Big Data Systems

Djellel Difallah

Spring 2023

Lecture 6 – Document Stores

MongoDB

Different Types of NoSQL

Taxonomy based on data models [Rick Cattell, SIGMOD Record 2011]

- Key-value stores
  - Value is arbitrary (think of it as a scalable hashmap)
  - e.g., Project Voldemort, Memcached, **Dynamo**
- Column Family (a.k.a Extensible Record Stores)
  - Some schema structure
  - e.g., HBase, **Cassandra**, PNUTS
- **Document stores**
  - Complex data structure (JSON seems to be the standard but can be XML)
  - e.g., SimpleDB, CouchDB, **MongoDB**

Agenda

- Document Stores Overview
- MongoDB Introduction
- Data Model
- Query Operations
- Consistency and Replication
- Storage Engine
- MongoDB vs Cassandra

Why Document Store?

- There is a big gap between the application and the data representation.

- Central concept: the Document
  - Similar, in some ways, to records or rows in relational databases, but are less rigid
  - They do not required to adhere to a strict schema, nor will they have all the same sections, slots, parts, or keys.

- Designed for storing, retrieving, and managing document-oriented information such as semi-structured data, e.g.,
  - JSON, XML, YAML, CSV, etc.

Why Document Store? (cont.)

- JSON is popular for data exchange (XML less nowadays)
- Data stored in document DB can be used directly
- Databases often store objects from memory
  - Using RDBMS, we must do Object Relational Mapping (ORM)
    - ORM is relatively demanding
  - JSON is much closer to structure of memory objects
    - It was originally for JavaScript objects
    - Object Document Mapping (ODM) is faster

Document Store Features

- Semi-structured data => optional schema
- Ad hoc queries (search by field, range, etc.)
- Often provide:
  - RESTful API
  - Data aggregation API (e.g., through Map/Reduce)
  - Full text search
  - Secondary indexes
  - Automatic sharding (scale writes)

- … but are typically missing:
  - Explicit locks
  - Strong consistency guarantees
  - Transactions
  - Joins
  - *Literally* No SQL

Flexible Schemas

Country	France
Region	Île-de-France
Department	Paris
Intercommunality	Métropole du Grand Paris
Subdivisions	20 arrondissements
Government	
• **Mayor** (2020–2026)	Anne Hidalgo[1] (PS)
Area[1]	105.4 km^2 (40.7 sq mi)
• **Urban** (2020)	2,853.5 km^2 (1,101.7 sq mi)
• **Metro** (2020)	18,940.7 km^2 (7,313.0 sq mi)
Population (Jan. 2019)[2]	2,165,423
• **Density**	21,000/km^2 (53,000/sq mi)
• **Urban** (2019[3])	10,858,852
• **Urban density**	3,800/km^2 (9,900/sq mi)
• **Metro** (Jan. 2017[4])	13,024,518
• **Metro density**	690/km^2 (1,800/sq mi)
Demonym(s)	Parisian(s) (en) *Parisien(s)* (masc.), *Parisienne(s)* (fem.) (fr), *Parigot(s)* (masc.), "Parigote(s)" (fem.) (fr, colloquial)
Time zone	UTC+01:00 (CET)
• **Summer (DST)**	UTC+02:00 (CEST)
INSEE/Postal code	75056 /75001-75020, 75116
Elevation	28–131 m (92–430 ft) (avg. 78 m or 256 ft)
Website	www.paris.fr

Abu Dhabi

Country	United Arab Emirates
Emirate	Abu Dhabi
Municipal region	Central Capital District[1]
Government	
• **Type**	Municipality
• **Body**	Abu Dhabi City Municipality
• **Director-General of City Municipality**	Saif Badr al-Qubaisi
Area	
• **Total**	972 km^2 (375 sq mi)
Elevation	27 m (89 ft)
Population (2021)[2] [3]	
• **Total**	1,512,000
• **Density**	1,600/km^2 (4,000/sq mi)
Demonyms	Abu Dhabian, Dhabyani
Time zone	UTC+4 (UAE Standard Time)
GDP PPP	2014 estimate
Total	US$ 178 billion[4]
Per capita	US$ 61,000
Website	tamm.abudhabi

