What’s the Difference Between Cassandra and MongoDB?

What's the difference between Cassandra and MongoDB?

Apache Cassandra and MongoDB are two NoSQL databases that store data in a non-tabular format. Cassandra is an early NoSQL database with a hybrid design between a tabular and key-value store. It’s designed to store data for applications that require fast read and write performance. In contrast, MongoDB is a document database built for general-purpose usage. It has a flexible data model that allows you to store unstructured data in an optimized JSON format called *Binary JSON*, or BSON. The MongoDB database provides full indexing support and replication with rich and intuitive APIs.

Read about Apache Cassandra »

Read about MongoDB »

What are the similarities between Cassandra and MongoDB?

Apache Cassandra and MongoDB both belong to the NoSQL database group. NoSQL databases can store structured, unstructured, and semi-structured data without a database schema.

Data storage in NoSQL databases isn’t constrained by the tabular format and relationships between tables, unlike in a traditional relational database management system. You can freely partition and replicate data across multiple nodes to scale efficiently.

Additionally, Cassandra and early versions of MongoDB are open source. This means you can download the source code of these NoSQL databases and configure them as you like.

Facebook developed Cassandra and then released it as an open-source project with Apache. MongoDB was developed by a small group of developers under MongoDB, Inc. All versions of MongoDB released before October 16, 2018 are available under the GNU Affero General Public License.

Read about NoSQL »

Read about open source »

Cassandra

Cassandra stores data as key-value stores. It allows you to define tables with rows and columns, but the tabular structure isn’t used in actual storage. Instead, it uses the wide column-oriented database model, so each row in the table can have a different set of columns.

You can group columns into column families based on their data type or usage. Every row has a primary key that you can use to quickly read data from Cassandra.

Apache Cassandra table structure can be visualized in the following example.

Customer id 1	Column - Name	Column - Country	
	Value - John Doe	Value - United States	
Customer id 2	Column - Name	Column - Age	Column - Email
	Value - Jane Doe	Value - 35	Value - jane_doe@example.com

MongoDB

In contrast, MongoDB stores data without a schema, using an optimized Binary JSON (BSON) format. It can store several data types in a single document, similar to JSON objects, and then serialize it with BSON.

MongoDB organizes documents into collections that can contain data with different structures. Its data model is flexible and can handle large volumes of unstructured data.

The following is an example of the customer data in MongoDB.

```
customers:[

{

  customer_id: "1",

  name: “John Doe”,

  country: "United States"
  age: “35”

  email: "jane_doe@example.com"

}]
```

Architectural differences: Cassandra vs. MongoDB

Due to differences in their data models, Cassandra and MongoDB implement several database features differently.

Basic unit of storage

In Cassandra, Sorted String Tables (SSTables) are the basic unit of storage used to persist data on disk. An SSTable is a file that contains a sorted set of key-value pairs for a particular column family (table) and partition. SSTables are immutable, which means that once they’re written, they can’t be modified.

In MongoDB, the basic unit of storage is a document. A document is a set of key-value pairs where the keys are strings and the values can be of various types. For example, values can be other documents, arrays, strings, numbers, dates, and Boolean values. Documents are stored in collections.

Query language

A query language is the statements you use to insert and retrieve data from the database.

Cassandra Query Language (CQL) is the query language that you use on Cassandra. While it has a similar syntax and structure to SQL, Apache has developed CQL to work with the column-family data model.

On the other hand, MongoDB uses MongoDB Query Language (MQL), which has similar commands as Node.js. MQL supports create, read, update, and delete (CRUD) operations. You can write MQL commands on the MongoDB Shell.

Indexing

Indexing is a technique used in databases to improve the speed and efficiency of data retrieval operations. It involves creating a data structure that maps the values of one or more columns in a database table to the physical location of the corresponding data on disk.

SASI indexes store index data directly in the SSTables. They support complex queries like *range*, *prefix*, and *full-text search* on columns with large numbers of unique values.

In contrast, MongoDB supports indexing at the collection level and field level. It provides multiple index types like single field, compound, and multikey. It also offers these indexes:

- specialized geospatial index for geographically distributed data
- a text search index for large volumes of text data
- hashed and clustered indexes for numerical data

Concurrency

In databases, concurrency refers to when multiple users or processes can access and perform database transactions simultaneously without interfering with each other.

Cassandra achieves concurrency through turntable consistency and row-level atomicity. Only a single user can operate on a single row at a time.

In turntable consistency, each replica node maintains a vector clock, which is a data structure that tracks the version history of associated data. When a write operation is performed, the vector clock updates to reflect the new version. When a read operation is performed, Cassandra returns the version with the highest timestamp across all replicas, which ensures that the most recent version of the data is always returned.

In contrast, MongoDB supports mechanisms for multi-version concurrency control (MVCC). MVCC allows multiple versions of the same data document to exist simultaneously. Each document has a unique revision ID that is incremented on each update. Document-level locking and MVCC provide a more robust concurrency strategy.

Availability

Availability means you ensure there’s no data downtime, even during server outage. Both Cassandra and MongoDB ensure availability by replicating data across multiple sever nodes.

