EVENT Get 50% off your ticket to MongoDB.local NYC on May 2. Use code Web50! Learn more >

CouchDB and MongoDB Compared

Get started free with MongoDB Atlas

The application modernization era is happening across a global variety of industries. The use of NoSQL databases such as MongoDB and CouchDB over legacy relational databases is one of the key points of this process. Although MongoDB and CouchDB share a similar concept of a document-oriented database, there are some key advantages to MongoDB over CouchDB technology as well as some key differences we should address.

This article will cover the main areas of comparison between CouchDB and MongoDB which developers need to know before planning their next application stack.

Table of Contents

The basics of MongoDB

MongoDB is a document database built for general-purpose usage. It stores data in an optimized JSON format called BSON (Binary JSON). This allows the MongoDB document model to benefit from the flexibility of JSON documents stored in logical groups called collections. The document model supports many different data types, including strings, dates, numbers, arrays, decimals, nested objects, geo data, and binary data.

MongoDB uses replication and data partitioning called sharding to distribute data for high availability and scalability purposes, making it a highly consistent and fault tolerant database. MongoDB supports multiple deployment methods, from running it for free on a variety of platforms and servers starting from your local machine to a fully blown deployment in the cloud of your choice using MongoDB Atlas. Additionally, MongoDB Enterprise Advanced and MongoDB Atlas offer enterprise-grade security features like authentication, authorization, and LDAP support as well as end-to-end encryption.

MongoDB's unified query API and powerful aggregations, in conjunction with the flexibility of the document model, has made it the most popular general purpose document database on the market. MongoDB was voted among the "Most Desired" databases in StackOverflow's 2023 annual developer survey.

The basics of CouchDB

CouchDB is a NoSQL document-oriented database from the Apache foundation, first released in 2005, and implemented in Erlang. CouchDB documents are JSON-based and stored in databases. The database software can be installed as a single instance or a cluster of instances for durability and high availability. CouchDB's overall design favors availability over consistency, and is considered eventually consistent in many read scenarios.

CouchDB is designed on the concept of "optimistic concurrency," which places an added burden on the application for handling data consistency. The query system does not lock database objects on writes, meaning any conflict resolution or locking mechanisms for consistency need to be implemented by the application developer. This places an additional burden on development teams, and adds unnecessary complexity to their application code.

CouchDB uses an HTTP Rest API to perform database access and document manipulation, rather than native drivers using a unified query language like many other modern databases. It supports basic types of authentication like session cookies/users and TLS/SSL for the Rest API traffic.

CouchDB vs MongoDB comparison table

CouchDB

Data Model	CouchDB stores JSON documents in internal format separated in logical databases only. The data formats are limited to: strings, numbers, arrays, objects, boolean.
Indexing	Indexing of documents is limited. For some workloads, indexing requires a special "Design Document" definition.
Query Language	CouchDB uses an HTTP Rest API to its server endpoint to query and manipulate data. Queries in the basic API support either returning all documents in a database, or retrieving a specific document based on a unique identifier. If you require more complex queries, you will need to use views and map-reduce aggregations to filter and query your documents. There is a possibility to use Lucene indexes for full text search but it requires additional configuration.
Transactions	CouchDB does not support transactions.
Concurrency	CouchDB is eventually consistent and uses optimistic locking as part of its database operations. Write operations do not lock database objects and conflict errors need to be resolved by the application developer.
	CouchDB supports a concept of a replication factor as well as a sharding setting on a global or per database basis. In general, each node holds some specific document ranges while others

High Availability and Scalability	hold the copy of this range. This is the main mechanism for fault tolerance when a server fails. It's also possible to partition the data so that different nodes will have different pieces of the entire database. However, each node is a replica as well as a shard, meaning access can be achieved via any data bearing node. This adds to the complexity of maintaining concurrency as all replicas can suddenly become a primary, adding additional latency and consistency gaps. Sharding is achieved through an internal mechanism which is obfuscated from the user. There is no way to specify specific shard keys or define specific document groups to sit in a specific shard.
Security	CouchDB security is built for simple website backend requirements. It supports: • Basic authentication for your API calls. • Cookie Session authentication for API calls.. • User defined roles. • TLS/SSL for your API calls transport. There are no additional enterprise-grade security features like field level encryption or storage encryption. No LDAP or Kerberos integrations.
	There is an implementation called PouchDB (lightweight java script store).

