A New Paradigm for Data Processing:

GRAPH DATABASE

TABLE OF CONTENTS

INTRODUCTION

Companies have many platforms at their disposal for processing big data. However, it's still far from easy to efficiently examine highly connected data for real-time recommendation engines, personalization, fraud detection, cyber security, master data management, social networking or 360-degree customer views. These use cases often span hundreds of millions of data points, each with its own relationships.

Since the speed with which distributed big data can be analyzed is often critical to these kinds of applications, many developers are turning to graph database technology. Emerging solutions look at the complex and dynamic connections within and across data sets to deliver new insights and intelligence.

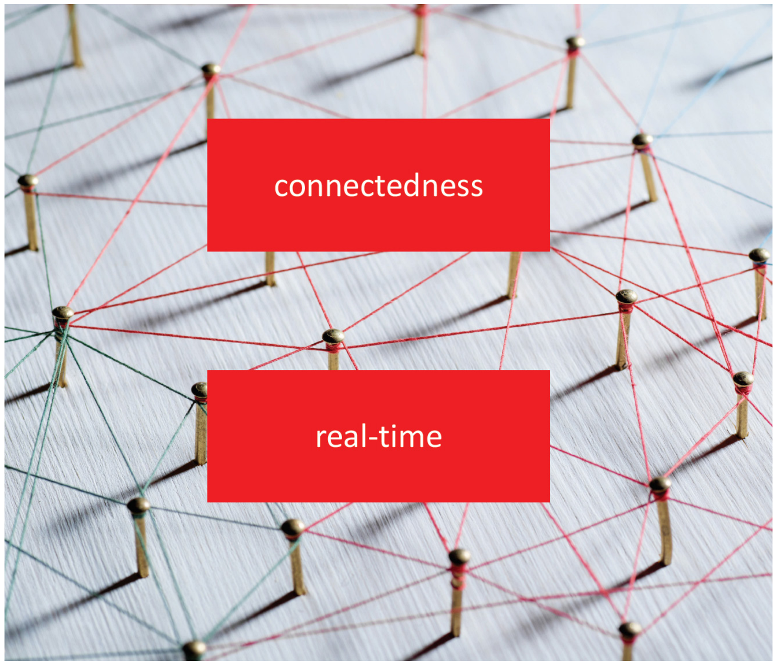
In this eBook, we'll discuss how the database ecosystem has evolved to open up interesting possibilities for exploring connected data in real time, and we'll detail one graph database in particular—RedisGraph from Redis Labs.

USER CASE: REAL-TIME FRAUD DETECTION

First, we'll illustrate the real-world value of a graph database in the context of one specific use case. Fraud detection is a data application that's critical for all financial institutions. Of course, it's ideally suited for graph technology because the faster these organizations can identify and uncover fraud risks, the fewer losses they'll experience.

A common threat banks face is fraud rings. These criminals steal data (e.g. social security numbers, addresses and more) from real people and create fictional personas out of their personal information. They start using this information to do run-of-the-mill banking that looks normal from the bank's perspective. Once they've gained the trust of these banks, the fraudulent "customers" take out loans, open lines of credit, and start accruing more and more debt before disappearing with the money.

With a high performance, real-time graph database, developers could mitigate these risks by quickly spotting patterns that might have flown under the radar of old-school fraud detection systems. For instance, we could plot out first names and last names that were used with multiple addresses or social security numbers. This kind of detail would be hard to recognize without graph algorithms. However, graph processing requires incredible computing power to detect new types of fraud in real time, rather than after the fact. The good news is that faster solutions are entering the market to support use cases just like this one.



THE EVOLUTION OF GRAPH THEORY

To understand how these graph databases have evolved, let's briefly review some of the academic and industry advances that have both helped and hindered real-time graph processing.

There's no question that information is inherently connected, yet—although graph theory has been around for a long time—most have found the challenge of understanding data relationships at scale to be very computationally difficult. That's why the graph approaches attempted by previous generations of database were slow and memory-intensive, and failed to tap into the full potential of connected data.

The Adjacency List Approach

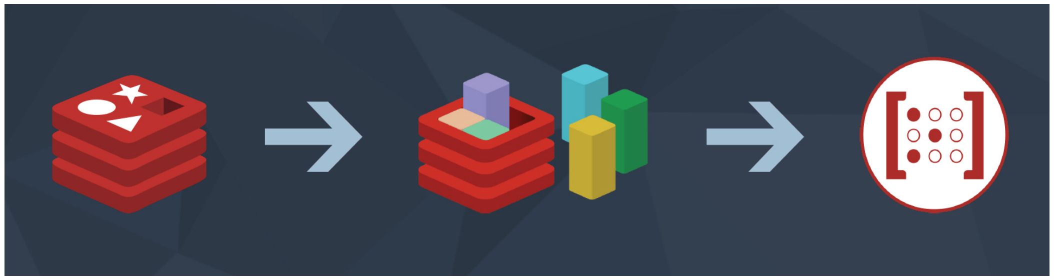
One common approach has been to use the relatively simple concept of adjacency lists to store information about both individual data points (the pins in this image), and the relationships (the strings) between those nodes. This works by creating a linked list for each node with all the other nodes it's connected to.