In Apache Cassandra, each node in the cluster holds data replicas for other nodes. Every node coordinates reads to the correct node to write or pull data. Simultaneously, it also repairs data that got out of consistency across the nodes. This can impact performance at scale.

In contrast, MongoDB uses single primary node replication to offer high data availability. MongoDB replicates data into replica sets. Only one primary node receives the writes, and the other nodes simply replicate data from the primary node. However, the primary node creates a single point of failure.

scale your database without causing bottlenecks.

Cassandra uses a distributed hash algorithm called *consistent hashing* to determine which node is responsible for a particular data value. Cassandra also supports *virtual nodes* (vnodes), which allow a single physical node to have multiple data ranges.

In contrast, MongoDB uses sharding keys to identify where the data value can go. Database administrators can define sharding keys to partition the data. You can divide the data based on factors like geographical location, alphabetical order, or any other system that is most efficient for your dataset.

When to use Cassandra vs. MongoDB

Cassandra's high uptime and distributed architecture make it a good choice for high-availability requirements. MongoDB's ability to handle unstructured data with a document-oriented approach makes it useful for systems where data is constantly changing.

Here are some factors to consider when you choose between the two.

Data format

Apache Cassandra has a more structured data storage system than MongoDB. If the data you're working with is in a fixed format, Cassandra is more suitable.

If the data is more dynamic and doesn't have a consistent structure, MongoDB works better.

Availability

MongoDB has a primary node and then a series of replicas. If the primary node goes down, MongoDB spends a few minutes choosing a replica node to substitute. This can cause a small amount of downtime.

Cassandra uses a distributed node system with many master nodes, which allows for 100% uptime availability.

Scalability

MongoDB gives you more control as you scale. You can decide how to partition data across nodes based on your requirements, and manage massively distributed databases at scale.

Cassandra's performance may drop slightly at scale, depending on the data values.

MQL has different implementations and syntax, and may have a steeper learning curve.

Support for programming languages

MongoDB supports twelve programming languages: C, C++, C#, Go, Java, Node.js, PHP, Python, Ruby, Rust, Scala, and Swift.

Cassandra supports fewer languages, like Java, JavaScript, Perl, Ruby, Scala, C#, Erlang, PHP, Python, etc.

Summary of differences: Cassandra vs. MongoDB

	Apache Cassandra	MongoDB
Data model	Cassandra uses a wide-column data model more closely related to relational databases.	MongoDB moves completely away from the relational model by storing data as documents.
Basic storage unit	Sorted string tables.	Serialized JSON documents.
Indexing	Cassandra supports secondary indexes and SASI to index by column or columns.	MongoDB indexes at a collection level and field level and offers multiple indexing options.
Query language	Cassandra uses CQL.	MongoDB uses MQL.
Concurrency	Cassandra achieves concurrency with row-level atomicity and turntable consistency.	MongoDB uses MVCC and document-level locking to ensure concurrency.

Partitioning	Consistent hashing algorithm, less control to users.	Users define sharding keys and have more control over partitioning.

How can AWS support your Cassandra and MongoDB requirements?

Amazon Web Services (AWS) offers two services that support common Apache Cassandra and MongoDB requirements.

Amazon Keyspaces (for Apache Cassandra) is a highly available, managed database that lets you move your Cassandra workloads to the cloud. Amazon Keyspaces is serverless, so you only pay for the resources you use—and the service can automatically scale tables up and down in response to application traffic. You can build applications that serve thousands of requests per second with virtually unlimited throughput and storage.

Amazon DocumentDB (with MongoDB compatibility) is a fully managed native JSON document database. It makes it easy and cost-effective to operate critical document workloads at virtually any scale without managing infrastructure. Amazon DocumentDB simplifies your architecture by providing built-in security best practices, continuous backups, and native integrations with other AWS services.

Get started with managed Apache Cassandra and MongoDB database services on AWS by creating an AWS account today.

Next Steps with AWS

Learn how to get started with Cassandra on AWS

Learn how to get started with MongoDB on AWS

Sign In to the Console

Learn About AWS

What Is AWS?

What Is Cloud Computing?

AWS Inclusion, Diversity & Equity

What Is DevOps?

What Is a Container?

What Is a Data Lake?

What is Generative AI?

AWS Cloud Security

What's New

Blogs

Press Releases

Resources for AWS

Getting Started

Training and Certification

AWS Solutions Library

Architecture Center

Product and Technical FAQs

Analyst Reports

AWS Partners

Developers on AWS

Developer Center

SDKs & Tools

.NET on AWS

Python on AWS

Java on AWS

PHP on AWS

JavaScript on AWS

Help

Contact Us

Get Expert Help

File a Support Ticket

AWS re:Post

Knowledge Center

AWS Support Overview

Legal

AWS Careers

Create an AWS Account

Get Started for Free

Contact Us

Language

عربي |
Bahasa Indonesia |
Deutsch |
English |
Español |
Français |
Italiano |
Português |
Tiếng Việt |
Türkçe |
Русский |
ไทย |
日本語 |
한국어 |
中文 (简体) |
中文 (繁體)