Abu Dhabi

Country	United States
Residence Act	1790
Organized	1801
Consolidated	1871
Home Rule Act	1973
Named for	George Washington, Christopher Columbus
Government	
• **Mayor**	Muriel Bowser (D)
• **D.C. Council**	**List** [show]
• **U.S. House**	Eleanor Holmes Norton (D), Delegate (At-large)
Area	
• **Federal capital city and federal district**	68.35 sq mi (177.0 km^2)
• **Land**	61.126 sq mi (158.32 km^2)
• **Water**	7.224 sq mi (18.71 km^2)
Highest elevation	409 ft (125 m)
Lowest elevation	0 ft (0 m)
Population (2020)[2]	
• **Federal capital city and federal district**	689,545
• **Estimate** (2021)[2]	670,050
• **Rank**	23rd in the United States
• **Density**	11,280.71/sq mi (4,355.39/km^2)
• **Urban**[3]	5,174,759 (US: 8th)
• **Urban density**	3,997.5/sq mi (1,543.4/km^2)
• **Metro**[4]	6,385,162 (US: 6th)
Demonym	Washingtonian[5][6]
Time zone	UTC–5 (EST)
• **Summer (DST)**	UTC–4 (EDT)
ZIP Codes	20001–20098, 20201–20599, 56901–56999
Area code(s)	202, 771 (overlay)[7][8]
International airports	Dulles International Reagan National Baltimore/Washington
Commuter rail	MARC Train Virginia Railway Express
Rapid transit	Washington Metro
Website	dc.gov

Washington D.C

Document Example

```
{
        FirstName:"Camilla",
        Address:"Saadiyat",
        Children:[
            { Name:"Mike",   Age:10},
            { Name:"Jennie",  Age:8},
            { Name:"Samantha",  Age:5},
            { Name:"Ines",     Age:2}
        ]
},
{
        FirstName:"Omar",
        Address:"Yas Island",
        Hobby:"sailing"
}
```

Can be serialized in CSV, JSON, XML, etc.

JSON

- JSON: Text-based open standard for data interchange Serializing and transmitting structured data
  - JSON = JavaScript Object Notation Derived from JavaScript scripting language Uses conventions of the C-family of languages
  - Language independent (see www.json.org)

- BSON (Binary JSON) is a binary-encoded serialization format used to store and transfer data in a more compact and efficient manner compared to JSON.

NOSQL Document Stores

MONGODB

** Illustrative figures from mongodb.com*

Data Model

- Collections of documents (analogue of a table)
- BSON documents (attribute-value pairs with nesting and arrays)
- Documents can reference each other
  - Apps must issue follow-up queries to resolve the references
- Documents can be embedded in each other (i.e., nested)
  - But then we have to worry about documents getting very large

Data Model (cont.)

- Documents have flexible schema
  - Collections do not enforce specific data structure
- Related data in a single document structure
  - Documents can have subdocuments (in a field or array)
- Key decision of data modeling:
  - References vs. embedded documents
  - In other words: another trade-off
    - Locality of data
    - Relationships between data

Relational vs. MongoDB

Relational	MongoDB Model
Database	Database
Table	Collection
Tuple	Document (BSON)
Row_id	_id
Column	Field in the document

Each JSON document:

- belongs to a collection
- has a field _id
  - Unique within the collection
  - Primary Key
  - Immutable
  - Can be generated automatically

Operations

- Javascript API and Javascript shell
- Querying:Selection
  - A query targets a collection of documents
  - Selection queries on attribute values (including arrays)
    db.inventory.find({ type: "snacks" })
  - Can also have conditions on embedded documents
  - Can also do projections, sort, limit and skip
- Querying:Aggregation
  - Aggregation pipelines db.orders.distinct(“cust_id”)
  - MapReduce API (JavaScript map/reduce functions)

