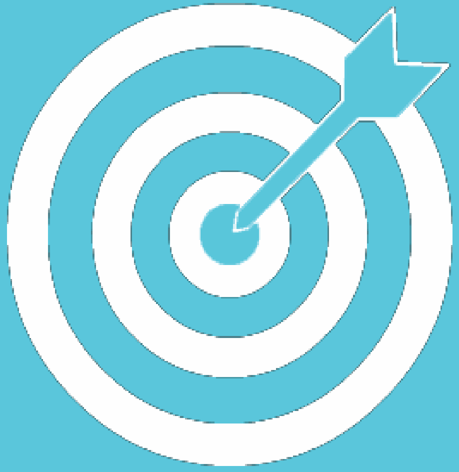
MongoDB Developer and Administrator Course

Features of mongodb

Anju Munoth

After completing this lesson, you will be able to:



- Describe what MongoDB is
- Identify the key features of MongoDB
- Explain MongoDB’s core server and tools
- Explain how to install MongoDB on Windows and Linux computers
- Identify the steps to start the MongoDB server
- Identify the data types available in MongoDB
- Identify the schema design and data modeling techniques in MongoDB

What is MongoDB?

MongoDB replaces the concept of ‘rows’ of traditional relational data models with ‘documents’. Following are two key characteristics of MongoDB:

Document Based

- Allows embedded documents, arrays, and represents a complex hierarchical relationship using a single record.

Schema Free

- The keys used in documents are not predefined or fixed. Without a fixed schema, massive data migrations have become unnecessary.

MongoDB as a Document Database

The data model in MongoDB is document-based, which does not conform to any pre-specified schema. The differences between a relational database and MongDB are as follows:

Relational Database

Rows are stored in a table and each table has a strictly defined schema that specifies the permitted types and columns. If a row in a table requires an extra field, the entire tabled needs to be altered.

MongoDB

Groups documents into collections that do not follow any schema. Each individual document in a collection can have a completely independent structure.

! A schema less model lets you represent data with variable properties.

JSON

JavaScript Object Notation (JSON) is an open data interchange format that:

- Is language independent.
- Uses conventions of the C-family of languages, Java, JavaScript, Perl, and Python.
- Supports the basic data types, such as numbers, strings, boolean values, arrays, and hashes.

JSON Structure

JSON is built on the following two structures:

1. A collection of name/value pairs, such as an object, record, struct, dictionary, hash table, keyed list, or associative array.
2. An ordered list of values, such as an array, vector, list, or sequence.

```
{
        "_id" : 1,
        "name" : {"MongoDB"},
        "customers" : ["Metlife", "OTTO", "Expedia", "ADP" ],
        "applications" : [
                {                        "name"   : "Forward",
                          "domain" : "e-commerce"
                },
                { "name"   : "Calipso",
                  "domain" : "content management"
                }
        ]
}
```

BSON

Lightweight

- When used over the network, BSONkeeps the overhead involved in processing extra header data to aminimum.

Traversable

- It is designed to traverse easily across network. This helps in its role asthe primary data representation for MongoDB.

Efficient

- It uses Cdata types that allow easyand quick encoding and decoding of data.

MongoDB Structure

The table and view structure is known as collection, which:

- Group documents that are structurally or conceptually similar.
- Embed related documents to query related data.
- Distribute data for a table among different shards similar to partition in Relational Database Management Systems (RDBMS).
- Are grouped into databases.

