databases address this want directly. Graphs are naturally additive, meaning we can add new kinds of relationships, new nodes, and new sub graphs to an existing structure without disturbing existing queries and application functionality. These things have generally positive implications for developer productivity and project risk. Because of the graph model’s flexibility, we don’t have to model our domain in exhaustive detail ahead of time—a practice which is all but foolhardy in the face of changing business requirements. The additive nature of graphs also means we tend to perform fewer migrations, thereby reducing maintenance overhead and risk.

Agility: We want to be able to evolve our data model in step with the rest of our application, using a technology aligned with today’s incremental and iterative software delivery practices. Modern graph databases equip us to perform frictionless development and graceful systems maintenance. In particular, the schema-free nature of the graph data model, coupled with the testable nature of a graph database’s API and query language, empower us to evolve an application in a controlled manner

V. Real World Use Cases Of Graph Databases

Graph databases are becoming very popular in the real world. Here are some arenas where graph databases have found their use among the world's leading companies:

Page rank: Google uses the concept of graph databases in calculating the order of displaying the search results. A directed graph is used to connect the world wide web pages together as nodes and the hyperlinks to each other as the edges. The number of outgoing edges per graph is assigned as the weight for the edge. Thus, page rank is decided as per the weight on one edge as compared to other edges.

Data Management: Cisco, one of the world's leading networking organizations, has recently adopted a hierarchical management system which is centrally based on the graph utility of the Neo4j database. This provides them with a very fast access to data as compared to Oracle RAC. They are implementing this concept on product hierarchy too in order to serve the user in real time.

Social Interconnect: Websites like Facebook, Twitter, LinkedIn, Viadeo, Glassdoor are storing their connections in the form of graph databases as relationships. Recommendations are important from the point of view of their users. Relationships and connections can be very well managed and accessed in real time as compared to relational databases.

Network management: Telecommunication companies like SFR, Telenor, Huwai, Just Dial have shifted to graph databases to model their network which consists of highly interconnected plans, customers and groups. Graphs help them in analyzing networks and data centers and also save them from the conventional time-consuming process of authentication. Most importantly, by using graphs, the failure cases are also covered and recovery plans are always just a node away which obviously saves a lot of time whenever any hazard occurs.

Security and access management : The creative cloud of Adobe uses a graph database structure to link authentication details and thereby grant access to contents for its administrators as well as users.

Bioinformatics : Era7 is a company that deals with DNA sequencing i.e., storing information on proteins, enzymes etc. This is done with the help of Bio4j, which is a bioinformatics graph DB system. It stores the information about genes, proteins and other complex interrelated information. Bio4j has all the features of Neo4j, world’s leading graph database, and is thus very scalable and flexible.

VI. Graph Applications

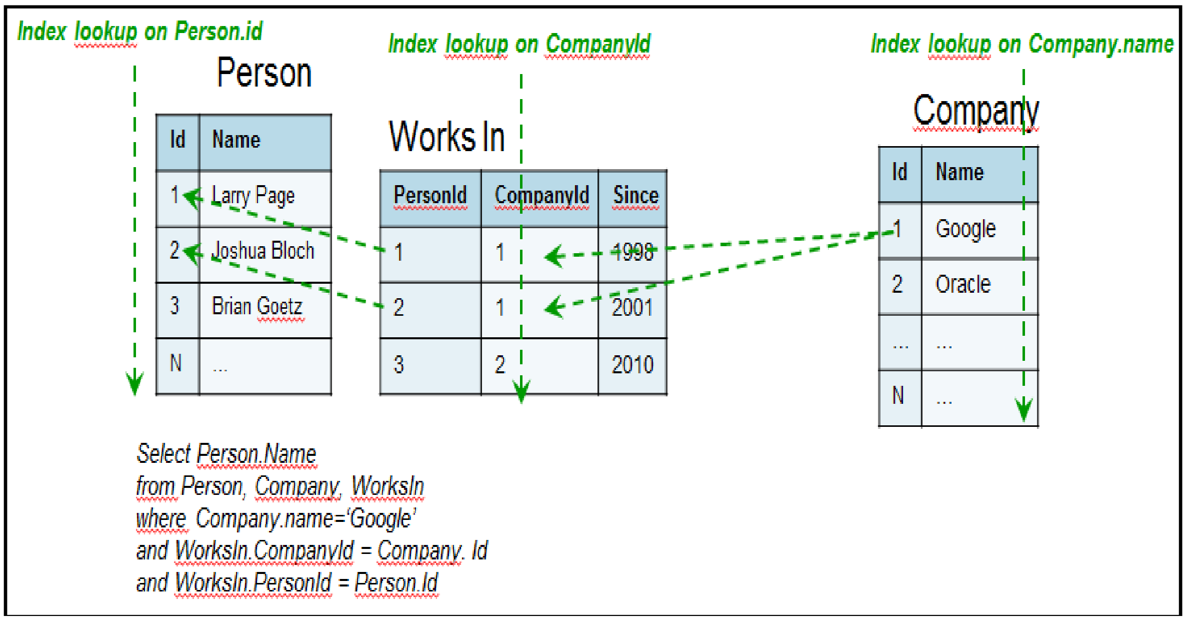
Some argue that most data is inherently a graph, and that all data can be stored as a graph. Using graphs to store data not only allows for a dynamic schema, but also provides representations of data not previously possible. The ability overlay different graphs (Ex. social, temporal, and spacial) on data extends the functionality of querying data. In Managing and Mining Graph Data

VII. Graph Databases Vs Relational Databases

A graph database is a good fit for exploring data that are structured like a graph, in particular when relationships between items are significant. By contrast relational databases are well suited to find All- like queries.