Operation Examples

```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

collection
query criteria
projection
cursor modifier

```
db.users.insertOne(
   {
      name: "sue",
      age: 26,
      status: "pending"
   }
)
```

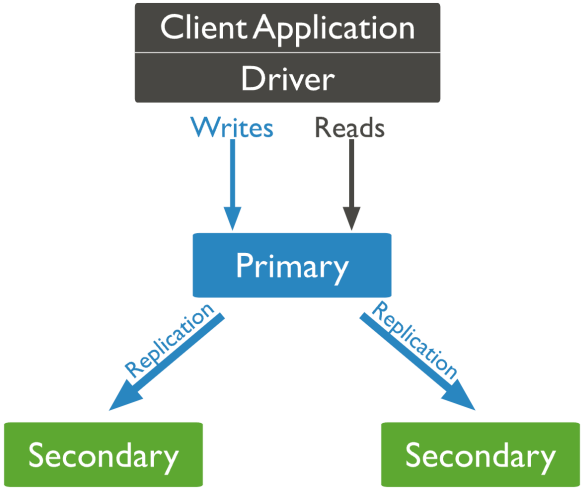
collection
field: value
field: value
field: value
document

```
db.users.updateMany(
   { age: { $lt: 18 } },
   { $set: { status: "reject" } }
)
```

collection
update filter
update action

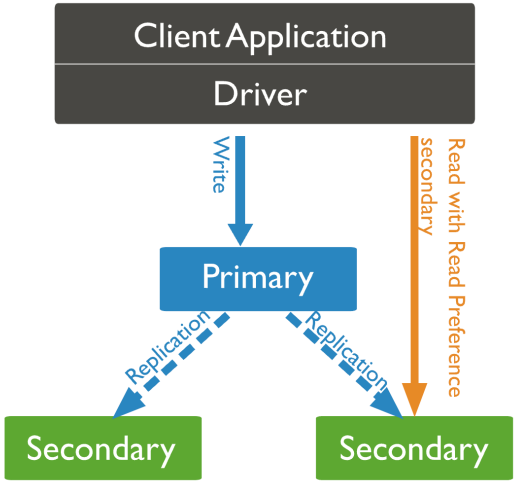
Replication in MongoDB

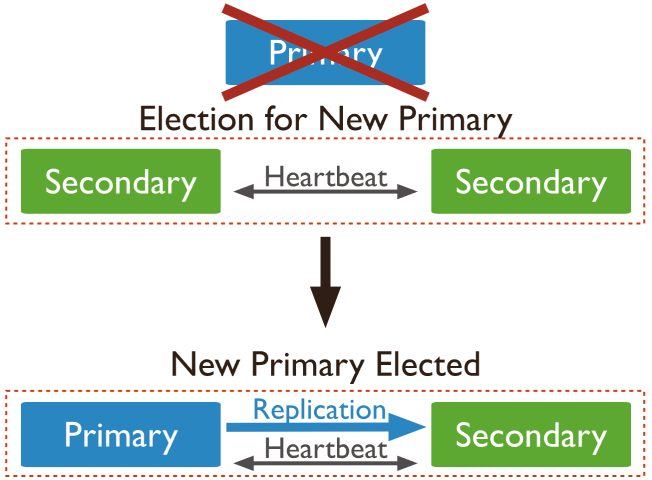
- MongoDB uses asynchronous replication for high availability.
- A Replica Set is a group of nodes that maintain the same data (max 50)
- The primary node is the master node that receives all write operations.