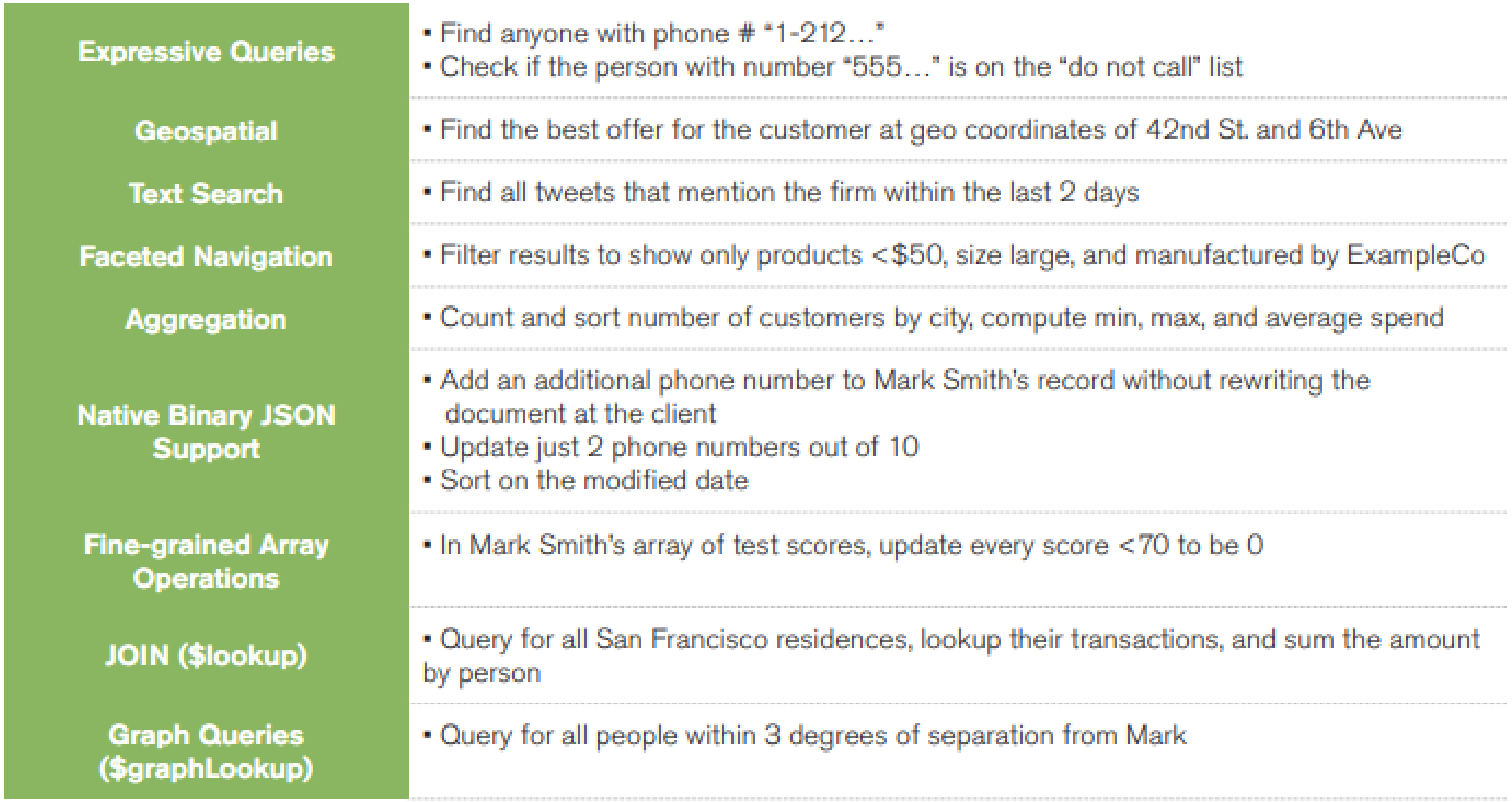
RDBMS	MongoDB
Database	Database
Table, View	Collection
Row	Document (JSON, BSON)
Column	Field
Index	Index
Join	Embedded Document
Foreign Key	Reference
Partition	Shard

! Data related to a single application should be stored at one database. Use separate databases when storing multiple application or users data on the same MongoDB server.

Document Store Example

```
> db.user.findOne({age:35})
{
        "_id" : ObjectId("5224e0bd52…"),
        "first" : “Mark",
        "last" : “Tailor",
        "age" : 35,
       "interests" : [
                “long distance running",
                "Mountain Biking ]
                "favorites": {
               "color": “Yellow",
               "sport": “Boxing"}
```

: MongoDB’s rich query functionality



Expressive Queries	▪ Find anyone with phone # “1-212…” ▪ Check if the person with number “555…” is on the “do not call” list
Geospatial	▪ Find the best offer for the customer at geo coordinates of 42nd St. and 6th Ave
Text Search	▪ Find all tweets that mention the firm within the last 2 days
Faceted Navigation	▪ Filter results to show only products <$50, size large, and manufactured by ExampleCo
Aggregation	▪ Count and sort number of customers by city, compute min, max, and average spend
Native Binary JSON Support	▪ Add an additional phone number to Mark Smith’s record without rewriting the document at the client ▪ Update just 2 phone numbers out of 10 ▪ Sort on the modified date
Fine-grained Array Operations	▪ In Mark Smith’s array of test scores, update every score <70 to be 0
JOIN ($lookup)	▪ Query for all San Francisco residences, lookup their transactions, and sum the amount by person
Graph Queries ($graphLookup)	▪ Query for all people within 3 degrees of separation from Mark

Working with Document Data

- To accelerate developer productivity, MongoDB provides native drivers for all popular programming languages and frameworks. Supported drivers include Java, Javascript, C#/.NET, Go, Python, PHP, Rust, Scala and others.

- All supported MongoDB drivers are designed to be idiomatic for the given programming language – with syntax and behaviors that are familiar to developers using those languages.

- This makes it much more natural for you to work with the database than string-based languages like SQL.

