**MongoDB**

MongoDB is a NoSQL database that uses a document-oriented data model, where each record is a document stored in a collection, instead of the rows and columns common to popular relational databases, such as MySQL.

MongoDB stores the JSON documents using a format called BSON, or binary JSON. The nonrelational nature of these documents mean they can store—and the database can process—structured application data as well as semi structured and unstructured data. Unlike relational databases, MongoDB doesn’t use rigid schemas. Instead, the documents are flexible and can contain arrays and nested documents, allowing for complex and hierarchical data storage.

When handling extremely large data sets, document databases, such as MongoDB, scale out or distribute data across multiple nodes or clusters using a technique called sharding. That model allows for fast storage and recall. This architecture makes sense given that MongoDB was created for ad serving, where potentially millions of ads might need to be called up across thousands of websites at any moment. There was no inherent need to analyse one ad against another, which allowed data to be physically distributed and separated.

Hierarchical document databases are very fast for read operations, but data analysis can be slow because systems must analyse data in all nested entities. Relational databases, by contrast, store their data in separate tables, and a single “object” may be referenced in many tables within the database, allowing for more efficient analytical operations at scale. Given these differing strengths, development teams will generally opt for the best data management system for their application’s current needs. Or they may choose a multimodal database that provides full SQL access to both relational and JSON format as well as many other data types.

**MongoDB vs. RDBMS**

Each type of database—relational, such as MySQL, Postgres, and Oracle Database, or document-oriented, such as CouchDB, DynamoDB, and MongoDB—has strengths and weaknesses, and the choice between them generally depends on the specific requirements and constraints of the application being developed.

A [relational database management system (RDBMS)](https://www.oracle.com/database/what-is-a-relational-database/) uses a Structured Query Language (SQL), whereas MongoDB's document-focused format uses document store APIs. Even so, MongoDB Query Language (MQL) uses a JavaScript-like language with operations such as creating, reading, updating, and deleting documents.

MongoDB has no concept of tables and rows and lacks schemas, so there’s less structure to define before the database can be used. With no central schema, however, each app that accesses the collections needs to understand the document. So the “schema” is in the application code and not defined in the database. If one app changes the schema, other apps may break. Compared with relational databases, where a schema is essentially a blueprint for the RDBMS and data organization and interrelation are explicitly defined, MongoDB lacks the inherent concept of relationships between data.

The flexibility of data stores is notable, as MongoDB uses different formats for data such as key-value stores, graphs, and documents, and data structures can change over time. This differs from an RDBMS, which uses strict definitions, hierarchies, and validation procedures based on these to help ensure data integrity.

While setting up a basic MongoDB instance is straightforward, configuring and maintaining a large-scale, distributed MongoDB cluster with sharding and replicas can be complex and requires a good understanding of its architecture and configuration options.

A screenshot of a computer

AI-generated content may be incorrect.

**MongoDB Features**

MongoDB has become popular with developers in part due to its intuitive API, flexible data model, and features that include:

* **Ad hoc queries.** MongoDB supports field, range, and regular-expression queries that can return entire documents, specific fields of documents, or random samples of results.
* **Indexing.** MongoDB supports several different index types, including single field, compound (multiple fields), multikey (array), geospatial, text, and hashed.
* **Replication.** MongoDB provides high availability with replica sets including two or more copies of the data. Writes are handled by the primary replica, while any replica can serve read requests. If the primary replica fails, a secondary replica is promoted to become the primary replica.
* **Scalability.** Scaling in MongoDB databases is enhanced with sharding, as clusters store only a portion of the data in a collection. Sharding keys determine the distribution of that data.
* **Load balancing.** MongoDB can scale vertically and horizontally, and thanks to sharded clusters, load balancing can be handled by the database’s basic structure. Replication can be used to reduce loads on primary servers.
* **File storage.** Data is stored in documents that readily map to objects in most programming languages, providing easy access within applications.
* **Batch processing.** Data processing can be accomplished in several ways. Sometimes it’s done in the documents themselves, other times with a bulk write method that reduces network operations.
* **Key Features of Mongo DB:**

