Bloor

White Paper

White Paper by Bloor
Author **Philip Howard**
Publish date **April 2019**

Database technology for the 2020s

Redis Enterprise supports multiple logical views into its database, now to include time series data and deep learning models, and has taken this one step further to deliver event-driven data transformations from one model to another, in real-time, in memory. Redis Enterprise will eliminate the need for organizations to deal with siloed data environments and multiple APIs.

Database technology for the 2020s

The business world has changed and continues to do so. Database technology, which supports the business, is evolving to meet the demands of organizations investing in digital transformation. But what does this mean in practice? The most obvious shift from historical approaches to both data and applications is that they are now multi-faceted. If you want to understand your customers better you need to track their communications with your call center and even their comments on social media: relying on traditional transactional data is no longer sufficient. For many applications – predictive maintenance, for example – you need to be able to track not just changes themselves but changes over time so that time itself becomes an important dimension to monitor and record.

So, a database for the 2020s needs to be able to store and process a variety of data types, and it needs to support analytics against all of that data. However, that is not all. In today's world, data is generated and arrives for processing much faster than it ever did before, and in greater volume. Moreover, users, whether internal to the organization or external, expect answers now: not tomorrow, not in minutes, not even in tens of seconds. For many use cases click-to-response end user experience is expected to be in less than one second.

There is therefore a need for an intelligent, data agnostic, performance-focused, globally distributed (at least for large enterprises), scalable database that is continuously available. In this paper we are going to concentrate on the data-agnostic part of this equation – generally known as a multi-model database - though we will not entirely ignore other requirements.

What is a multi-model database?

This is arguably the wrong question. The right question is: what would be your ideal database architecture, if you were starting from scratch, and what would be its characteristics? We suggest that this is what you would want:

1. **Physical model –** that is, in what format the data is persisted to physical storage. You don't actually care about this. What you care about is that this is sufficiently flexible to support all of the logical models that we discuss next, which means a separation between the implementation of the database's logical and physical models. This is where relational databases fell down: although the relational model, as originally conceived, included this separation it wasn't actually implemented by the leading vendors. Moreover, the polyglot persistence approach (multiple databases each doing their own thing) is based on the same, flawed, methodology that your physical model mirrors your logical model.
2. **Logical model –** you want to be able to store any sort of data, in any format, and you want to be able to access and process that data by any appropriate model. For some applications you may want to view the data from a tabular (relational) perspective but for others you may want to view relationships via graphs, you may want to do full text search, or you may want to access data as JSON documents. All of these models, and others, should be possible.
3. **Collaboration across models –** there are lots of applications where you may want to combine data from different logical views into the database: for example, tabular and document-oriented data or time series data combined with graph data. This should be possible in a seamless fashion and without complexity. It's insufficient to remotely call APIs – instead databases must have the ability to aggregate data and automatically triggers business logic on demand to dynamically manipulate data. There are some so-called multi-model databases that do not allow this and, for our money, these would be more accurately described as multi-persistence databases or multi-storage databases. In our view they do not qualify as true multi-model offerings.

“In today's world, data is generated and arrives for processing much faster than it ever did before, and in greater volume. Moreover, users, whether internal to the organization or external, expect answers now: not tomorrow, not in minutes, not even in tens of seconds. For many use cases click-to-response end user experience is expected to be in less than one second.”

While these three characteristics define a multi-model database, there are other elements involved in defining an ideal database that relate to the supported infrastructure and to physical storage. In the case of the former the database should be deployable across any platform: on-premises, private cloud, public cloud, in multi-cloud, and hybrid environments. This will empower organizations to distribute workloads, deliver a combination of security and cost savings and remove limitations and any lock-in imposed by a single service provider. You would also like it to support a microservices based approach with orchestration through Kubernetes and/or similar offerings. As far as physical storage is concerned, while in-memory processing is a requirement for low latency, real-time performance in memory can be expensive. You will

