research

CHECKLIST REPORT 2020

Seven Steps to Complementing Your Analytics Arsenal with a Graph Database

By David Loshin

Sponsored by:

neo4j

tdwi | TRANSFORMING DATA WITH INTELLIGENCE™

OCTOBER 2020

TDWI CHECKLIST REPORT

Seven Steps to Complementing Your Analytics Arsenal with a Graph Database

By David Loshin

TABLE OF CONTENTS

tdwi

Transforming Data With Intelligence™

555 S. Renton Village Place, Ste. 700
Renton, WA 98057-3295

T 425.277.9126
F 425.687.2842
E info@tdwi.org

tdwi.org

FOREWORD

We live in a connected world. Thousands of companies execute transactions involving millions of items that are moved through complex supply chains. Individuals communicate and share ideas via social networks. Organizations are comprised of collections of connected communities working together. Smart mobile devices are everywhere, and many industries are implementing a wide array of sensors, controllers, and similar devices across different domains that are continuously broadcasting streams of data that contain important information about connections.

Modern analytics calls for understanding the connections or links between things, not just the characteristics of the things themselves. Recognizing the existence of connected networks is an opportunity for graph database applications. Graph databases can ingest and represent the qualitative characteristics of both the entities and the relationships or links among them.

Graph-shaped business problems are all around us, and with some practice they become easy to recognize. For example, a global bank wants to implement a customer relationship management system that provides information differentiating individual accounts from corporate accounts and visibility into each customer's banking relationships (both by account and by branch location) and acquired products. At the same time, the bank must comply with data sovereignty laws about where customer data may be managed, data privacy protection laws that vary by geopolitical region, and disclosure laws for taxation purposes. These dependencies qualify customers by account type but also relate customers to products, customers to locations, locations to laws, and laws to compliance requirements.

A simple implementation can provide reporting and analysis using a relational database management system. However, the potential complexity of the relationships and the implications of hidden connections pose risks that cannot be remediated with simple SQL queries. A graph database can expose connections among individuals as well as between individual and corporate accounts. It can help you examine behavior patterns among related entities to identify compliance risks or new business opportunities that are difficult to find using traditional means. Complementing traditional reporting and analytics frameworks with additional analysis techniques can add value.

This checklist looks at how data-driven organizations will benefit from expanding their analytics arsenal to include graph databases.

1 UNDERSTAND THE DIFFERENCES BETWEEN A GRAPH DATABASE AND A RELATIONAL DATABASE

People typically assume that the word "database" refers to the standard framework for representing structured data configured using rows and columns—the relational database management system (RDBMS). Although the RDBMS has become the go-to platform for typical transaction or operational processing applications, it is not the only framework for managing data. That's good news because much has changed about the shape, size, and sources of data since the original relational model was conceived.

Different business problems are often better addressed using different technology platforms. For example, consider graph databases. A *graph* is a mathematical abstraction used to represent the ways different items are connected in a network. Formally, a graph is a set of vertices (or nodes) connected by a set of edges (or links). A *graph database* is a NoSQL data management application designed to model, represent, and answer questions about connected entities and their relationships.

A native graph database (as opposed to a database with a graph API) has a storage model designed to represent vertices and edges and optimized to also capture their associated attributes and properties. The storage model is optimized for traversal of the graph without using an index, allowing applications to apply algorithms for graph traversal (shortest path, connected components, betweenness, etc.) that can surface different types of insights about entities based on their characteristics and the patterns inherent within the network.

The RDBMS excels for transaction and operational processing and most analytics applications. An RDBMS could also be used to represent a graph—the relational structure can be manipulated to address problems that analyze the nature and characteristics of relationships among a variety of entities.

However, data modelers and application developers struggle to implement graph analyses using the RDBMS framework. An RDBMS is not naturally adaptable to the graph paradigm. In an RDBMS, graph data models are complex, application developers and analysts alike struggle to write SQL statements that require multiple nested JOINs just to perform logically simple relationship queries, and query performance is notoriously slow.

Understanding the difference between these two types of databases allows you to make informed decisions about which type of platform to use for your graph-shaped business problems.

2 LEARN TO IDENTIFY "GRAPHY" PROBLEMS

Typical business analytics questions focus on counting or aggregating things—the number of products sold over a time period, the number of newly acquired customers, the average time for certain parts to be delivered, etc. However, there are certain types of business problems involving relationships and the nature of connectivity that are better suited to a graph database solution.