CouchDB

Mobile Support	It can sync as offline-first to a CouchDB Server using a built-in mechanism. PouchDB is only designed for web browsers, and for mobile apps you will need to use other third-party solutions. Conflict resolution needs to be solved by the developer.
Cloud offerings	Offered as part of Cloud Providers marketplaces. No cross provider clusters available. No simple migrations between clouds.
Documentation & University	The documentation is very sparse, lacking examples and tutorials. No official courses provided.
Native Data Visualization Tooling	CouchDB does not offer any native data visualization tools. This means users must implement third-party software on top of their CouchDB deployment in order to build dashboards for business intelligence or data analytics.

MongoDB

Data Model	MongoDB stores documents in an optimized BSON format. Documents are grouped in databases and collections and returned as JSON documents with support for a large number of data types including: strings, numbers, geo data,

	dates, arrays, decimal, nested objects, binary data. Schema validation is available for different levels of validation. Optimized storage for time series, capped collections and more.
Indexing	Secondary indexes on any field are available and supported in different types: Compound, Text, Geo, TTL, Partial, Wildcard, and Compound Wildcard.
Query Language	MongoDB has a very rich query language called MQL. It supports a wide variety of modern native drivers as well as a shell. The data can be edited, deleted, inserted, and queried in many shapes and forms. The queries can use complex operators like geo queries, text queries, regular expressions, and compound conditions. Any query can be sorted or have projections. Additionally, the aggregation framework provides an impressive and robust pipeline to aggregate and reshape your data, as well as join collections and write them to other destinations.
Transactions	MongoDB supports fully ACID compliant transactions, even on sharded environments.

MongoDB

Concurrency	MongoDB allows multiple database users to concurrently access the same data by managing a well defined concurrency control. MongoDB uses document level locking so writes to a single document occur either in full or not at all, and clients always see consistent data. Together with those mechanisms, MongoDB supports different read and write concerns for distributed clusters and retryable reads and writes.
High Availability and Scalability	MongoDB was built from the ground up to support distribution of data using replication and sharding mechanisms. Replica sets host an identical copy of the data and elect a primary which receives all the writes, while other nodes are secondaries replicating all the data. This is the main mechanism for high availability and fault tolerance as primary failover is automatic. With read-preference, secondaries can be used for workload isolation on read operations. Sharding allows you to easily scale your collections across multiple replica sets. With geo-zone sharding, you can also easily manage data sovereignty requirements. The ability to define specific shard keys and reshard collections with zero downtime when a shard key is no longer optimal gives your application a huge advantage when managing massively distributed datasets at scale. You can

MongoDB

	also take advantage of the shard key advisor commands, introduced in MongoDB 7.0, to make the best of your refined shard key. Additionally, a cluster router is designed to load-balance reads and writes to their relevant shards, allowing users to avoid overhead of coordination and quorum protocols.
Security	MongoDB supports enterprise-grade security mechanisms to secure your MongoDB deployments. Most of them are on by default in MongoDB Atlas cloud offering: • Authentication and authorization using built-in SCRAM or certificates. • TLS/SSL, x509, Client Side Field Level Encryption and Queryable Encryption. • Server Side storage engine encryption. • LDAP and Kerberos integrations. Additionally, MongoDB cloud offerings have strong security compliance certifications. Read more on our trust center.
	Atlas for the Edge allows MongoDB to run on diverse edge infrastructure, from self-managed, on-premises servers to cloud deployments offered by major cloud providers. Data seamlessly flows between and is kept synchronized across all sources, including mobile devices, ensuring real-time data delivery with minimal latency.

MongoDB

Mobile Support	The Atlas Device SDKs provide a lightweight reactive object-store designed to work seamlessly with mobile frameworks. MongoDB also offers an offline-first sync service to Atlas clusters across various client platforms. Atlas Device Sync offers automatic conflict resolution and strong eventual consistency. From Realm's perspective, changesets may arrive any time that connectivity allows. Atlas Device Sync allows data to sync seamlessly between MongoDB Atlas clusters and user or IoT devices.
Cloud offerings	MongoDB Atlas, the *database-as-a-service* platform, offers clusters in all three major cloud providers, starting from a free tier to a fully blown production cross-region and cross-cloud cluster. Together with Atlas, you get the advantages of using Atlas App Services application services, Atlas Charts, Atlas Data Federation for querying Atlas clusters and cost-effective data storage, as well as Atlas Search for optimized full text search.
Documentation & University	Detailed documentation with examples and full tutorials including a full community and developer hub websites. Online university with some free courses available at

	MongoDB
	https://university.mongodb.com.
Native Data Visualization Tooling	MongoDB Charts provides a quick, simple, and powerful way to perform data visualization with MongoDB Atlas data.

CouchDB vs MongoDB queries explained

Let's explore some basic and a bit more advanced queries and CRUD operations and compare them.

Create a document

With both methods, we submit the document to be stored in “planets” (Database/Collection).

