What’s the Difference Between Redis and MongoDB?

Remote Dictionary Server (Redis) and MongoDB are two NoSQL databases that store data in an unstructured format. NoSQL databases store data differently than relational databases, which use tables, rows, and columns. Redis is an open-source, in-memory database that stores data as key-value pairs. It stores data in RAM for high performance but does offer on-disk persistent storage as an additional feature. MongoDB is a source-available document database that stores data in serialized JSON format. It stores data on external memory but includes an in-memory storage engine in the enterprise edition.

Read about Redis »

Read about MongoDB »

Data model: Redis vs. MongoDB

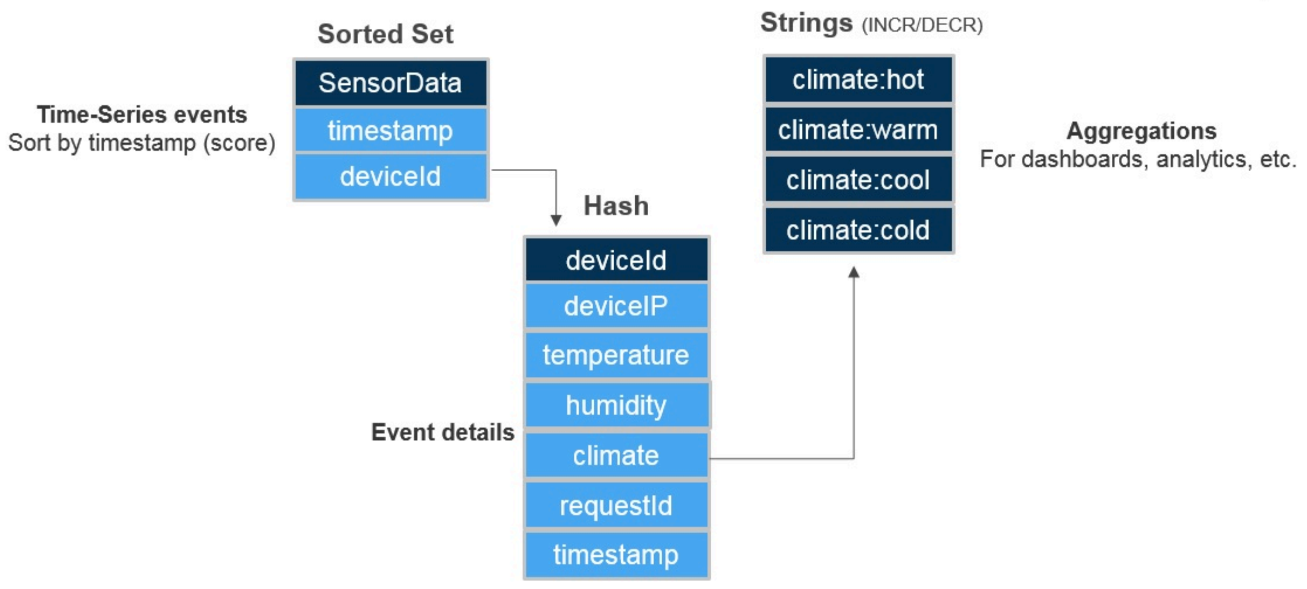
Redis and MongoDB use distinct data models and architecturally store data differently.

Redis

Redis stores data in RAM, so you can access data directly from memory. While this provides low-latency responses, it also limits the volume of data you can store. Redis saves the dataset to disk through snapshotting and append-only file (AOF) logging, which provides data durability.

Redis stores data as key-value pairs, where each data entry has a unique key. It supports various data types like sorted sets, hashes, sets, lists, and strings. Keys can be any length up to a total of 512MB.

The following image shows the Redis data model.