We can call these problems "graphy" problems, and all graphy problems exhibit some key characteristics:

- They all involve blending data about entities and their relationships
- They are more easily answered with queries that find and highlight relationship patterns
- They sometimes take advantage of graph analytics algorithms to reveal interesting and relevant patterns inherent in the network

Graph analytics reveals insights by analyzing a variety of entities and the connections that link them. Importantly, both the entities and the connections that link them are first-class objects and have properties that reflect the characteristics of the environment being modeled. Examples include:

- Exposing fraud networks by analyzing communication patterns among suspicious actors
- Identifying individuals in a social network who are key influencers among desired customer segments
- Reviewing medical provider referrals to facilitate effective continuation of care among patients within certain regions
- Analyzing shipping deliveries to identify home-based businesses for marketing purposes
- Scanning real estate transaction patterns to identify speculation cohorts targeting specific areas

These and other graphy problems share fundamental similarities, such as:

- A desire to document different types of relationships among the different entities
- A desire to surface interesting patterns such as frequent customer-product purchase combinations
- A need to model and represent relationships that have varying depth, such as logs of sequential tasks or friends of friends
- Scenarios involving self-referencing among similar entities, such as when modeling organization charts
- Interest in analyzing or discovering transitive relationships, such as the reach of influencers in a social network
- A desire to evaluate characteristics of a collection of interacting entities, such as the time it takes for messages to propagate across a social network

Once you know how to identify graphy problems, you are better positioned to take advantage of graph databases designed for high-performance graph analyses.

3 GET ACCUSTOMED TO THINKING ABOUT HOW THINGS ARE CONNECTED

The constant use of a relational database management system has focused data consumers' thought processes in terms of conventional database structure. Our data management professionals default to modeling using tables, columns, and rows. They conceptualize connectivity among entities in terms of foreign key relationships. They capture connection properties in associative or lookup tables that are inserted into a relational model.

By constantly looking at how to shoehorn every problem into an RDBMS, we have stopped thinking about the fundamental nature of the problems we are trying to solve. When it comes to graph-shaped problems, it is much better to think about the entities and the connections themselves instead of how they would be represented in tables, columns, and rows.

Get used to thinking about how things are connected, the different ways things can be connected, and what those connections actually mean. For example, earlier we referred to assessing the "reach of influencers in a social network." We can look at this statement in two ways. From a business perspective, identifying influencer reach focuses on the breadth of an individual's sphere of influence along with measuring how powerful that influencer's messaging remains among those within that sphere. From a technical perspective, the statement focuses on a breadth-oriented traversal of the graph: how many individuals are directly connected, how many are connected via varying distances, and the efficiency of all the individuals in each path in conveying a message.

There are a wealth of practical computer science algorithms that are intended to be used to address common questions about finding paths, seeking out strongly connected communities, breadth and power of influence, and distance between specific entities. Some native graph database vendors have incorporated implementations of some of these algorithms into their products. These algorithms can be used to augment the typical types of aggregation analytics that are performed using a traditional RDBMS.

Acclimate to thinking about how things are connected. This directs your perception of the place of a graph database within the organization's analytics strategy, allows you to reframe business problems that are best suited to graph databases, and inspires integrating a graph database into the enterprise.

4 ADD A GRAPH DATABASE TO YOUR ANALYTICS INFRASTRUCTURE

Even with graph databases that can fully support ACID-compliant transactions, a graph database is not likely to unseat a traditional RDBMS from its roles in transactional, operational, or even analytical processing. However, the more you consider organizational analytics demands, the more graph-shaped problems you begin to uncover.

Adding a graph database to your analytics infrastructure provides an additional dimension of analytics depth that your analysts and data scientists can leverage. However, integrating a graph database has some operational implications, especially when the graph database is primarily employed for analyzing data sets that are combined from a variety of sources. You must consider how you will coordinate data interchange between conventional database systems and a graph database as well as enable the downstream data consumers. Factors include:

- **POPULATING THE GRAPH DATABASE.** In some cases, analyses performed using a graph database can be rapidly spun up by extracting data from the relational systems, loading the data into the graph database, and executing the analyses.
- **SYNCHRONIZING THE GRAPH DATABASE.** In other cases, the volume of data in the graph is too large to efficiently reload on a regular basis. In this case, there must be some type of change data capture that can be used to maintain coherence between the relational systems and the graph database.
- **ACCESS SERVICES.** Different analysts will have different strategies for engaging with the graph database. Some analyses can be encapsulated in a series of queries executed at the command line; others make use of user interfaces that display the results of queries. Still others access the graph via defined APIs. All these methods of data access should be supported by your graph database.
- **EXECUTING ANALYSES.** One potential benefit of using graph databases is that they come bundled with libraries of standard graph analytics algorithms that are frequently employed, such as finding paths, detecting communities, assessing measures of rank or influence, etc. These integrated algorithms simplify the analysts' ability to develop graph solutions.