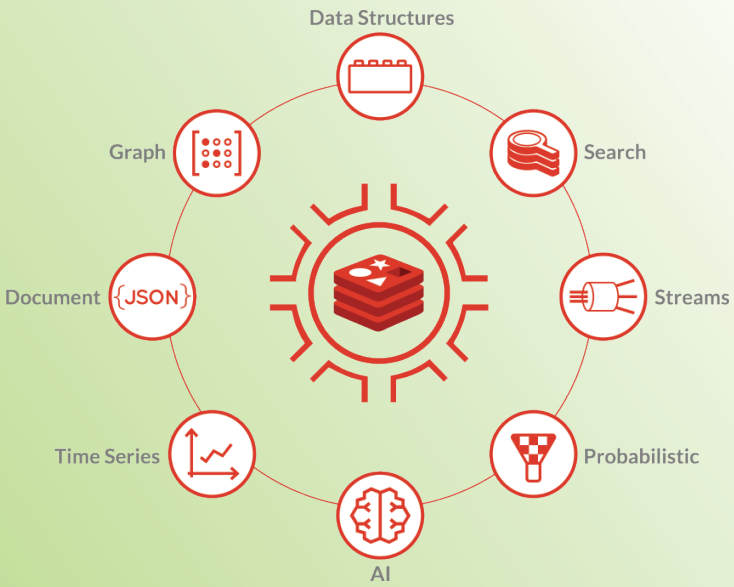

Figure 1 – Modules for extending Redis models

therefore need spill-to-disk capabilities for processing purposes, and you will need options for storage that extend beyond spinning disks to SSDs and the recently announced persistent memory technology from Intel.

What we have come around to is that an *"ideal"* database is actually the same thing as a multi-model database, provided it is a true multi-model offering. There are obviously other features of a database that you are going to require: high availability, workload isolation and management, consistency, a database optimizer, etc. What we shall do in the remainder of this paper is discuss how Redis Enterprise, with its latest releases, supports the criteria we have described earlier. However, as this paper is going to focus specifically on how it supports a multi-model approach through its implementation of *"modules"*, we should say that Redis fully supports the infrastructure and physical storage requirements we have outlined, notably with Intel's Optane DC persistent memory having just been launched (April 2019).

Redis' native support for different data models

While you shouldn't have to care about this model it is important to understand how Redis has been able to provide the flexibility needed to support its multi-model capabilities. Briefly, Redis Enterprise is an in-memory, distributed (automated partitioning), NoSQL database with a key-value store as its underpinning. However, over and above its key-value store Redis supports multiple data structures. These are what, in a relational database, you might think of as datatypes, except that, as can be seen, the structures supported are far broader than you would expect from relational datatypes. As a trivial example consider that SQL has no concept of order so you cannot have sorted sets in a purely SQL environment.

Before going on to discuss Redis modules, which are at the heart of the company's multi-model capabilities it is worth commenting that these structures support a variety of use cases in their own right. For example, Sets are useful for developing real-time recommendations and ad targeting, while Sorted Sets support real-time range analyses, top scores and so forth. The use of Geospatial Indexes is obvious but some of the others may not be. Hyperloglogs are helpful in anomaly detection while hashes are often used to support session management.

Multi-model support

As previously indicated, Redis Enterprise supports multiple logical views into its database, through the use of *"modules"* – see **Figure 1**. The first thing we should say about these is that by module the company means functionality embedded into the product as opposed to something tacked on top. In other words, these are part of the database engine. We should also say that it is possible to develop your own modules, but we are going to focus on the modules that Redis provides, including those that have just been announced as of April 2019 – RedisTimeSeries, RedisAI, and RedisGears. However, we should say that there are two different types of modules that are offered by Redis where the first type are specific to supporting a particular logical model and the second type is about supporting cross-module operation. The latter builds on a fundamental precept of Redis modules, which is that all modules should be able to interact and interoperate with one another. That said, Redis offers a number of specific modules designed to provide particular functionality over and above that interoperability. However, regardless of the module type, they all inherit the properties that Redis is well loved for – its superior performance, flexibility and extensibility and act to support Redis' approach to multi-model support.

From a model specific standpoint, Redis has native key-value capability plus six other models that it supports, represented by RedisJSON, RediSearch, RedisBloom, RedisGraph, RedisAI, and RedisTimeSeries. Of these, the first two have been available for some time, the fourth was released in November 2018, and the last two are new as of April 2019. We will spend more time discussing the most recent releases.

Briefly, RedisJSON, as you might expect, supports JSON documents. Its most notable characteristic is that the binary tree structure it provides enables rapid access to sub-elements within a document. As far as RediSearch is concerned, this supports stemming, document and field scoring, numeric and other filtering, support for aggregate queries, and auto-suggestions. In part, thanks to the secondary indexing provided, Redis' benchmarks suggest significantly improved performance compared to Elasticsearch. RedisBloom implements a scalable bloom filter and a cuckoo filter and the built-in commands making it quite attractive to easily apply the space-efficient probabilistic data structures for data in Redis.

