11-28-2017

The Graph Database: Jack of All Trades or Just Not SQL?

George F. Hurlburt
STEMCorp

Maria R. Lee
Shih Chien University

George K. Thiruvathukal
Loyola University Chicago, gkt@cs.luc.edu

The Graph Database
Jack of All Trades or Just Not SQL?

George F. Hurlburt, *STEMCorp*
George K. Thiruvathukal, *Loyola University Chicago*
Maria R. Lee, *Shih Chien University, Taiwan*

The notion of graph reasoning is not new. The heretofore curious and obscure branch of mathematics, graph theory, extends to Leonhard Euler in the 18th century.[1] The application of graph theory to all manner of networks is quite new, however. Graph theory has exploded, largely due to the existence of the Internet.

An unintended consequence of the Internet was to reveal how interconnected the world has always been. While the term "network" used to refer almost exclusively to electronic transmission systems, suddenly, networks are everywhere. They are social. They are supply chains. They are part and parcel of ecosystems. They are food chains. They transport people, signals, and energy. They exist throughout biology, physics, chemistry, and economics. They are in our DNA, our software, and our malware. Even dark matter now appears to be interconnected, almost like gigantic intergalactic neurons. Albert-László Barabási's *Linked: How Everything Is Connected to Everything Else* defines the value proposition behind the rapidly emerging field of network science.[2]

Unlike the Industrial Age—which comprised lots of immutable physical laws governing highly predictable, linear, and deterministic behavior—today's networked world is nonlinear and seemingly messy by comparison. Cause and effect in the networked world are often decoupled, making interpretation difficult without the right mathematical tools. In a networked world, reductionism fails, given that the whole often exceeds the sum of the parts. Nonetheless, applied graph theory leads to quantitative sense-making in an otherwise seemingly senseless world—a world increasingly dominated by data that are instantaneous, enormous, and interconnected in innumerable ways. In such a volatile world, where

 Published by the IEEE Computer Society

even things share massive data, the graph database is a logical fallout and a suitable alternative to the venerable relational database management system (RDBMS).

The Relational Legacy

The long-favored RDBMS is an artifact of the foregone linear age. Relational algebra, the mathematics underlying the RDBMS, grew out of a need to efficiently compress data during the 1960s, when storage was both limited and very expensive.[3] The RDBMS still serves well in transaction-rich, process-stable environments in which relationships can be expressed as one-to-one, one-to-many, or many-to-one. For example, huge credit-card processing systems operate reliably from large-scale 24/7/365 RDBMS installations. For this reason, and because RDBMS technology is good for data aggregation, this technology will not be going away. Much like television did not replace radio, the RDBMS has a definite niche. When predefined transactions become less prevalent or process dynamics vary frequently, the RDBMS becomes exceedingly costly to document and manipulate. When the preponderance of relationships become many-to-many, RDBMS performance takes a nosedive. Moreover, the RDBMS schema, as a formal means of defining entity relationships, is typically inflexible, requiring high maintenance to effect the most minute change.

As an interesting aside, although RDBMS became the de facto standard for databases, the 1960s also produced early database technology similar to graph databases. The hierarchical model was created at IBM to represent tree-structured relationships, in which individual records are arranged in a treelike fashion (an idea that is mimicked with foreign keys in RDBMS today and enforced with triggers). Similarly, the network model of the late 1960s was an early attempt to model objects and their relationships—an idea that would re-emerge in the 1980s with object-oriented databases. Thus, the notion of graph databases can be thought of as a more modern rendition of these nascent attempts to build more tree- or graph-like databases combined with the advances of the web era, wherein unstructured (or weakly structured) data (for example, in JSON) can be used to represent node and edge data (and metadata).

Enter the Graph Database

By contrast, the graph database is built around the notion of efficiently managing many-to-many, property-laden relationships that coexist in highly dynamic environments. Sporting impressive performance numbers for the many-to-many relationships common to networks, the graph database becomes an ideal way to represent and analyze complex nonlinear networks. It has the drawback, however, of being inefficient with end-to-end transaction processing and rapid associative summarization as its relational predecessor.

Graphs are expressed in node-arc-node (subject-predicate-object) triples. This notion of a graph is fundamentally straightforward. Nodes generally represent physical or conceptual objects, typically associated with objects as represented in a programming language. Nodes can typically have one or many descriptive properties ascribed to them. Arcs, or edges, represent metaphysical constructs that connect or create relationships between nodes or properties.[4] In some graph representations, properties can also be assigned to arcs.

Thus, the assertion that "Jack knows Jill" is a simple triple expressing a relationship between two nodes—in this case, human beings. Other triple assertions might also apply: "Jack uses a pail," "Jill has thirst," "Jack carries the pail," "a hill leads to water," "Jack climbs the hill," "Jill climbs the hill," and so on. Many graph databases go a step further and allow the attachment of properties to both nodes and relationships. Thus, Jack can take the properties such as "male," "age 19," "loving," or "physically fit," while the pail can be ascribed the properties "used" or perhaps "leaky." The "Jack uses a pail" relationship might also have an assigned property to better describe what the use of a pail can be.