Tools and Connectors

- Vital that developers, business analysts, and data scientists can extract insights from data – whether that is for traditional reporting and BI or to build more intelligent applications with machine learning.
- MongoDB provides a range of visualization tools and connectors to make this straightforward:
- **MongoDB Charts** is the fastest and easiest way to create visualizations of MongoDB data.
- Can construct graphs and build dashboards, sharing them with other users for collaboration, and embed them directly into your web apps to create engaging, data-driven user experiences
- **MongoDB Connector for BI** lets you use MongoDB as a data source for your existing SQL-based BI and analytics platforms such as Tableau, Microstrategy, Looker, Excel, and more, without having to perform any ETL operations.
- **MongoDB Connector for Apache Spark** exposes MongoDB data to all of Spark's libraries, including Scala, Java, Python and R. MongoDB data is materialized as DataFrames and Datasets for integration with machine learning frameworks.

Tools and Connectors

- MongoDB goes beyond many other databases with features like **Change Streams** that allow applications to access real-time data changes in the database,

- **MongoDB Atlas Triggers** allows to execute server-side logic in response to database change events.

- This might be updating related data in other documents, or calling functions that, for example, send an email to a new customer when they sign up to your service, or firing an alert to a user’s mobile app when a large charge is made to their account.

- With **the MongoDB Connector for Apache Kafka**, can build robust data pipelines that move events between systems in real time, using MongoDB as both a source and sink for Kafka. The connector is supported by MongoDB and verified by Confluent.

ACID Transactions

- Most document databases offer atomic guarantees for writes to a single document.

- Suitable for many applications because the document model brings together related data that would otherwise be modeled across separate parent-child tables in a tabular schema.

- With single document atomicity, one or more fields may be written in a single operation, including updates to multiple subdocuments and elements of an array.

- These guarantees ensure complete isolation as the document is updated; any errors cause the operation to roll back so that clients receive a consistent view of the document

- However not every application can be served with atomicity that is scoped to only a single document.
- This is why some document databases go further by offering support for multi-document ACID transactions.
- Many implement multi-document transactions via server-side stored procedures or in client-side code.

- MongoDB takes a different approach by implementing multi-document transactions in a way that is consistent with relational databases

Transaction Management in MongoDB

Transactions in MongoDB are as follows:

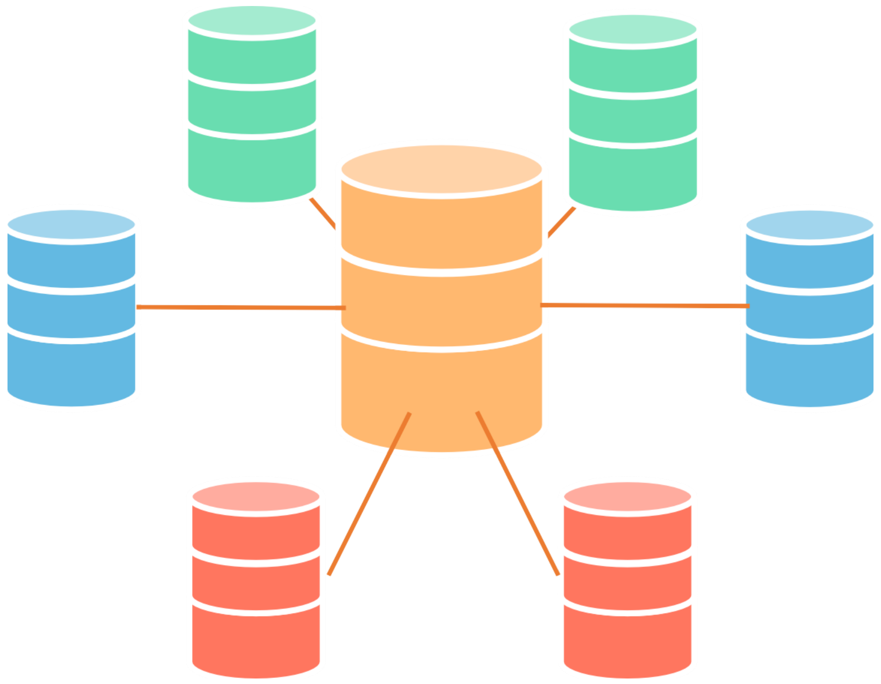
- MongoDB supports single phase commit at each document level.
- A write operation in a single document is considered atomic in MongoDB even if the operation alters multiple embedded documents.

! A write operation is atomic but the entire operation is not atomic and other operations may interleave.

