

# **FACT SHEET**

# DATA MODEL AND SCHEMA

- MongoDB is a document-based data model
- Data is stored in BSON (Binary JSON), which allows for a schemaless design
- Documents consist of key-value pairs
- Supports complex data structures like arrays and nested documents Unlike relational databases, collections do not enforce a fixed

# CONSISTENCY AND REPLICATION

#### Theorem & Consistency

schema'

MongoDB is a NoSQL database and follows <u>CAP theorem principles:</u>

#### Partition Tolerance (P):

Built to work in distributed environments

## Consistency vs. Availability Trade-off:

- Strong consistency when reading from the primary node
- Eventual consistency when reading from secondary nodes

#### Tunable Consistency:

- Developers can choose read and write concerns to balance consistency and availability
- "Majority" read concern ensures data consistency across multiple nodes

# **SECURITY**

#### - Authentication & Authorization:

- o Supports username/password authentication o Integrates with LDAP and x.509 certificate-based authentication
- o Role-Based Access Control (RBAC) for finegrained permissions

## - Encryption:

- o Encryption at Rest: Protects stored data
- o Encryption in Transit: Uses TLS/SSL to secure clientserver communication
- Auditing & Compliance: Provides detailed logging for security monitoring

## SPECIFIC USE CASES

- -Rapid Development & Schema Flexibility: Ideal for startups and evolving applications
- -Big Data & Real-Time Analytics:
- Supports high-velocity data processing
- -Content Management Systems:
- Stores complex and unstructured data
- -IoT & Mobile Applications:
- Handles large-scale, high-speed data ingestion
- -Geospatial & Search Applications: Provides built-in geospatial indexing and full-text search

# **EXTRA FEATURES**

- -Indexing: Supports various types of indexes (compound, geospatial, text, etc.) to optimize queries
- -Aggregation Framework: Provides a pipeline-based system for data transformation and analysis
- -Flexible Query Language: Supports rich queries, including filtering, sorting, and regex searches
- -Change Streams: Enables real-time data updates for applications
- -Replication and High Availability: Uses replica sets for failover and data redundancy



# **CLUSTERS**

#### **Replica Set**

- A replica set is a group of MongoDB servers that store identical copies of data.
- Purpose:
  - High availability (data is always accessible)
  - Redundancy (backup in case a server fails).
  - Handles failovers and maintenance with minimal downtime.
- Common in production environments.

### Sharded Cluster

- Also known as horizontal scaling.
- Data is split and distributed across multiple servers (shards).
- Purpose:
  - Scales read and write operations.
  - Useful when dealing with large datasets or high traffic.

## HISTORY

In 2009, MongoDB was officially released as an open-source project, allowing developers worldwide to leverage its features for free. The database quickly gained popularity due to its document-oriented design, scalability, and flexibility. Unlike traditional SQL databases, MongoDB provided a schema-less structure, making it particularly well-suited for rapidly evolving applications and big data workloads.

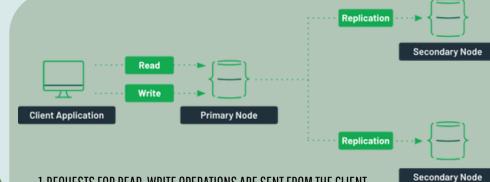
#### **Economic information**

MongoDB reported \$2.01 billion in revenue for fiscal year 2025, with its cloud service Atlas making up 68% of that, showing strong growth in the cloud database market. It holds a leading position among NoSQL databases and is used by major companies like Adobe, eBay, and Coinbase.'

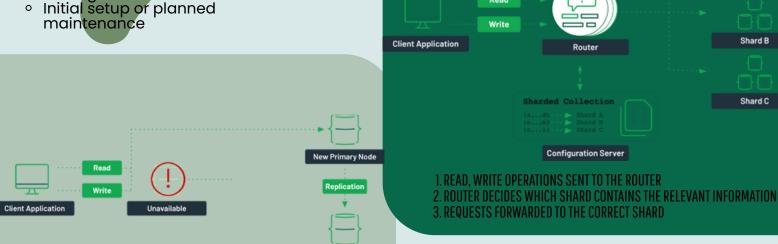
## OPLOG AND ELECTION PROCESS

- Oplog:
- special capped collection that stores a rolling log of operations.
- Used to sync secondary nodes with
- the primary.

  Dynamically resizes to avoid deleting important commits prematurely.
- **Election Process:**
- Triggered when the primary becomes unavailable.
- A new primary is elected from secondary nodes.
- Triggers include:
  - Node failure
  - Timeout >10 seconds (default)
  - Adding new nodes



- 1. REQUESTS FOR READ, WRITE OPERATIONS ARE SENT FROM THE CLIENT
- 2. PRIMARY NODE HANDLES OPERATIONS (OPLOG KEEPS TRACK OF OPERATIONS)
- 3. OPERATIONS REPLICATED IN SECONDARY NODES USING OPLOG



- 1. READ, WRITE OPERATION SENT FROM THE CLIENT TO THE PRIMARY NODE
- 2. PRIMARY NODE IS DETECTED AS NOT WORKING
- 3. SECONDARY NODE IS ELECTED AS THE NEW PRIMARY NODE (ELECTION PROCESS)
- 4. OPERATION IS EXECUTED ON THE NEW PRIMARY NODE



Shard A

Shard B

Shard C