To make it clear let’s consider a example where we try to store companies, people who try to work for them, and how long they have been working there. For example- we are trying to find all people working at Google.

When relational model we execute the following query which could require probably 3 index lookups corresponding to the foreign keys in the model.


Index lookup on Person.id
Index lookup on CompanyId
Index lookup on Company.name
Person
Works In
Company

Person

Id	Name
1	Larry Page
2	Joshua Bloch
3	Brian Goetz
N	...

Works In

PersonId	CompanyId	Since
1	1	1998
2	1	2001
3	2	2010

Company

Id	Name
1	Google
2	Oracle
...	...
N	...

```
Select Person.Name
from Person, Company, WorksIn
where Company.name='Google'
and WorksIn.CompanyId = Company. Id
and WorksIn.PersonId = Person.Id
```

Figure 2

In case of graph the query will require 1 index lookup ,then will traverse relationships by dereferencing physical pointers directly.

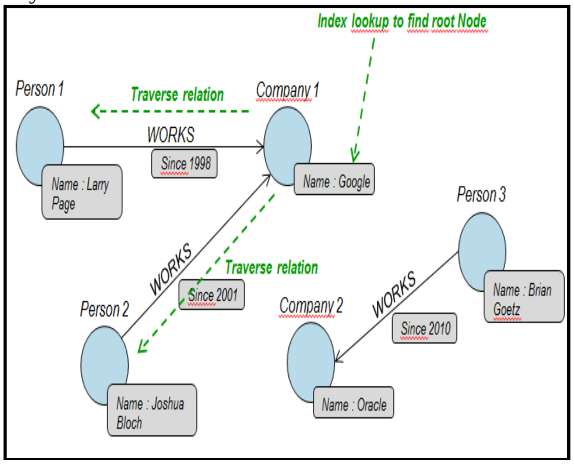


Figure 3

Comparing the performance of relational databases on graph analytics. Here a leading graph database system (Neo4j) and three relational databases: a row-oriented database (MySQL), a column-oriented database (Vertica), and a main-memory database (VoltDB) are compared. T wo queries, PageRank and Shortest Paths, on each of these systems.

Considering two datasets from the Stanford large network dataset collection:

- A Facebook dataset having 4K nodes and 88K edges, and
- A Twitter dataset having 81K nodes and 1.8M edges.

Figures 4(a) and 4(b) show the result (below).

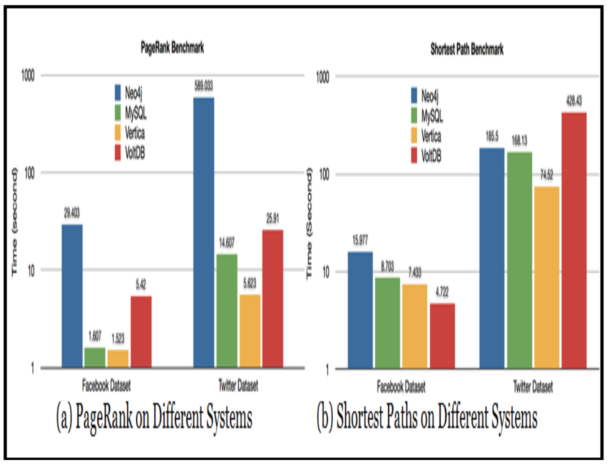


Figure 4

We can see that relational databases outperform Neo4j on PageRank by up to two orders of magnitude. This is because PageRank involves full scanning and joining of the nodes and edges table, something that relational databases are very good at doing. Finding Shortest Paths involves starting from a source node and successively exploring its outgoing edges, a very different access pattern from PageRank. Still, we see from Figure 4(b) that relational databases match or outperform Neo4j in most cases. In fact, Vertica is more than twice faster than Neo4j. The only exception is VoltDB over Twitter dataset.

VIII. 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 finding sub- graphs of a graph. For most data, the number of links or relationships is too large to efficiently compute an optimal partition; therefore most databases use random partitioning. Horton(2010) 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(2010)[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.

IX. 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 o, or on top of, existing key-value stores allowing for greater scalability and partitioning of graph nodes. Vertex DB(2009) is a key-value disk store that makes use of Tokyo Cabinet. The graph database focuses on a vertex graph with added support for automatic garbage collection. Cloud Graph(2010) 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. Cloud Graph has also created its own graph query language (GQL).Redis Graph(2010) 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(2011] 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 benefitt of in-memory key value storage can be seen with increased performance

X. CONCLUSION

Through this paper we have tried to cover about the brief overview about the graph databases. The databases such as key value database, distributed graph databases. These databases can be used for web development using php and Neosql which can be dealt in future.

XI. 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] Innitegraph. http://infinitegraph.com/.
[11] Neo4j. http://www.neo4j.org/.
[12] Orientdb. http://www.orientdb.org/.
[13] Phoebus. https://github.com/xslogic/phoebus.
[14] redis graph. https://github.com/tblobaum/redis-graph.
[15] Sones. http://www.dekorte.com/projects/opensource/vertexdb/.
[16] Trinity. http://research.microsoft.com/en-us/projects/trinity/.
[17] Vertexdb. http://www.dekorte.com/projects/opensource/vertexdb/.