Easy Scaling

Data set sizes are growing with the growing technology. This has created a demand for:

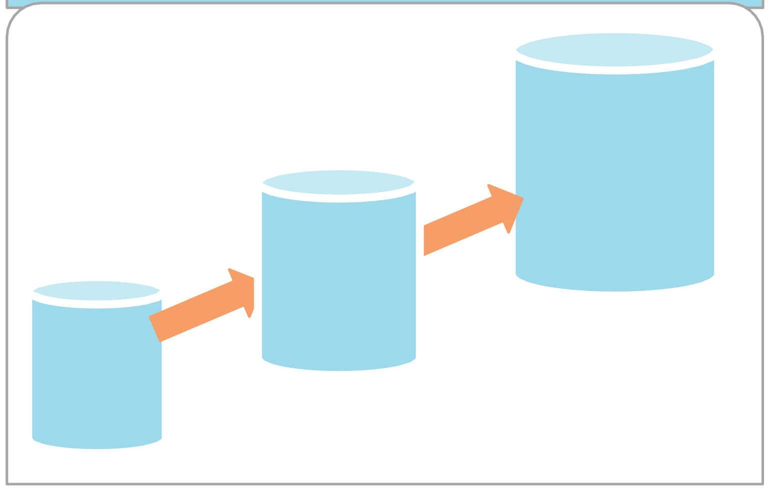
- More data storage with more databases capabilities.
- Scaling databases for size and performance.



Scaling Up vs. Scaling Out

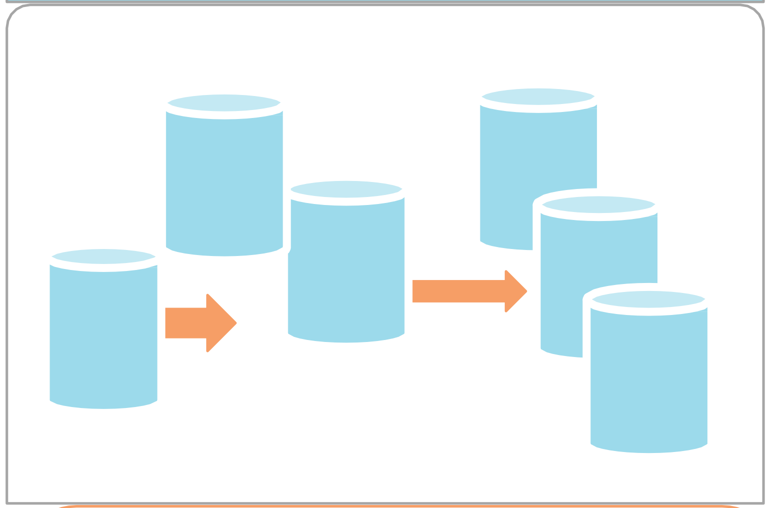
Scaling a database involves two choices:

Scaling Up



- Easy but costly affair
- Even the most powerful database may not be able to manage growing volumes of data

Scaling Out



- Extensible and cost-effective
- Commodity server can be added to increase storage and enhance performance

Vertical Scaling

A database can be scaled by upgrading the hardware. The technique of enhancing a single node hardware is called vertical scaling or scaling up. Vertical scaling is:

- Simple
- Reliable
- Cost-effective to some extent

! If your database is running on a physical hardware, and the cost of a more powerful server is not permitted, consider horizontal scaling.

Horizontal Scaling

Horizontal scaling distributes the database across multiple machines. The advantages include:

- Commodity hardware are used.
- Mitigates the risk of failure.
- Failure is less disastrous because a single machine is only a small part of the entire system.

MongoDB supports horizontal scaling and uses a range-based partitioning mechanism called ‘auto-sharding’ which:

- Automatically manages data distribution across multiple nodes.
- Allows addition of new nodes.
- Is transparent to the client.

Multi-Cloud Global Database: Freedom and Flexibility

- To harness the tremendous rate of innovation in the cloud and reduce the risk of lock-in, project teams should build their applications on data platforms that deliver a consistent experience across any environment.

- MongoDB can be run anywhere – from developer laptops to mainframes, from private clouds to the public cloud.

- Broadest Reach: Private, Hybrid, Public Clouds

- With MongoDB, the developer experience is entirely unaffected by the chosen deployment model.

- Enables to take advantage of unique capabilities in each platform without changing a single line of application code and without the heavy lift and risk of complex database migrations.

MongoDB Atlas: Fully Automated Database as a Service

- MongoDB Atlas is the global cloud database service for modern applications.
- Can deploy fully managed MongoDB across AWS, Azure, or Google Cloud with best-in-class automation and proven practices that guarantee availability, scalability, and compliance with security standards
- MongoDB Atlas is available through a pay-as-you-go model and billed on an hourly basis.
- Easy to get started – use a simple GUI or programmatic API calls to select the public cloud provider, region, instance size, and features

MongoDB Atlas: Fully Automated Database as a Service

- ➢ Automated database and infrastructure provisioning along with auto-scaling so teams can get the database resources they need, when they need them, and can elastically scale in response to application demands.