Unfortunately, it requires developers to write less efficient code for graph traversal operations, so problems arise when we need to explore those connections at a deeper level. The deeper we go, the faster adjacency lists become complex and time-consuming. This latency is the primary reason graph technology has been considered relatively exotic and out of reach until now. It's something most developers only tackled for forensic purposes after the fact, not for real-time analysis.

Math Algorithms & GraphBLAS

Inspired by the Basic Linear Algebra Subprograms (BLAS) of dense linear algebra, a project called GraphBLAS out of the University of Texas at Austin identified a core set of matrix-based graph operations that can run on sparse matrices. This new library of mathematical building blocks for graph algorithms is key because it solves the previous traversal issues with adjacency matrices, simplifies graph query computations and finally makes instant insight feasible for highly connected data sets.

Opening the Database Ecosystem



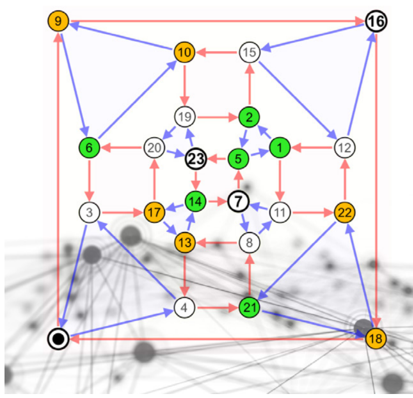
Laying more groundwork for graph technology, a new database category was created back in 2009 when Salvatore Sanfilippo open sourced his first alpha version of Redis – combining the best of in-memory, schema-less design with optimized data structures and a key value store. A few years ago, the community extended the power of Redis with Modules capabilities that allow developers to augment the platform with their own data types and commands. This was important because it turned Redis into to a true multi-model database that developers could adapt to their data rather than the other way around.

Eventually, our team at Redis Labs saw the opportunity to build a Redis Module that could understand and explore relationships between data points. The evolution of adjacency lists, the discoveries of GraphBLAS and the availability of Redis Modules were all crucial pieces of the puzzle.

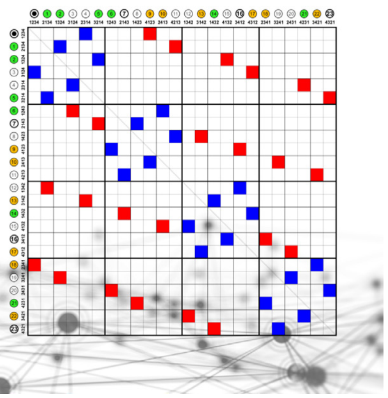
A NEW SOLUTION: REDIS GRAPH

RedisGraph uses two common data representation and traversal approaches (adjacency lists and adjacency matrices)—along with linear algebra algorithmics—to quickly and efficiently analyze highly connected data. The module's fundamental difference from other graph databases is its use of matrices and series of matrices to organize data points and their relationships. Using compressed matrix representation, RedisGraph's matrix traversal methodology overcomes typical graph performance and scale challenges. As a result, RedisGraph can be used to instantly answer complex questions and deliver contextual insights.

A



B



Under the Covers with Matrix Representation

To drill into exactly how this works, let's review image A, which shows nodes and relationships for a set of data, and image B, which represents that data in matrix form. Looking at the chart on the right, we can see all direct connections for node number five. For example, 5 and 1 are connected (1 is a column along the top and 5 is a row on the side), and since their relationship is a blue arrow, we've plotted it in the graph as a blue square. This approach basically gives us a simple matrix of 1s and 0s, with many more empty boxes than full.

In most use cases, adjacency matrices are not densely filled in with red and blue squares, but are sparse like the one above. A social network's matrix typically contains a vast array of empty space, which takes up countless bytes of memory. Because of this, standard adjacency matrices can only support very minimal graph analysis. To solve this problem, the RedisGraph team found a way to represent all data in a sparse matrix without having to store empty entries.