MongoDB Example

```
db.planets.insertOne({"name" : "Earth"});
```

CouchDB Example

```
curl -X PUT 'http://127.0.0.1:5984/planets/doc' -d '{"name": "Earth"}'
```

Query a document by Id

Retrieving the data in MongoDB is done with a query parameter, whereas in CouchDB, it's part of the API URL.

CouchDB has to return a revision field called “_rev”.

MongoDB Example

```
db.companies.findOne({"_id" : "8843faaf0b831d364278331bc3001bd8"});

{
```

```
"_id"  : "8843faaf0b831d364278331bc3001bd8",
"name" : "Example, Inc."
}
```

CouchDB Example

```
curl -X GET http://127.0.0.1:5984/demo/8843faaf0b831d364278331bc3001bd8

{"_id":"8843faaf0b831d364278331bc3001bd8",
"_rev":"1-33b9fbce46930280dab37d672bbc8bb9",
 "name":"Example, Inc."}
```

Update a document

To update a record in MongoDB, we need to issue an updateOne command and specify the new field values under the $set operator in the updated clause.

In CouchDB, there are two operations to be performed. We need to first query the document for the latest revision and then overwrite the whole document specifying the revision.

MongoDB Example

```
db.albums.updateOne({"_id" : "6e1295ed6c29495e54cc05947f18c8af"}, {$set : {title : "On
```

CouchDB Example

```
curl -X GET http://127.0.0.1:5984/albums/6e1295ed6c29495e54cc05947f18c8af

{"_id":"6e1295ed6c29495e54cc05947f18c8af","_rev":"1-2902191555","title":"There is Noth

curl -X PUT http://127.0.0.1:5984/albums/6e1295ed6c29495e54cc05947f18c8af \
     -d '{"_rev":"1-2902191555","title":"One by One","artist":"Foo Fighters","year":20
```

Aggregate a group by tag

Aggregations with MongoDB are a query language API receiving a pipeline of stages.

CouchDB, on the other hand, requires definitions of various views with map-reduce stages to perform simple aggregations like this one.

MongoDB Example

```
db.products.aggregate([{$group : {_id : "$tag" , value : {$sum : 1}}}]);
```

CouchDB Example

```
## A view and a design document is required for map-reduce.

## Map function
curl -X PUT http://127.0.0.1:5984/db/_design/products
-d '{"views":{"my_filter":{"map":
"function(doc) {
    if(doc.tag) {
        doc.tags.forEach(function(tag) {
            emit(tag, 1);
        });
    }
}
",
"reduce" : "function(keys, values) {
    return sum(values);
}
" }}}'

curl -X GET http://127.0.0.1:5984/db/_design/products/_view/my_filter
```

Conclusions

At first, CouchDB looks like a compelling solution for the web world since it's built on top of the JSON document model and HTTP rest API interface. However, it lacks in security, performance, ease of use, query language, data types support, and cloud offerings when compared to MongoDB Atlas.

Atlas Device Sync & SDKs close the only gap of offline-first application development, which was a historical point in favor of CouchDB mobile sync solutions.

MongoDB Atlas offers you a fully featured data platform with all the benefits of MongoDB and much more including Charts, Atlas App Services, and Data Federation.

When it comes to general-purpose document databases, MongoDB is clearly the leader.

FAQs

What is CouchDB and MongoDB?

CouchDB and MongoDB are document databases built on the concepts of flexible JSON schemas. However, MongoDB is a more robust and fully featured general purpose database built to support all modern applications use cases.

Why is MongoDB so popular among developers?

The ease of use and increase in developer productivity when working with MongoDB has built a large and growing community of MongoDB developers. Project numbers are growing as users are becoming more and more successful with MongoDB adoption.

Which NoSQL database is best?

According to various third-party assessment providers (like https://db-engines.com), MongoDB is the leading NoSQL database out there. If you would like to find out more about how MongoDB can serve your use case, check out these resources.

Does MongoDB have a REST API?

MongoDB Atlas has a new feature called Data API providing users of Atlas with a REST API interface to basic CRUD and aggregations.

Is CouchDB a SQL database?

No. CouchDB is a NoSQL database providing a REST API interface.

Is CouchDB related to Couchbase?

No. CouchDB is an open-source database produced by The Apache Software Foundation. Couchbase, Inc does not own or produce CouchDB.

MongoDB.

English

About

Careers

Investor Relations

Legal Notices

Privacy Notices

Security Information

Trust Center

Support

Contact Us

Customer Portal

Atlas Status

Customer Support

Manage Cookies

Social

GitHub

Stack Overflow

LinkedIn

YouTube

X

Twitch

Facebook

© 2024 MongoDB, Inc.