- ➢ Always-on security features to protect your data, including network isolation, fine-grained access controls, auditing, and end-to-end encryption down to the level of individual fields.

- ➢ Certifications with global standards to help you achieve your compliance requirements, including ISO 27001, SOC 2, and more.

- ➢ Atlas can be used for workloads subject to HIPAA, PCI-DSS, or the GDPR.

- ➢ Built-in replication both within and across regions for always-on availability, even in the face of complete regional outages.

- ➢ Global Clusters for fully managed, globally distributed databases that provide low latency, responsive reads and writes to users anywhere, with strong data sovereignty controls for regulatory compliance.

MongoDB Atlas: Fully Automated Database as a Service

- Fully managed backups with point-in-time recovery to protect against data corruption, and the ability to query backups in-place without full restores.

- Fine-grained monitoring, real-time metrics, query profiler, and customizable alerts for comprehensive performance visibility.

- Intelligent schema and index recommendations with the Performance Advisor, which analyzes slow query logs of your database collections and ranks suggestions by impact to your database performance.

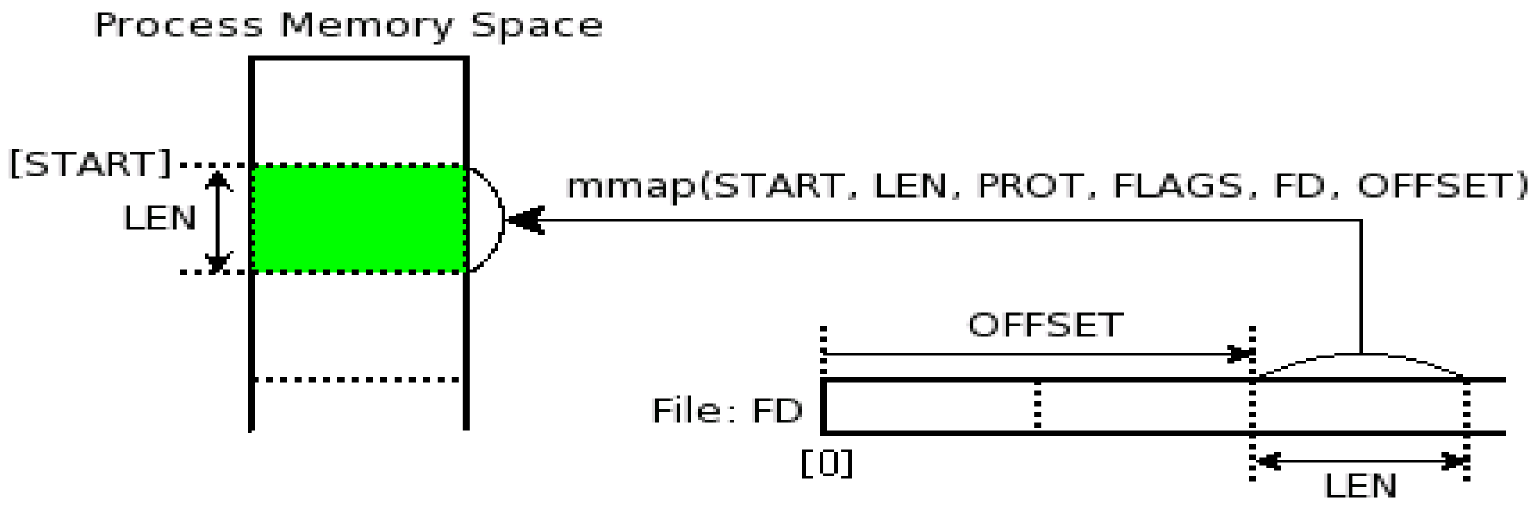
- Automated patching and single-click upgrades for new major versions of the database, enabling you to takeadvantage of the latest MongoDB features.

- Auto-archiving of aged data from your live database clusters to fully-managed cloud object storage with Online Archive.

- Federated query enables you to analyze your live MongoDB Atlas data and historical data on object storage together and in-place, with a single query, for faster insights.

Memory Management

MongoDB stores the data in memory mapped files and uses all the system memory for these mapped files for faster operations. MongoDB allows its operating system to manage its memory which impacts its performances and operations.



Secondary Indexes

MongoDB implements multiple secondary indexes as B-trees that can be optimized for range scan queries and queries with sort clauses.

MongoDB allows creation of up to 64 indexes per collection and supports indexes, such as ascending, descending, unique, compound-key, and geospatial.

! MongoDB uses the same data structure for indexes as most RDBMSs.