  - The primary records all changes to its data sets in its operation log or oplog (similar to a commit log)
- Secondary nodes replicate the primary's oplog and apply the operations to their data sets.
- **Strongly consistent system.** Once a write completes, any subsequent read will return the most recent value.



Replication in MongoDB

- **Read Preference:** MongoDB client can route some or all reads to the secondary members optionally, but writes must be sent to the primary.
- **Failover:** If the primary is unavailable, an eligible secondary will hold an election to elect itself the new primary.





Replication and Consistency

- Consistency
  - Write operations are atomic at the document level
  - Operations that modify more than a single document in a collection still operate on one document at a time

- Replication
  - Master scheme with master performing all reads & writes by default
    - Achieves strong consistency by always going through the master
  - Eventual consistency by default when reading from replicas
    - Updates propagate to replicas asynchronously
    - But can be configured to use synchronous replication

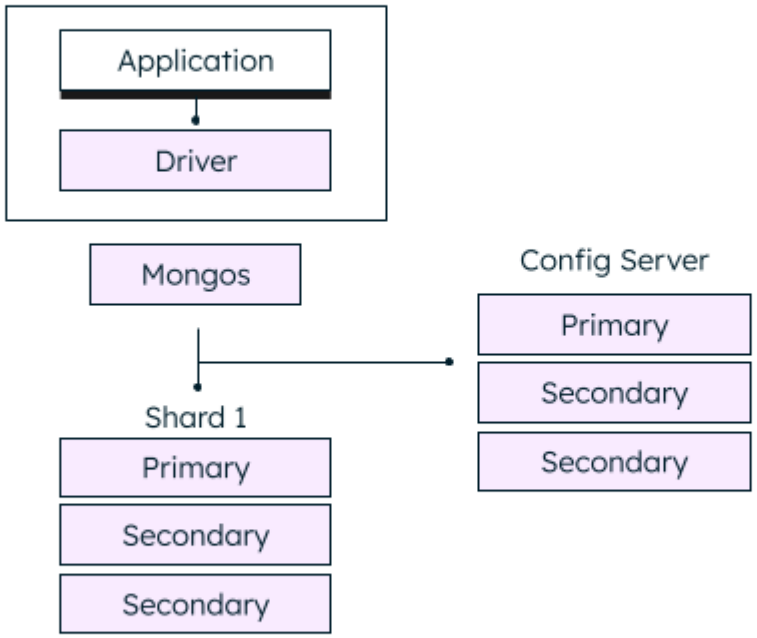
Sharding in MongoDB

- Sharding? Partitioning? 🤔
  - Sharding is the concept of splitting a databasex table (in mongo's case a collection) across multiple machines.

- MongoDB implements sharding using the following components:
  - shard: Each shard contains a subset of the sharded data. Each shard can be deployed as a replica set.
  - mongos: The mongos acts as a query router, providing an interface between client applications and the sharded cluster.