RedisGraph provides a graph model that uses adjacency matrices and supports the OpenCypher language. With respect to adjacency matrices Redis has deployed what is known as GraphBLAS. This is an open effort that provides standardized 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 optimizes 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 – rather than matrices – 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

Regardless of the module type, they all inherit the properties that Redis is well loved for – its superior performance, flexibility and extensibility and act to support Redis' approach to multi-model support.

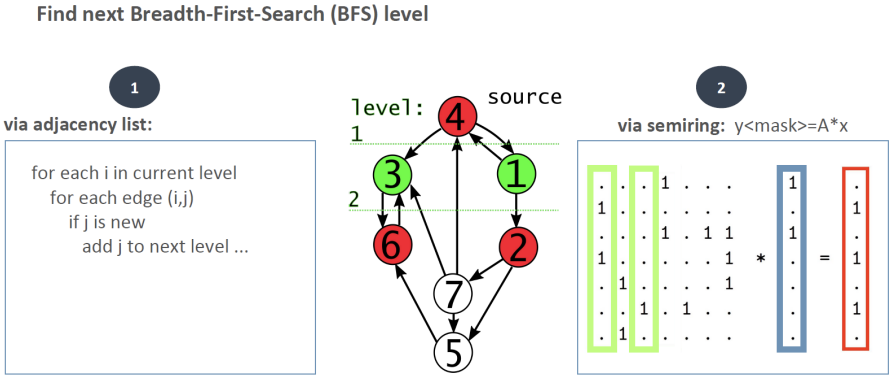

Figure 2 – Breadth-First-Search via adjacency lists and linear algebra

noting. Matrix operations (such as multiplication) are extremely and easily parallelizable, and this property carries over to queries and graph algorithms based on linear algebra. This means that RedisGraph benefits significantly from parallelization. It will, in the future, be able to take full advantage of the massive parallelization offered by GPU based processing.

A further logical model supported through a Redis module is time series, via RedisTimeSeries, which has only just been released. In practice, time series support could previously be managed through the use of sorted sets but the functionality you could enable was limited. However, with this new module you have built-in aggregation functions such as calculating minima, maxima, sums, averages and so on. More significantly, you can label data – based on timestamps – on either an individual or global basis and then you can use those labels to support analytics. This will be particularly useful in Internet of Things (IoT) environments where you are doing analytics at the edge. Additional capabilities include the ability to compress data across different time series functions, programmable retention policies, and support for downsampling (reducing the sampling rate so that you can determine the granularity – typically of sensor data – you require). There is also a built-in Prometheus interface where Prometheus is an open source metrics-based monitoring system designed to measure the overall health, behavior and performance of a service, and the results thereof can be visualized used Grafana.

Moving on to Redis Streams, this is similar – and is an alternative – to Apache Kafka. As such, Redis Streams supports all the functions you would expect, such as data ingestion, acting as a message broker, and stream processing and analytics. There is built-in support for time series data, and radix trees are used to support lookback queries. One of the other advantages claimed for Redis Streams – apart from superior performance – is that it has integrated lifecycle management and does not require the use of additional software (Zookeeper in the case of Kafka) for this purpose.

As we have previously mentioned, all Redis modules are designed to interoperate with one another on a single copy of the dataset and also use a shared Hash data structure to extend and combine queries. RedisGears takes this one step further by facilitating inter-model transformations. It provides a serverless environment – that is, with zero administration – and allows you to aggregate data across multiple Redis database instances and react to activity based on pre-defined triggers. In other words, you get event-driven data transformations from one model to another, in real-time, in memory.

Finally, RedisAI replaces RedisML. The latter, as its name suggests, enabled the deployment of machine learning models and model serving, with the database supporting relevant languages such as Python, R, and Scala. RedisAI extends this into deep learning and adds the ability to embed TensorFlow, PyTorch, and TorchScript models into your analytic workflows. Further, it delivers the ability to run models and graphs that classify data residing in Redis, thereby eliminating any need to separate the data ingestion process from where the model is applied and, at the same time, reducing processing overheads. Note that you can also apply models across GPUs. A limitation is that while RedisAI supports the import and deployment of third-party analytics models, it does not currently support the ability to train models on the Redis platform.