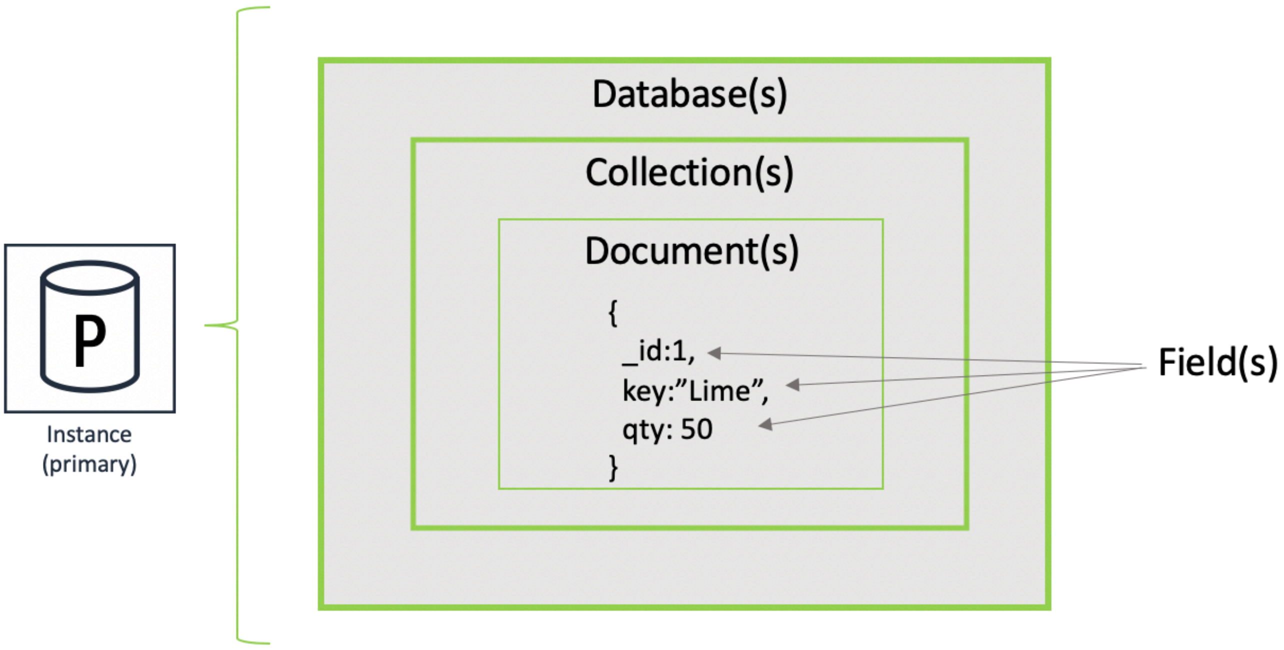


MongoDB

MongoDB follows a document data model to primarily store data in external memory storage. It stores data as serialized Binary JSON (BSON) documents. The maximum document size is 16MB.

MongoDB's architecture is useful for scaling large data volumes as it can increase capability based on disk space. Enterprise users can also use an in-memory storage engine for a hybrid approach. You can use memory caching for data you frequently access, while still primarily relying on disk storage for persistence.

The following image shows the MongoDB data model.



Similarities between Redis and MongoDB

Both Redis and MongoDB are NoSQL databases that offer flexible schema design, horizontal scalability, and high availability. You can use them to store unstructured data like documents and images more flexibly. Unlike relational databases, they allow for dynamic data models that don't use schemas.

