Jeet Shah and Jash Popat

Professor Prajish Prasad

CSIT372 – Advanced Database Systems

11 March 2024

Decoding MongoDB

MongoDB is a popular **open-sourced**, **cross-platform NoSQL** database that stores data in a flexible document format. It was founded by Dwight Merriman, Eliot Horowitz, and Kevin Ryan – the team behind DoubleClick. It was started in 2007 and later open-sourced in 2009. The database is built with distribution and vertical scaling in mind. Fearless concurrency is one of the pillars on which MongoDB has been developed.

What and How is MongoDB?

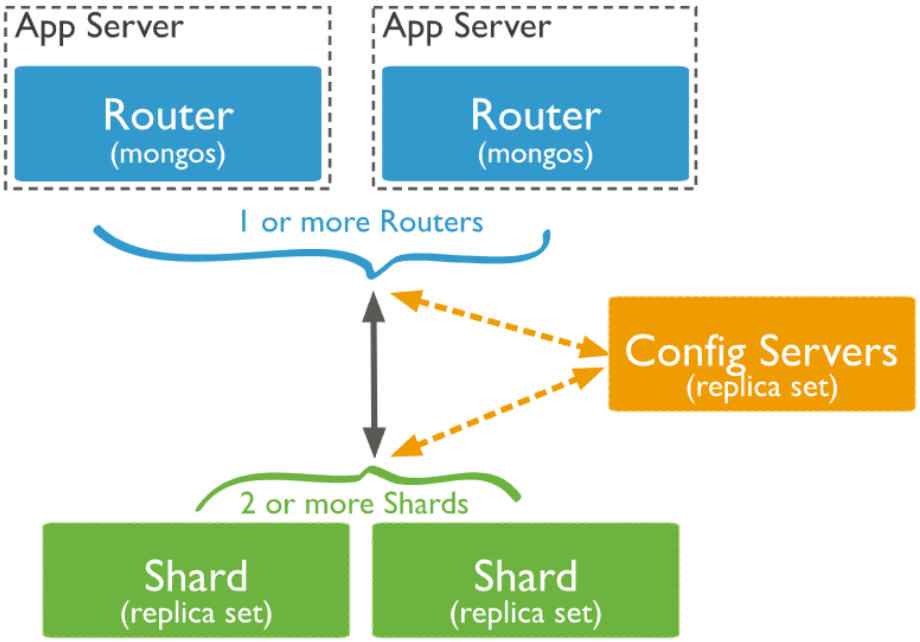
MongoDB is a NoSQL database which means that the databases created with MongoDB and other databases of such formats do not have to adhere to a strict schema. This allows for the database to be easily extended to make space for new features that might require the use of database access. There is no need for the database to be taken down for the schema to be updated, one can keep embedding new documents with new key-value pairs.

Dynamic schemas are powered by the file format of BSON, a JSON file serialized in binary. BSON is used over other key-value storage formats due to its efficiency in being parsed to perform CRUD operations over the database. Moreover, there are a significantly larger number of data types supported by BSON; these data types are akin to native C data types, allowing for parsers written in C to read and write documents (to extents) on disk.

The vertical scaling of MongoDB is made easy due to the ability of the whole database being easily sharded. Sharding is a method to distribute data across multiple machines to allow for high throughput operations across very large datasets. One can also make use of load balancing servers or endpoints that allow for us to systematically assign operations to different shard to maintain high throughput across large datasets.

A sharded cluster has three components:

- **Shard**: Each shard contains a subset of the data. Each shard must be deployed as a replication set, which is just a canonical term describing the fact that a shard will implement data replication and automatic failover. Data redundancy is the default in MongoDB database.
- **Mongos**: These act as query router, being an interface between client applications and sharded clusters. They act as default load balancers for a MongoDB deployment, one can customize the load balancing or rewrite the *mongos* to suit their needs.
- **Config Servers**: They store the metadata for the entire cluster, not just an individual.

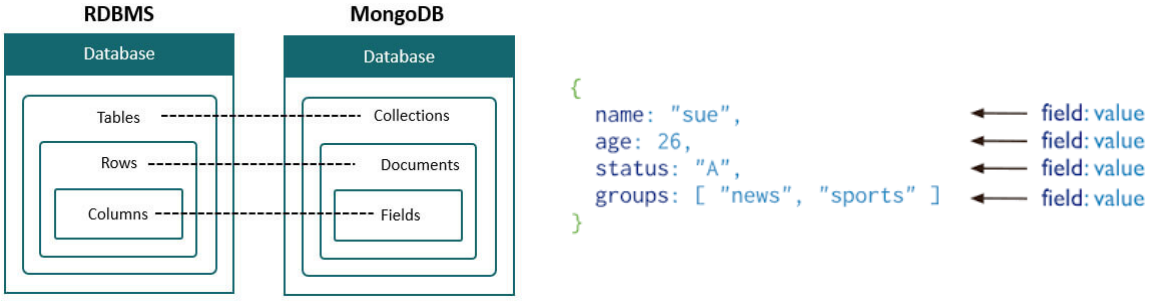


Where is MongoDB?