  - config servers: Config servers store metadata and configuration settings for the cluster. Deployed as replica set.

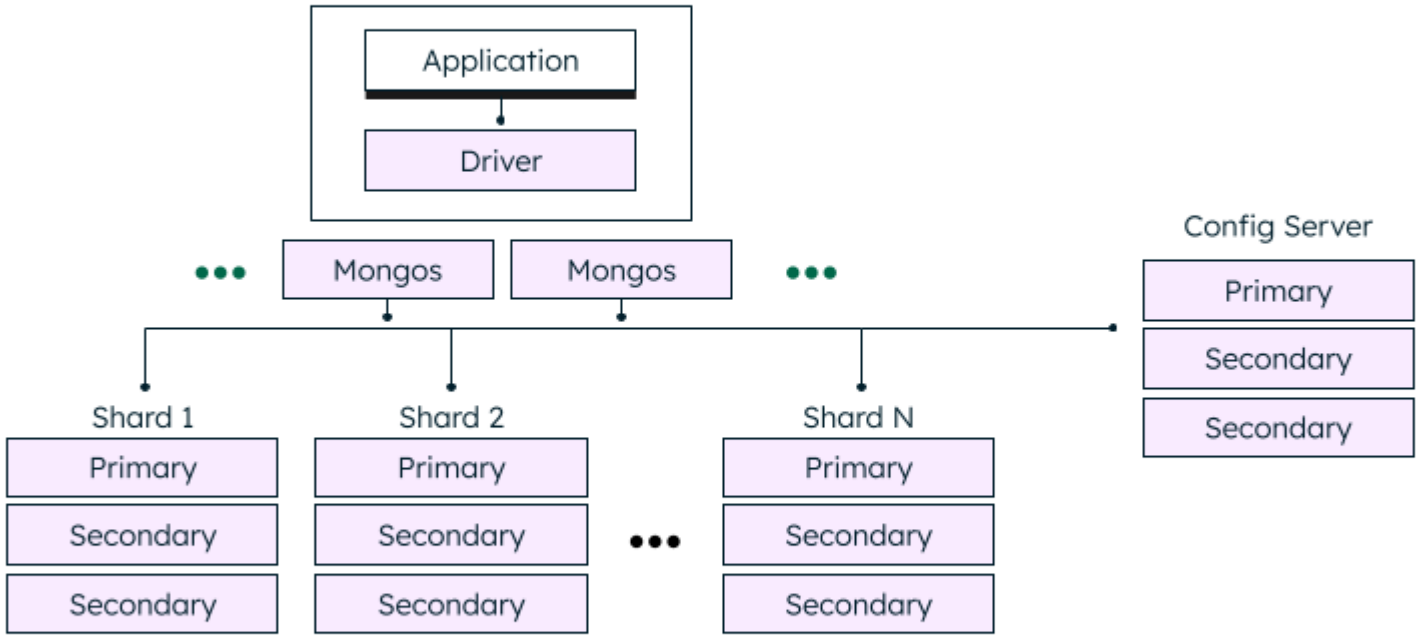
- User selects the "shard key"

MongoDB Architecture



Local Deployment

MongoDB Architecture



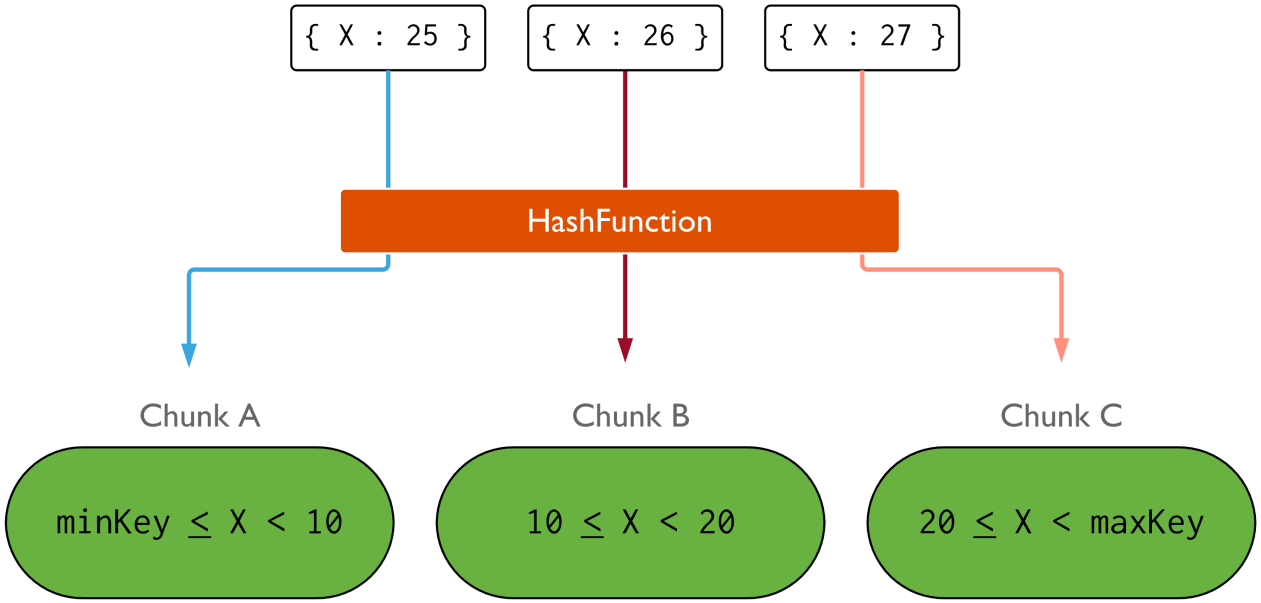
Cluster Deployment

Tuning Consistency

- Set read preference to secondary nodes.
  - Primary
  - Secondary
  - Nearest
  - Hedged reads
- Write Concern
  - { w: <value>, j: <written to journal>, wtimeout: <timeout> }
- Read Concern
  - “Local” (most recent data)
  - “Majority” (data has been replicated to a majority of secondaries)

Sharding Strategies in MongoDB

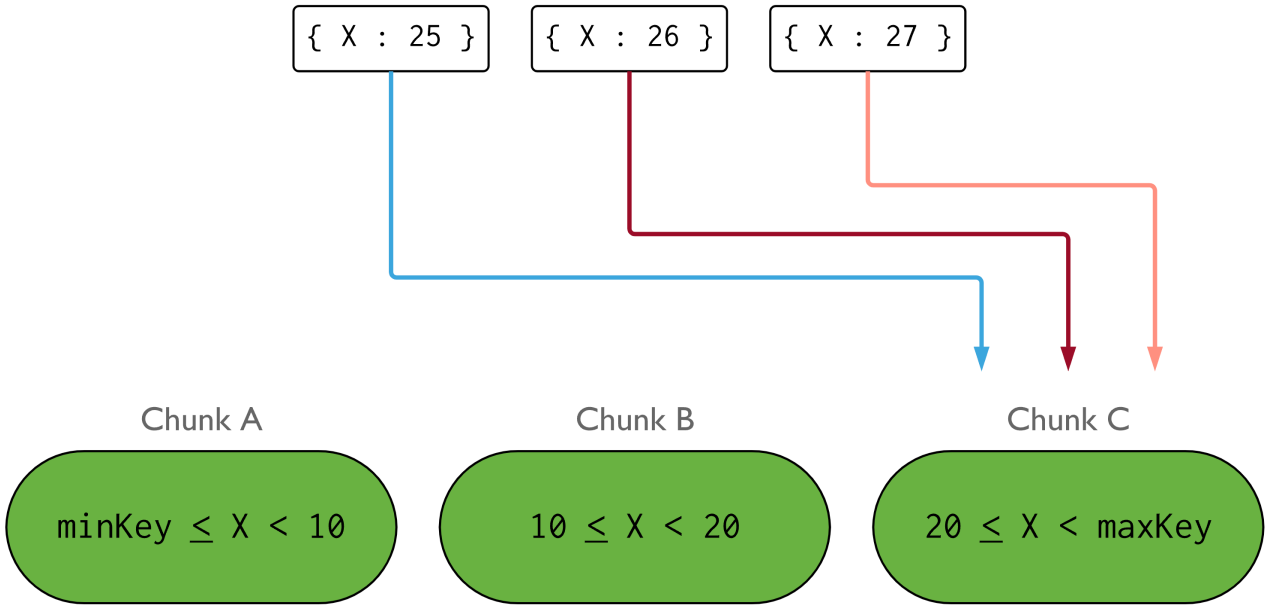
- Hash Sharding