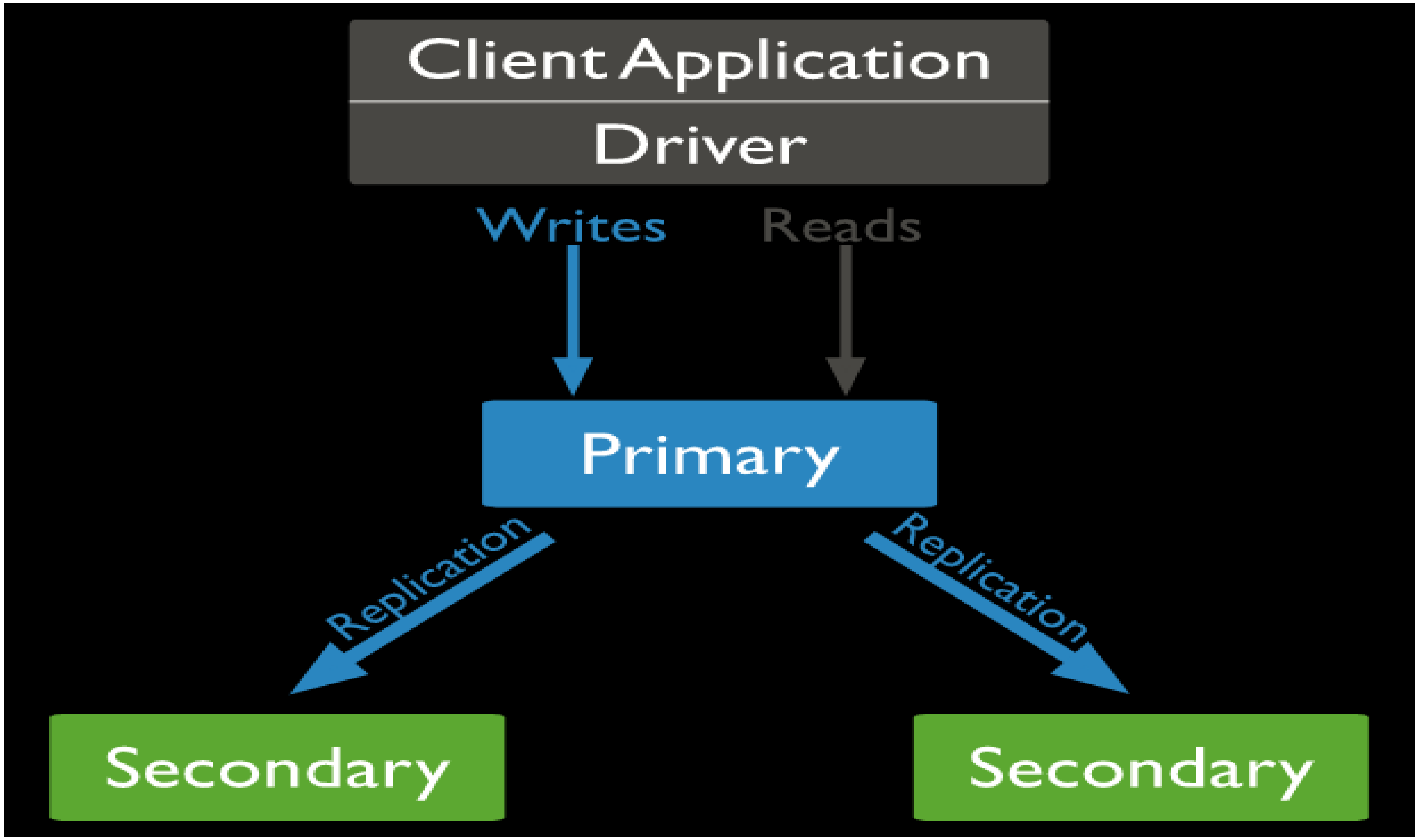
Replication

MongoDB uses replica sets to create database replication. A replica set performs the following actions:

