Bloor

InBrief

Daniel Howard – Senior Researcher

Bloor MUTABLE AWARD 2018 ONE TO WATCH

redis

www.redis.com

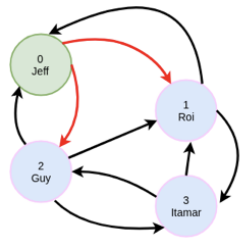
700 E El Camino Real Suite 250
Mountain View, CA 94040, USA
Tel: +1 415 930 9666
Email: info@redis.com

RedisGraph

The company

Redis is a leading in-memory database platform that uses a key-value storage paradigm. It is also the commercial provider of Redis Enterprise, an enhanced (and proprietary) version of Redis that provides additional functionality (primarily scale-out cluster support, and high availability) designed to make it better suited for the enterprise. This is available both on-premises and in the cloud (Amazon, Microsoft, Google, IBM, Heroku and others). Redis itself is often rated as one of – if not the – most popular NoSQL databases and Redis Enterprises has more than 8,500 paying customers.

Redis is a privately-held company backed by venture capital. It was founded in 2011 – originally as 'Garantia Data' – and is based in Mountain View, California. It has additional offices in London and Tel Aviv.



Node IDs	0	1	2	3
0		1	1	
1	1			1
2	1	1		1
3		1	1	

Figure 1 – A graph and corresponding adjacency matrix

What is it?

RedisGraph is the graph database module for Redis, where by "module" the company means functionality embedded into the product as opposed to something tacked on top. Version 1.0, available via a Docker container, was released as a preview in July 2018 and was made generally available in November 2018, along with Enterprise (multi-node) support. We are therefore commenting on a product in its very early stages and, as one might expect, it currently lacks some of the advanced features of more mature products (for example, strong consistency is not yet available). On the other hand, it has a major point of distinction from other graph products. This is that, apart from operating as part of the Redis platform, it represents and stores graphs as sparse adjacency matrices instead of adjacency lists. This enables much faster ingestion and query performance than would otherwise be the case.

The product itself is a property graph. Queries are written in (a subset of) the Cypher query language and, further, Redis is participating in the GQL project to create a standardised graph querying language. Consequently, we fully expect RedisGraph to support GQL when it arrives.

What does it do?

Generally speaking, there are two ways to store and represent graphs. The first of these is the adjacency list, which consists of a list of all the nodes in the graph it represents, each paired with the set of nodes with which it shares an edge (or, in other words, has a relationship with). This is effectively the industry standard. The second is the adjacency matrix, which represents its graph as a single, square matrix with one row and column for each node within the graph. Within this matrix, nonzero values indicate the presence of an edge between the nodes represented by the corresponding row and column. An example of an adjacency matrix, alongside the graph it represents, is shown in ***Figure 1***.

This method has several advantages in terms of performance. For example, determining whether two nodes share an edge is significantly faster when using matrix representation. Queries can be performed using direct mathematical operations such as matrix multiplication, which is often much faster than the traditional approach using adjacency lists. Performing a self-join, for instance, is simply a matter of multiplying a matrix by itself. Similarly, ingestion rates can be improved using adjacency matrices.

Historically, matrix representation has had two disadvantages. Firstly, it scales extremely poorly in terms of storage. An adjacency matrix

for even a moderately large graph can take up far more space in memory than is commercially viable. However, the vast majority of matrices representing real world graphs are 'sparse' – meaning that almost every value is zero – and RedisGraph stores adjacency matrices in Compressed Sparse Column (or CSC) format, meaning that they are, effectively, only storing nonzero values. This almost always results in a very large saving in terms of memory. Notably, storing matrices in CSC format does not impact RedisGraph's ability to use them in mathematical operations.

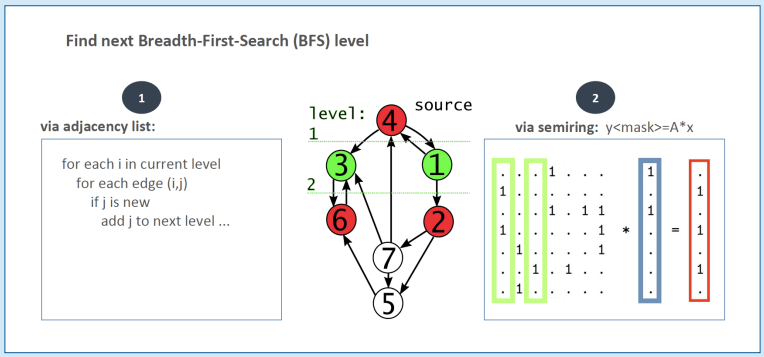


Figure 2 – Breadth-First-Search via adjacency lists and linear algebra

The second hurdle is that, until recently, there was no easy or standardised way to implement queries based on linear algebra (which refers to mathematical operations on matrices, such as matrix multiplication). This has been solved by the release of the GraphBLAS engine (see ***http://graphblas.org***), which underpins RedisGraph. GraphBLAS is an open effort that provides standardised building blocks for graph algorithms based on linear algebra and using algebraic constructs known as semirings (rings with an additive inverse). 'BLAS' stands for Basic Linear Algebra Subprograms. The combination of matrix representation and linear algebra optimises and simplifies many different graph queries and algorithms. For example, a comparison between implementations of Breadth-First-Search using linear algebra and the standard approach using adjacency lists is shown in ***Figure 2***. It should be clear that the algebraic approach is easier to write and to understand. It is also computationally simpler.

There's one more advantage of the matrix representation that is worth noting. Matrix operations (such as multiplication) are extremely and easily parallelisable, and this property carries over to queries and graph algorithms based on linear algebra. This means that RedisGraph benefits very significantly from parallelisation. It will, in the future, be able to take full advantage of the massive parallelisation offered by GPU based processing.

Why should you care?

RedisGraph is an extremely performant graph database. Thanks to the matrix representation it uses and the linear algebra algorithmics it implements, it is able to create over one million nodes in under half a second, and form three million relationships in the same timeframe. What's more, early benchmarks performed by Redis suggest that their graph is order(s) of magnitude faster than its competitors.

Moreover, RedisGraph's CSC storage format mitigates the problems with storage and memory usage that the matrix representation of graphs has had in the past, providing a sixty to seventy percent reduction in memory usage. This makes adjacency matrices a practical proposition, allowing RedisGraph to benefit from all of their advantages with few – if any – of the accompanying downsides.

The Bottom Line

Although RedisGraph has yet to be officially released, it has already seen use by dozens of non-production users running real-time graph use cases. While it is early days in terms of features the theoretical advantages of using adjacency matrices are considerable. Even if you are not already using Redis, it is certainly worth looking into.

FOR FURTHER INFORMATION AND RESEARCH CLICK HERE