```
sh.shardCollection("<database>.<collection>", { <shard key field> : "hashed" } )
```

Sharding Strategies in MongoDB

- Range Sharding




```
sh.shardCollection("<database>.<collection>", { <shard key field> : 1 } )
```

Storage Engines

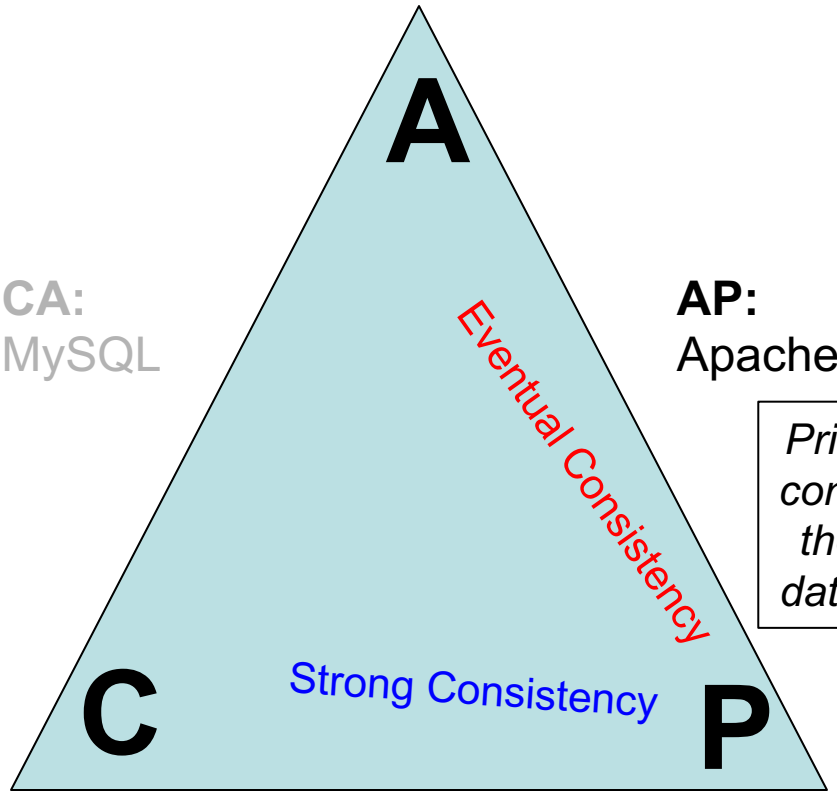
- MMAPv1 storage engine (historical)
  - Filesystem mmap: volume inserts, reads, and in-place updates.
- WiredTiger (starting version 3.2) which provides:
  - Document-level concurrency control for write operations
  - Optimistic concurrency control
  - Durability:
    - Multi-version concurrency control (MVCC)
    - Write ahead log (WAL), or Journal

Additional Concepts

- Indexing
  - MongoDB automatically indexes the **_id** field
  - Users can add indexes on other attributes (secondary index)
  - Geospatial Indexing
- Data modeling techniques: Document nesting vs. Referencing
- More features
  - Can insert document with a “Time To Live” to expire date
  - FIFO management of inserted docs and efficiently support operations that insert/read docs based on insertion order (Capped collections)

```
db.createCollection("recent", { capped : true, max : 10000 } )
```

NOSQL



CA:
MySQL

AP:
Apache Cassandra

Prioritize availability over data consistency, which means that they may return inconsistent data to ensure high availability

CP: MongoDB

To maintain data consistency, the system sacrifices availability by rejecting the request or cancelling it.

NoSQL - Takeaways

- Understand the rough limits of systems based on their architecture, data models, tradeoff
- MongoDB vs Cassandra
  - Data Model: MongoDB stores data in JSON-like format, while Cassandra uses a column-family (rows and columns)
  - Query: MongoDB has a rich query language and supports ad-hoc queries, while Cassandra has a limited query language and requires data to be modeled based on the queries to be performed.
  - Scalability: Both databases are horizontally scalable, but Cassandra is designed to handle massive amounts of data across multiple nodes more efficiently (decentralized).
  - Consistency: MongoDB provides strong consistency by default, while Cassandra provides eventual consistency by default. Both allow for tunable consistency levels.

Installation

- Install MongoDB
  - On MacOS: `brew install mongodb-community@6.0`
  - Start the client: `./mongosh`

- More info:
  - http://cassandra.apache.org