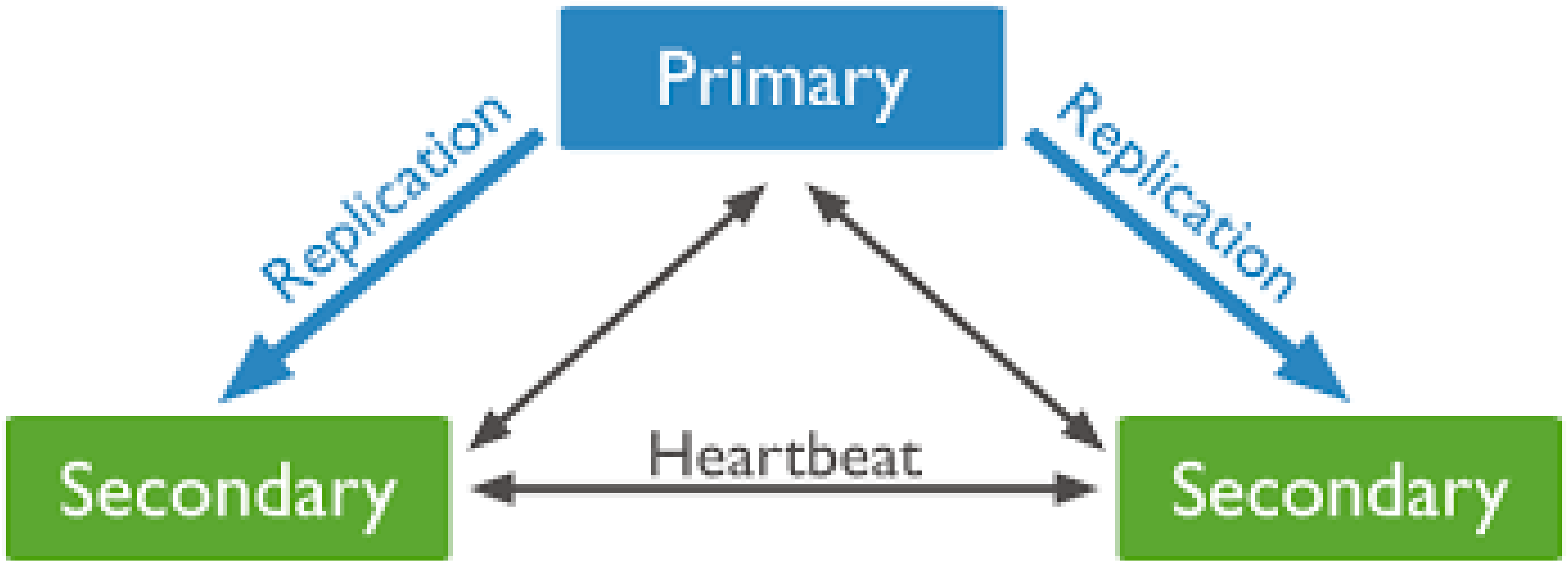
- Distributes data across various MongoDB nodes called shards for redundancy.
- Automates failover when server or network outages occur.
- Scales database reads.

A replicaset contains primary node and one or more secondary nodes, the primary node supports both read and write, whereas the secondary supports read.