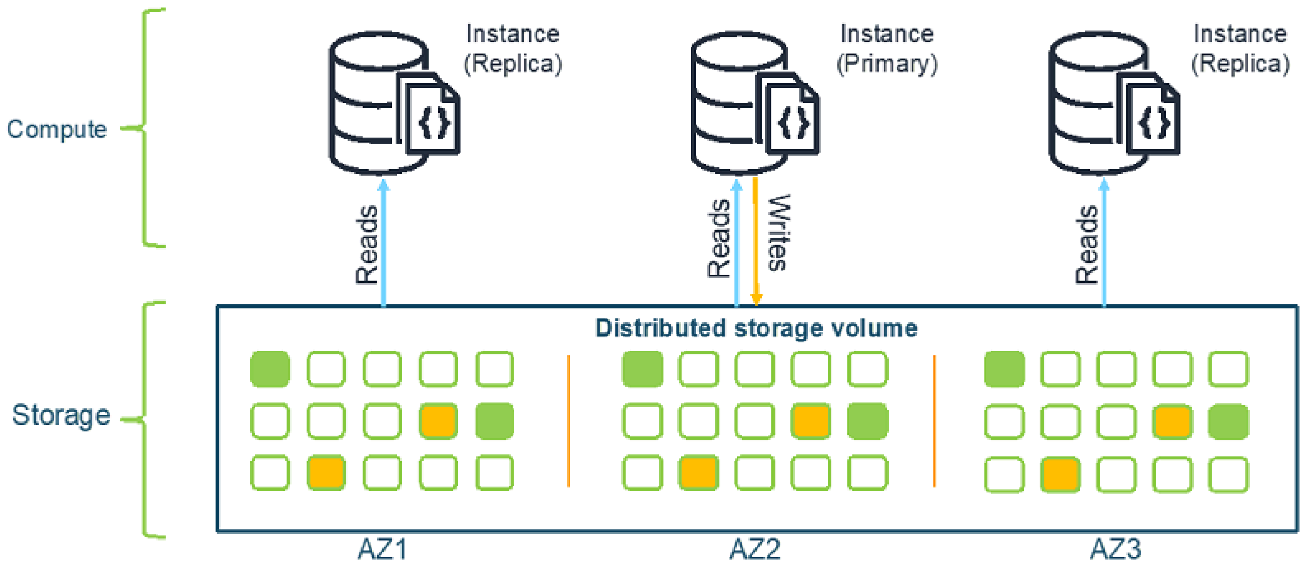
There are various similarities between these two non-relational databases.

Secondary indexing

Redis and MongoDB provide secondary indexing. With this function, you can create additional indexes on non-primary key fields. Once you have made these additional indexes, you can query and retrieve data faster while searching different criteria. Secondary indexing increases flexibility and query performance in both NoSQL databases.

Replication

To provide high availability and durability, MongoDB and Redis use replication. You create replica sets (or clusters), where the NoSQL databases replicate data across several nodes to provide redundancy. In both databases, there's a primary instance that accepts write operations and one or more secondary instances that replicate the data from the primary instance. If the primary instance fails, a secondary one can take over.



Performance

Both Redis and MongoDB deliver low-latency responses and can handle high-throughput workloads. Redis is an in-memory database, which means it stores data in memory. This storage system allows Redis to deliver high-speed read and write operations. MongoDB combines memory storage with disk-based storage, which provides speed and data durability.

Key differences: Redis vs. MongoDB

Redis and MongoDB also have differences in their functions and capabilities. Here are some other key points of difference between the two.

Scaling

Horizontal scaling enables MongoDB to handle large volumes of data efficiently. It uses sharding to distribute data across multiple regions and nodes. With cross-sharding operations, you can query and update across multiple shards.

Redis doesn’t offer the same degree of scalability as MongoDB. Redis only uses a single shard by default for primary operations. You must manually maintain sharding on a hash basis, which makes management more complicated. Redis also lacks cross-shard functionality.

Availability

Both MongoDB and Redis support availability through replication. However, MongoDB supports a higher degree of availability by using replica sets. MongoDB can create up to 50 copies of your data, distributed across multiple nodes, data centers, and even geographic regions. It supports automatic failover mechanisms; if a primary node goes offline, it elects a new primary node from the replicas.

In contrast, Redis doesn’t provide automatic failover by default. Admin users initiate manual failover, especially if the replica is in a different data center. You have to set up and configure a separate component, called Redis Sentinel, if you want automatic failover.

Integrity

MongoDB supports multi-document atomic, consistent, isolated, and durable (ACID) transactions. So, you can keep data consistent across multiple operations. Using multi-document transactions, you can perform multiple operations as one unit. Within a session, MongoDB commits all changes or rolls them back, which helps ensure ACID compliance.

Conversely, Redis does not provide built-in ACID support. However, you can use the *MULTI* command to group multiple commands into a single atomic operation. But this alone isn’t a solution. You also need to implement rollback functionality within your application code, as Redis doesn’t support this natively within transactions.