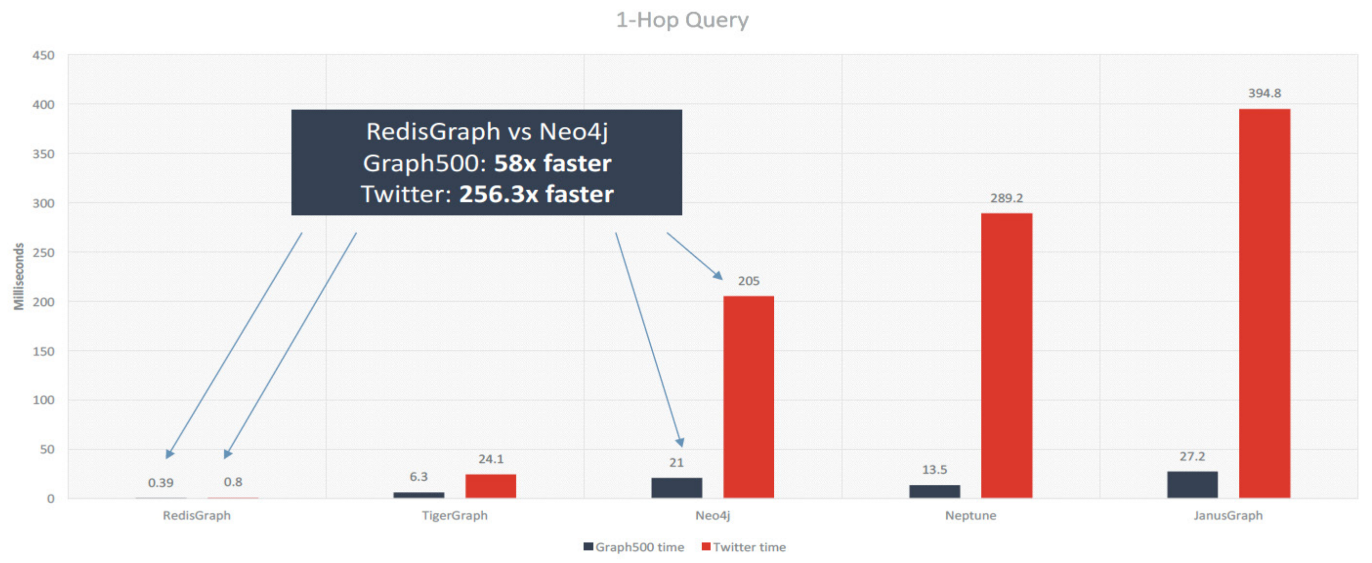
RedisGraph employs GraphBLAS to encode matrices in the Compressed Sparse Column (CSC) form, so the database only has to store non-zero matrix values. This optimizes the solution for large, sparse data sets and ensures memory efficiency. By reducing the amount of memory required, RedisGraph delivers cost savings while converting adjacency matrices into a functional format that can compute more than just basic operations.

Using Cypher for Algebraic Expression

Next, let's dive into how developers can use RedisGraph's matrix representation capabilities. The solution uses a fairly common query language called Cypher, which should be familiar to anyone with a SQL background—making it easy to create the algebraic expressions we need. In the sample code above, each parenthesis represents a node, and the arrows (the dash and greater than symbol) represent relationships. The query on the left gives us a fairly straightforward way to trigger the matrix arithmetic on the right in order to find "friends of a friend" whose ages are greater than 30.

```
MATCH
(src)-[:follows]->(f)-[:follows]->(foaf)
WHERE src.age > 30
RETURN foaf
```

Age_Filter
×
Follower
×
Follower



PERFORMANCE BENCHMARK:
600X BETTER LATENCY

RedisGraph can create over one million nodes in under half a second and form three million relationships in the same timeframe. Our early benchmarks have found that this makes it up to 600 times faster than other graph databases on the market today (as measured by a recent TigerGraph benchmark).

The two data sets used to produce these results were Graph500 (2.4 million nodes and 64 million edges) and Twitter (41.6 million nodes and 1.57 billion relationships). Unlike other databases with queries that took hundreds of milliseconds, RedisGraph maintained minimal latency for real-time application performance—completing all operations in less than one millisecond (.39 milliseconds for Graph500 and .8 milliseconds for Twitter).

What's more, while most graph processing requires a vast amount of memory just to calculate relationships, RedisGraph provides a 60-70% reduction in memory usage. Even with more complex queries, RedisGraph scales effortlessly froma both a performance and memory standpoint.

CONCLUSION

As detailed in this eBook, RedisGraph's efficiency and query simplification make it today's fastest graph database with the smallest memory footprint. This powerful technology enables applications to rapidly collect, process and analyze complex, connected data and understand data relationships with record-setting performance. As a result, you can build the richest possible experiences for your users.

RedisGraph also lowers data storage costs by up to 60% through the optimized graphical representation of large data sets. And by supporting the widely adopted Cypher query language, which automatically translates queries into linear algebraic expressions, it helps improve productivity so you can develop new applications even more quickly.

The RedisGraph module is currently offered with Redis Enterprise (a multi-model data platform available for deployment in the cloud, on-premise or in hybrid environments), or under Redis Labs' open source license.

For more information:

Learn more details about RedisGraph
Download the module at bit.ly/dl-redis-graph
Get demo code at bit.ly/dm-redis-graph
Read the benchmark at bit.ly/bm-redis-graph
Find out more from expert@redislabs.com

ABOUT REDIS LABS

Modern businesses depend on the power of real-time data. With **Redis Labs**, organizations deliver instant experiences in a highly reliable and scalable manner. Redis Labs is the home of **Redis**, the world's most popular in-memory database, and commercial provider of **Redis Enterprise** that delivers superior performance, matchless reliability and unparalleled flexibility for personalization, machine learning, IoT, search, eCommerce, social and metering solutions worldwide.

Redis Labs, ranked as a leader in top analyst reports on NoSQL, in-memory databases, operational databases, and database-as-a-service, is trusted by seven Fortune 10 companies, three of the four credit card issuers, three of the top five communication companies, three of the top five healthcare companies, six of the top eight technology companies, and four of the top seven retailers.

Redis has been voted the **most loved database**, rated the **most popular database container**, and the **#1 cloud database**.