The graph database, a popular variant of the NoSQL ("not only SQL") database, has grown as an effective tool for representing dynamic network-related relationships. Unlike the simple Jack and Jill nursery rhyme relationships, which only entail a few triples, significant graphs can easily grow to many millions, or even billions, of triples. To accommodate databases of this size, specialized supportive and performance-enhancing hardware and algorithms have arrived on the market. The parallel-processer-based graphics processing unit (GPU) also offers hardware

alternatives to accelerate large-scale graph processing.[5] Some firms, such as Cray,[6] have developed specially configured supercomputers to digest and return rapid results from massively scaled graphs.

The graph database, while still in its relative infancy, shows great promise for traversing complex paths to establish the linkages and influences that momentarily connect cause to effect via a chain of events. Because time is a factor, and nodes come and go over time, the resulting cause and effect relationships themselves are often fleeting. As such, the graph database offers the tantalizing ability to understand a growing myriad of network behaviors both qualitatively and quantitatively.

Qualitatively Speaking

The qualitative aspect of a graph database comes into play when data are queried, particularly when based on like properties common to numerous nodes or arcs. This, however, necessitates some salient precautions. The quality of the data in any graph depends on the quality of the relationships. Thus, while not initially requiring the demanding rigor of the RDBMS-related entity-relationship diagrams (ERD), the graph database nonetheless requires some degree of effective front-end modeling.

The good news is that, unlike ERDs, simple graph models are visual, flexible, and accommodate changes on the fly. If the relationships are straightforward, and their number is limited, the data can be self-describing. A simple graph model shows the family of overarching relationships between nodes in a given graph environment. As the data grows to large collections of instances, such as the number of sensors in a burgeoning Internet of Things (IoT), the graph model becomes necessary to sort out the many varied nodal properties and meta-relationship types these nodes and properties might possess.

The bad news is that as mission criticality and scale grow, the requisite modeling can grow to the proportion of full-blown semantic ontology. This sophisticated level of modeling often requires at least as much or more design forethought than an ERD.

In either case, poorly thought-through relationships contribute to poorly defined graph environments. For example, defining a node by a foreign key from a former RDBMS does not yield a great deal of meaningful information in a graph data environment. Rather, the declarations of all nodes, properties, and relationships need to be explicit and should conform to a generalized pattern defined via the specific graph model. To alleviate storage consistency concerns, many graph databases do support the Atomic, Consistent, Isolated, and Durable (ACID) consistency model, which is a spin-off storage-locking scheme from RDBMS technology (bit.ly/2gfNjQu).

Scale introduces yet another qualitative concern. It is easy for graph data to grow rapidly as more and more instances are brought to bear. This relates to the classic "how much is enough" dilemma when building any model. The problem is heightened in the graph database environment, because each graph instantiation is typically isolated on a single graph server. As the graph grows in scale, the related query complexity grows as well. Fortunately, graphs can be intelligently reduced to more salient subgraphs that can be better managed, queried, and understood. This reinforces the growing practice of persisting data in a relational or appropriate nongraph NoSQL environment. The choice of an appropriate data persistence tool often depends on tolerance for the amount of structure or lack thereof in the data. From this larger corpus, subgraphs (database "views") can be intelligently isolated for further analysis of dynamics as a specific subgraph drawn from a larger graph. As scale increases, modern algorithms can assist in the subgraph mining process.[7]

Ironically, SQL might prove to be a useful means to permit building pattern-based relationships from simple taxonomy-based arrays of descriptive data stored as relational data. For example, we might wish to view an IoT graph phenomenon from its supply chain perspective to examine the efficiency of material flow to meet an operational IoT requirement in a given timeframe. At another time, however, isolating actual IoT operations as governed by interaction among the nodes and their properties might be important to fully understand mission effectiveness. Although the RDBMS environment cannot support the requisite number of many-to-many relationships entailed in the graph, it can nonetheless serve as a storehouse for the data necessary to efficiently generate the requisite model (both role and rule)-based relationships. In such

fashion, if accompanied by a well-designed user interface, data entry becomes more straightforward and does not require graph database language expertise on top of subject matter expertise.

This notion of hybrid systems also has the distinct advantage of permitting some level of seamless coupling between various graph databases. Although some query languages, such as Neo4j's Cypher[8] language, are becoming popular, there is little semantic commonality among the various graph languages in use and their rules of syntax. For example, higher volume graph databases rely on stylized variations of RDF[9] to enumerate their triples. Thus, the notion of sharing data between various graph databases is a function of the user's tolerance for expressing the same triples in differing syntactical frameworks. Needless to say, if cross graph database sharing is required, this concern could burden a beleaguered subject matter expert who must now enter the data as well as master new methods of its expression.