Add a graph database to your analytics infrastructure. Augmenting the analytics environment with a graph database provides a functional means for integrating graph analytics as part of the overall analytics strategy.

5 LEVERAGE GRAPH VISUALIZATION TO MOTIVATE ACCEPTANCE

Business leaders and decision makers are avid consumers of information but sometimes need additional encouragement to adopt analytics results when they do not completely understand how those results are produced. Because graph databases differ from the conventional RDBMS, you may need to take additional steps to socialize their use and demonstrate their value.

Fortunately, good graph database platforms are coupled with a means for visualizing graphs, and when it comes to using a graph database, it is true that a picture is worth a thousand words. Use graph visualizations to demonstrate how a graph database represents connectivity. Individuals are more likely to understand the power of a graph database when confronted with a visual representation of the entities and the connections between them.

Recognize, though, that graph visualization is a double-edged sword. An interactive interface may allow a business analyst to execute queries, see the resulting subgraph, review the properties of specific nodes or edges, and drill down to expand the visual display. These are useful capabilities that provide insight into how graph databases work and the types of questions you can ask them.

However, as the size of the graph grows and as graph analytics becomes more sophisticated, it becomes more difficult to take advantage of a visualization interface—the number of nodes and edges becomes far too dense to be able to make sense out of a large, complex graph. In fact, in many production applications using graph databases, the need for visualization is diminished. There is no need to have a user interface; instead, analytics results can be summarized and conveyed in a more succinct way.

Leverage graph visualization to motivate acceptance among the business leaders in your organization. Visualizations are a good way to raise awareness and socialize the value proposition and should not be discounted as a critical step in graph database understanding and adoption. At the same time, engineer your applications to reduce your dependence on native graph visualization to tell the stories that emerge from graph analytics.

6 DIFFERENTIATE GRAPH-NATIVE SYSTEMS FROM GRAPH-ADAPTED APIS

Over the past few years, many database vendors have recognized the power of the graph paradigm, and a number of these vendors have attempted to present their users with a mechanism that allows their traditional database to be used like a graph database. As more people become accustomed to identifying graphy problems, more vendors will retrofit a graph database solution onto their existing platforms.

However, there are significant fundamental differences between what we would call a graph-native system and a database with a graph application programming interface (API) bolted onto it, most notably:

- **INFRASTRUCTURE.** The underlying storage model for a graph-native database is designed to support connectivity and management of relationships and is designed to scale performance as the graph data volume grows. This is not true of an API implementation layered on top of an RDBMS, which is limited to the conventional table/column/row data models.
- **PERFORMANCE.** The ability to optimize and then rapidly execute graph queries is rooted in the underlying storage architecture and processing model that is tightly coupled with a graph-native database. With an RDBMS infrastructure, the graph APIs may map simple graph queries into complex SQL queries or may demand data extraction and out-of-band execution using a high-performance engine such as MapReduce, both of which create artificial performance bottlenecks.
- **FLEXIBILITY.** An RDBMS has a strict predefined data schema into which a graph model must be wedged. A graph database allows for flexibility and adaptation as the understanding of the business scenarios and problems evolves.
- **AGILITY.** It is easy for analysts to get started with graph databases and with a fundamental knowledge of how a graph-native database works. Analysts can rapidly update and transform their schemas and models in alignment with the agile development methodology.

Understanding these fundamental differences will inform your decision about the technical approach to adopt when augmenting your analytics arsenal with graph analytics capabilities. If you plan to scale up your use of the graph database approach, adopt a graph-native system.

7 ADOPT AN AGILE TECHNIQUE IN DEVELOPING GRAPH ANALYTICS SOLUTIONS

We must emphasize that there are significant benefits in employing agile techniques in developing graph analytics solutions. In contrast to the traditional RDBMS's dependence on a predefined schema, graph databases have much greater flexibility in defining and dynamically refactoring the graph data models that are used to develop an analytics application.