“Redis modules are designed to interoperate with one another on a single copy of the dataset and also use a shared Hash data structure to extend and combine queries.”

The company

Redis is a leading in-memory database platform that originated with the use of a key-value storage paradigm. It is also the commercial provider of Redis Enterprise, an enhanced version of Redis that provides additional functionality (primarily scale-out cluster support, multi-master architecture, high availability, and security) designed to make it enterprise-ready. This is available both on-premises and in the cloud (Amazon, Microsoft, Google, Heroku, others) or can be deployed as a fully hosted and managed service. Redis itself is often rated as one of – if not the – most popular NoSQL database. The company, with more than 7,100 paying customers, extends the native characteristics of Redis to many popular industry use cases implemented through modules.

Redis is a privately-held company backed by venture capital, as of February 2019 raising $146 million in financing. It was founded in 2011 based in Mountain View, California. It has additional offices in London and Tel Aviv.

> **Redis itself is often rated as one of – if not the – most popular NoSQL database.**

www.redis.com

700 E El Camino Real Suite 250 Mountain View
CA 94040, USA

Tel: +1 415 930 9666

Conclusion

A fully functional, multi-model database – as defined here – is about as good as it gets when it comes to database design. While there are certainly applications that only require a single logical view of your data, such applications are reducing in relative importance. Or, conversely, we should say that the number of environments within which you might want to combine different types of data, is increasing and expanding. As a result, it is our belief that multi- and hybrid-cloud multi-model databases represent the future of modern data-centric applications.

We should say that Redis Enterprise is not the only multi-model database in the market. For example, there are several graph/document model databases. However, there are very, very few products that have a sufficiently flexible physical model that they are capable of supporting more than a couple of logical models, especially when you consider our defined requirement that to be truly multi-model you should not have to be dealing with siloed data environments and multiple APIs. Redis has achieved this distinction – and is providing commercial support for all of these models – and is to be commended for having done so.

“
...it is our belief that multi- and hybrid-cloud multi-model databases represent the future of modern data-centric applications.
”

About the author

PHILIP HOWARD
Research Director / Information Management

Philip started in the computer industry way back in 1973 and has variously worked as a systems analyst, programmer and salesperson, as well as in marketing and product management, for a variety of companies including GEC Marconi, GPT, Philips Data Systems, Raytheon and NCR.

After a quarter of a century of not being his own boss Philip set up his own company in 1992 and his first client was Bloor Research (then ButlerBloor), with Philip working for the company as an associate analyst. His relationship with Bloor Research has continued since that time and he is now Research Director, focused on Information Management.

Information management includes anything that refers to the management, movement, governance and storage of data, as well as access to and analysis of that data. It involves diverse technologies that include (but are not limited to) databases and data warehousing, data integration, data quality, master data management, data governance, data migration, metadata management, and data preparation and analytics.

In addition to the numerous reports Philip has written on behalf of Bloor Research, Philip also contributes regularly to ***IT-Director.com*** and ***IT-Analysis.com*** and was previously editor of both ***Application Development News*** and ***Operating System News*** on behalf of Cambridge Market Intelligence (CMI). He has also contributed to various magazines and written a number of reports published by companies such as CMI and The Financial Times. Philip speaks regularly at conferences and other events throughout Europe and North America.

Away from work, Philip's primary leisure activities are canal boats, skiing, playing Bridge (at which he is a Life Master), and dining out.

Bloor overview

Technology is enabling rapid business evolution. The opportunities are immense but if you do not adapt then you will not survive. So in the age of Mutable business Evolution is Essential to your success.

We'll show you the future and help you deliver it.

Bloor brings fresh technological thinking to help you navigate complex business situations, converting challenges into new opportunities for real growth, profitability and impact.

We provide actionable strategic insight through our innovative independent technology research, advisory and consulting services. We assist companies throughout their transformation journeys to stay relevant, bringing fresh thinking to complex business situations and turning challenges into new opportunities for real growth and profitability.

For over 25 years, Bloor has assisted companies to intelligently evolve: by embracing technology to adjust their strategies and achieve the best possible outcomes. At Bloor, we will help you challenge assumptions to consistently improve and succeed.

Copyright and disclaimer

Bloor Research International Ltd
20–22 Wenlock Road
LONDON N1 7GU
United Kingdom

Tel: **+44 (0)20 7043 9750**
Web: **www.Bloor.eu**
Email: **info@Bloor.eu**