Numerically Speaking

Although not fully incorporated in most commercial graph databases, the qualitative promise will come as metric algorithms, derived from graph theory, are applied as analytical tools. Search algorithms, based on path traversal—although already an important family of graph database algorithms—are just the beginning of a wide array of metrics, now extending beyond mere networks to far more daunting networks of networks.[10] Although research is still increasing our understanding of the complex characteristics of networks of networks, it is becoming evident that these relationships can be demonstrated mathematically using graph theory. In time, built-in mathematical functions, residing in graph databases, could add a level of depth to truly understanding and quantifying graph relationships. If it is advantageous to eventually design networks of all types to perform useful purposes, the quantitative aspect of this design cannot be overlooked. For example, as large-scale systems embracing massively embedded software are already known to form complex adaptive relationships, it is doubtful whether large software systems of systems can be effectively evaluated without some quantitative expression of their operations.

What Lies Ahead

This special issue explores the emergent world of graph databases in increasing depth. It starts with four articles that establish the dimensions of graph data modeling. We begin with Zuopeng Zhang's "Graph Databases for Knowledge Management," which differentiates between an RDBMS ERD and a graph data model (GDM). In "Modeling Graph Database Schema," Noa Roy-Hubara, Lior Rokach, Bracha Shapira, and Peretz Shoval then demonstrate a technique to map the ERD to the GDM. As graph databases grow to support full-blown *enterprise knowledge graphs*, appropriate graph modeling presents elevated sophistication and rigor. Jans Aasman explores these challenges in "Transmuting Information to Knowledge with an Enterprise Knowledge Graph." Finally, in a specialized use case, "Modeling XACML Security Policies Using Graph Databases," Fidel Paniagua Diez, Amrutha Chikkanayakanahalli Vasu, Diego Suárez Touceda, and José María Sierra Cámara demonstrate a method to efficiently store eXtensible Access Control Markup Language (XACML) security policies using an optimized graph model.

Our final article deals with enhancing graph database performance. Here, we turn our focus to hardware specifically focused on optimal graph database performance. "High-Performance with an In-GPU Graph Database Cache," by Shin Morishima and Hiroki Matsutani, examines GPU efficiency when distributed and optimized for performance.

We commend these excellent articles to you for the awareness they build regarding the state of the art in graph databases. We look forward to a growing trend in such insightful articles as the still young graph database industry continues to mature.

References

1. B. Bollobas, *Modern Graph Theory*, Springer, 1998.
2. A.L. Barabási, *Linked: The New Science of Networks*, 1st ed., Perseus Books Group, 2002.
3. D. Kronke and K. Dolan, *Database Processing*, 3rd ed., Scientific Research Associates, 1998, pp. 19–20.
4. N. Jatana et al., "A Survey and Comparison of Relational and Non-Relational Database," *Int'l J. Eng., Research & Technology*, vol. 1, no. 6, 2012, pp. 1–5.

5. J.D. Owens et al., "GPU Computing," *Proc. IEEE*, vol. 96, no. 5, 2008, pp. 879–899.
6. S.R. Sukumar and N. Bond, "Mining Large Heterogeneous Graphs Using Cray's Urika," *Proc. ORNL Computational Data Analytics Workshop*, 2013.
7. J. Huan et al., "Spin: Mining Maximal Frequent Subgraphs from Graph Databases," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, 2004, pp. 581–586.
8. I. Robinson, J. Webber, and E. Eifrenm, *Graph Databases, New Opportunities for Connected Data*, 2nd ed., O'Riley Media, 2015.
9. R. Angles and G. Gutierrez, "Querying RDF Data from a Graph Database Perspective," *Proc. European Semantic Web Conf.*, 2005, pp. 346–360.
10. S. Boccaletti et al., "The Structure and Dynamics of Multilayer Networks," *Physics Reports,* vol. 544, no. 1, 2014, pp. 1–122.

George F. Hurlburt *is the chief scientist at STEMCorp, a nonprofit corporation that works in the public sector to further economic development via adoption of network science to advance autonomous technologies as useful tools for human use. Contact him at ghurlburt@change-index.com.*

George K. Thiruvathukal *is a full professor of computer science at Loyola University Chicago and visiting computer scientist at Argonne National Laboratory. His research interests include parallel and distributed computing, software engineering, history of computing, and interdisciplinary computing applications. Contact him at gkt@cs.luc.edu.*

Maria R. Lee *is a full professor of information technology and management at Shih Chien University Taiwan, and a visiting professor at the Advanced Data Analytics (ADA) Lab at SooChow University, China. Her research interests include big data analytics, e-commerce, and artificial intelligence. Lee received a PhD in computer science and engineering from the University of New South Wales, Australia. Contact her at maria.lee@g2.usc.edu.tw.*