MongoDB uses a document-based NoSQL approach to data storage; this leads to multiple advantages such as a dynamic approach to storing data for a client application. This approach also leads to a differing method in storing such data on disk and we will discuss how MongoDB goes about the same.

The smallest unit of data storage in MongoDB are a key-value pair. Multiple key-value pairs put together form a document, which is essentially the BSON file that stores individual record (borrowing terms from relational database). One can imitate the idea of foreign keys within a document by making the value of a key-value pair a document itself. This is not encouraged within NoSQL database and it is recommended to use a relational database solution to model data with multiple relations.

Multiple documents grouped together is known as a collection, this is akin to a table in a relational database. Multiple collections stored together is known as a database; an application with multiple features will have to separate data storage across multiple databases. Each collection in MongoDB has an overarching namespace attached to it (this is not to be confused with the traditional definition of namespace in programing). Namespace is the name a maintainer of a database might use to call and interact with a given collections and the indices created for the collection.



Each collection stores three types of files:

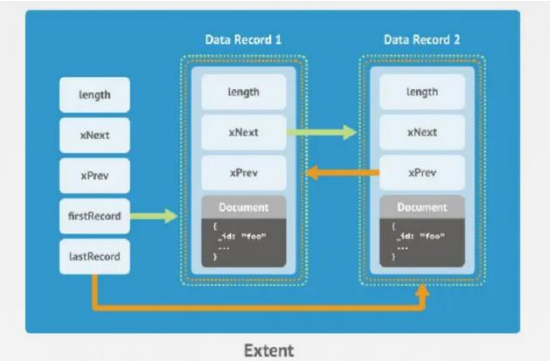
- Extents
- Namespace file
- Index file

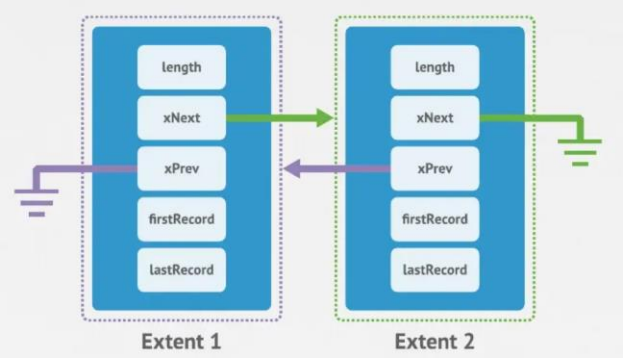
Extents are the files that actually store documents that are written to a collection in a database. Each extent is aggressively pre-allocated and each extent file can grow up to 2 GiB in size. If one were to open and try to read the contents of an extent file (after you get around the fact that it is stored in binary). One would find multiple documents written to the file, the latest extent

in a collection directory generally has nothing written to it as an extent is pre-allocated as soon as something is written to the current latest extent. Each document is also stored as a data structure embedded within a doubly linked list.

Namespace files are a doubly linked list serialized into binary. Each node of the list stores information about each extent file: length, first record, and last record.

The index files are usually stored in a subdirectory within the collection directory. Indices are usually kept in primary memory, while the index files are stored on disk as a form of cache so one does not need to continually rebuild the index. The index files are the serialized form of a B+ tree that is built using one of the fields in the documents present in the collection. Indices make read operations extremely fast, but this comes at a cost of write operations as the entire index needs to be updated to reflect the current state of the data.





Why is MongoDB?

MongoDB has many other features to ease the reading and writing data within one's databases. The biggest, new feature is the introduction of data aggregation pipeline. This allows one to chain multiple aggregation operations, like a transaction. This allows one to use MongoDB as a backend to something like a dashboard. Aggregation operations allow us to perform arithmetic and set operations across an entire collection.

Moreover, MongoDB is being used in production by multiple large companies like Bosch, Sojo, Sanoma, and many more. The database has in constant development for over a decade and a half and should definitely be one of the first choices when picking a NoSQL database.

How to MongoDB (with Python)?

```
import pymongo

# Assuming you have a MongoDB server running on localhost:27017
client = pymongo.MongoClient("mongodb://localhost:27017/")

db_name = "your_database_name"
collection_name = "your_collection_name"

# Get the database (creates it if it doesn't exist)
db = client[db_name]

# Sample dummy data (replace with your desired structure)
data = [
    {"name": "Alice", "age": 30, "city": "New York"},
    {"name": "Bob", "age": 25, "city": "London"},
    {"name": "Charlie", "age": 42, "city": "Paris"},
]

# Insert the data into the collection
db[collection_name].insert_many(data)

# Create and index on the 'name' field
index_name = db[collection_name].create_index({"name": 1})  # Ascending order (1)
print(f"Successfully created index '{index_name}' on collection '{collection_name}'.")

# Close the connection
client.close()
```