Graph databases typically are NoSQL databases that provide greater flexibility in defining and then refining your underlying models. This provides a number of benefits, such as:

- **SHORT RUNWAY.** It is easy to develop an initial graph schema and immediately start to analyze relationships and connections.
- **RAPID REFINEMENT.** With some experience with a particular graphy problem, one often finds that the original model is not exactly right for the kinds of analyses to be performed. Because the modeling is flexible, it is easy to abandon a graph model and replace it with a better one.
- **DYNAMIC FLEXIBILITY.** Flexibile annotation of properties to entities and connections provides flexibility in revising the analytical approach. This means that there may be cases where what was originally considered to be an edge between two nodes can be remodeled as a node, which then allows you to model relationships between relationships.
- **DIFFUSION ACROSS APPLICATIONS.** Different graph algorithms can be applied to the same graph, allowing interesting patterns to be discovered. Those discovered patterns can then inform the revision of the underlying model.
- **COLLABORATION.** Different analysts can begin at the same starting point and develop their own graph analytics solutions.

The flexibility in developing graph models using a graph-native database dovetails nicely with the agile development methodology. It stimulates iterative refinements and improvements in the model that allow for greater innovation when crafting a solution while allowing different analysts to leverage the same data in different ways to solve different types of business problems.

AFTERWORD

Although this checklist has made recommendations about ways to add a graph database to your analytics environment, what we have really examined is fundamentally changing the ways your analysts and data scientists employ different technologies such as graph databases to incorporate additional analytics perspectives into the enterprise.

The process begins with differentiation—not just of the graph database technology but understanding the types of problems that are more easily addressed with graph databases as well as retraining your analytics thought process to think about how things are connected and what you can learn from analyzing relationships.

That being said, when evaluating graph database tools, look for products that allow you to integrate graph database algorithms and analytics while simultaneously maintaining flexibility and interoperability to ensure customer/analyst/data scientist enablement. Look for a native graph database that provides the underlying data structures and computational models to effectively scale as the sizes of your graphs increase.

Realize that a visual interface is an effective add-on that can help socialize graph analytics, and be willing to graduate to more sophisticated analyses that no longer depend on the visualizations to convey the story. Finally, look for graph databases that facilitate rapid development yet remain flexible for a variety of users. These steps to including graph databases as an additional tool in your analytics arsenal will allow your analysts to develop more diverse analytics stories.

ABOUT OUR SPONSOR

Neo4j is a leader in graph database technology. As a widely deployed graph database, we help global brands – including Comcast, NASA, UBS, and Volvo – to reveal and predict how people, processes and systems are interrelated. Using this relationships-first approach, applications built using Neo4j tackle connected data challenges such as analytics and artificial intelligence, fraud detection, real-time recommendations, and knowledge graphs.

Find out more at neo4j.com.

ABOUT TDWI RESEARCH

TDWI Research provides industry-leading research and advice for data and analytics professionals worldwide. TDWI Research focuses on modern data management, analytics, and data science approaches and teams up with industry thought leaders and practitioners to deliver both broad and deep understanding of business and technical challenges surrounding the deployment and use of data and analytics. TDWI Research offers in-depth research reports, commentary, assessments, inquiry services, and topical conferences as well as strategic planning services to user and vendor organizations.

ABOUT THE AUTHOR

David Loshin, president of Knowledge Integrity, Inc., (www.knowledge-integrity.com), is a recognized thought leader, TDWI affiliate analyst, and expert consultant in the areas of data management and business intelligence. David is a prolific author on topics related to business intelligence best practices. He has written numerous books and papers on data management, including *Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL, and Graph* and *The Practitioner's Guide to Data Quality Improvement*, with additional content provided at www.dataqualitybook.com. David is a frequent invited speaker at conferences, online seminars, and sponsored websites and channels including TechTarget and The Bloor Group. David is also the program director for the Master of Information Management (MIM) program at University of Maryland's College of Information Studies.

David can be reached at loshin@knowledge-integrity.com.

ABOUT TDWI CHECKLIST REPORTS

TDWI Checklist Reports provide an overview of success factors for a specific project in business intelligence, data warehousing, analytics, or a related data management discipline. Companies may use this overview to get organized before beginning a project or to identify goals and areas of improvement for current projects.