Query language

MongoDB provides a high level of flexibility in its querying, even performing complex spatial computations and data analysis functions. MongoDB uses MongoDB Query Language (MQL), which supports JSON-like syntax to simplify advanced querying. With MQL, you can perform advanced queries across single or multiple keys, text searches, and ranges.

In contrast, Redis is optimized for fast key-value access operations rather than complex querying and searching capabilities. You use Redis mainly by providing keys and retrieving corresponding data. Redis doesn't have a query language like MQL. Instead, it offers a wide range of commands to interact with data. For example, you can use the *GET* command to retrieve values, by providing the corresponding keys.

When to use Redis vs. MongoDB

For temporary data storage with rapid querying, use Redis. For long-term persistent storage of complex data with rich querying, use MongoDB.

Redis provides fast access to frequently accessed data, which makes it suitable for caching and session storage. You can use it in real-time applications or event-driven architectures due to its built-in support for publish-subscribe (pub/sub) messaging patterns. It also offers advanced data structures like sorted sets and lists, which can be used for implementing rate limiting, task queues, and job scheduling systems. It's also efficient for counting and aggregating data, which makes it suitable for tracking leaderboard data or other statistics.

In contrast, you can use MongoDB for storing complex application data at scale. It offers more traditional database structures and storage that doesn't use a schema, so developers can take a more flexible approach. It handles high-volume writes and reads efficiently and can handle large datasets. You can use it for content management or for managing user profiles at scale. MongoDB also has built-in geospatial indexes and supports spatial queries, which makes it suitable for location-based applications or data with geospatial components.

Can you use Redis and MongoDB together?

Using Redis and MongoDB together is a common strategy in many applications. The speed of Redis naturally complements the long-term storage capabilities of

MongoDB. You can use both Redis and MongoDB to optimize database performance, boost scalability, and provide a flexible system for your applications.

For example, you could use Redis for real-time data processing. Redis is ideal for capturing and processing live streaming data, as it handles real-time data processing scenarios. Then, you can store the data or results that you process with Redis in MongoDB for archiving and more complex integrated analytics.

Another example is a hybrid data model across Redis and MongoDB. You could use Redis' key-value store and MongoDB's document-oriented model at the same time. Redis provides a simple system to access metadata you frequently access, while you can use MongoDB for more complex data structures.

Summary of differences: Redis vs. MongoDB

	Redis	MongoDB
Data model	Key-value based in memory data store.	Persistent document database.
Scaling	Redis doesn't offer as much scalability.	A MongoDB database is highly scalable through horizontal scaling, sharding, and partitioning data.
Availability	You need a separate component called Redis Sentinel to monitor clusters for automatic failover.	Automatic failover by default.
Integrity	Redis offers commands to create single atomic operations. Rollback must be managed in application code.	MongoDB has built-in support for multi-document ACID transactions and rollback.
Query language	Redis uses commands for querying.	MongoDB uses MongoDB Query Language (MQL) to query and manipulate data.

How can AWS help with your Redis and MongoDB requirements?

Amazon Web Services (AWS) has many offerings to support your work with Redis and MongoDB.

Amazon MemoryDB for Redis is a Redis-compatible, durable, in-memory database service that delivers ultra-fast performance. It's purpose-built for modern applications created with microservices architectures. MemoryDB stores data durably across multiple Availability Zones (AZs) using a Multi-AZ transactional log to enable fast failover, database recovery, and node restarts. You can achieve microsecond read and single-digit millisecond write latency alongside high throughput.

Amazon ElastiCache for Redis is a fully managed caching service that makes it easy to set up, operate, and scale a cache in the cloud. With ElastiCache for Redis, you can accelerate application speed and unlock microsecond read and write latency by caching data from primary databases and data stores.

Amazon DocumentDB (with MongoDB compatibility) is a fully managed native JSON document database that scales enterprise workloads effortlessly. With the ability to store, query, index, and aggregate data in a flexible JSON format, you can develop and evolve applications faster than ever. A managed database eliminates the need for you to perform manual database management tasks, which increases productivity and streamlines development.