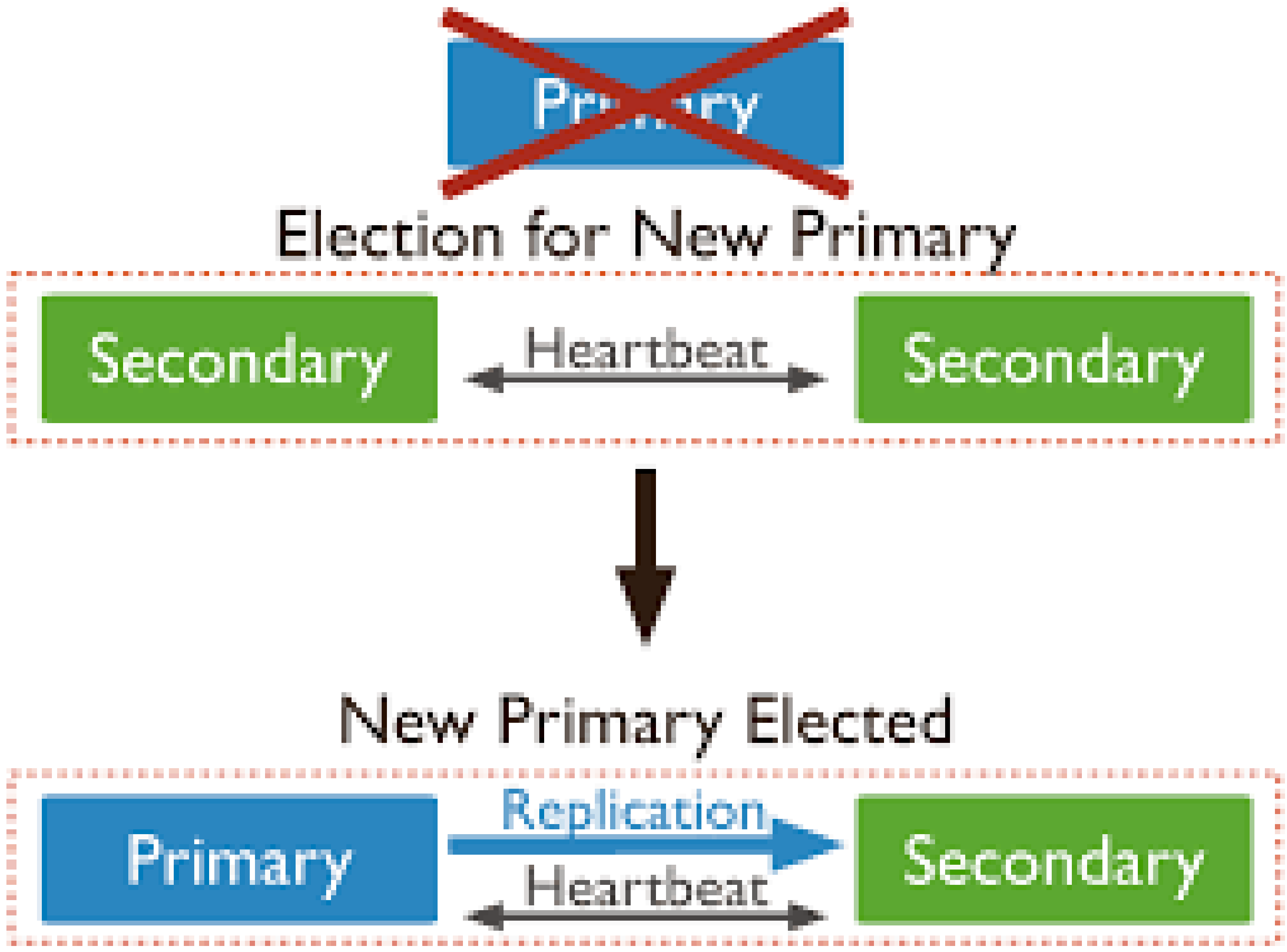
Replication



Replication



Replication



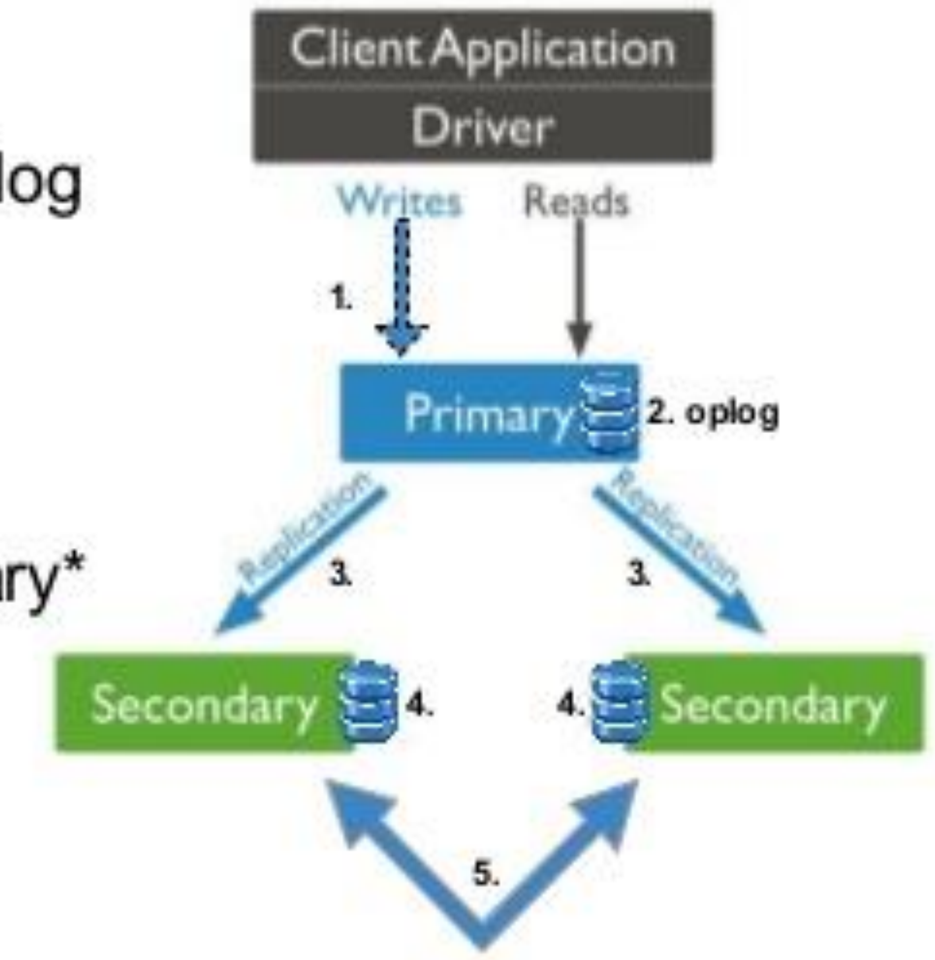
Replication

Replication concept

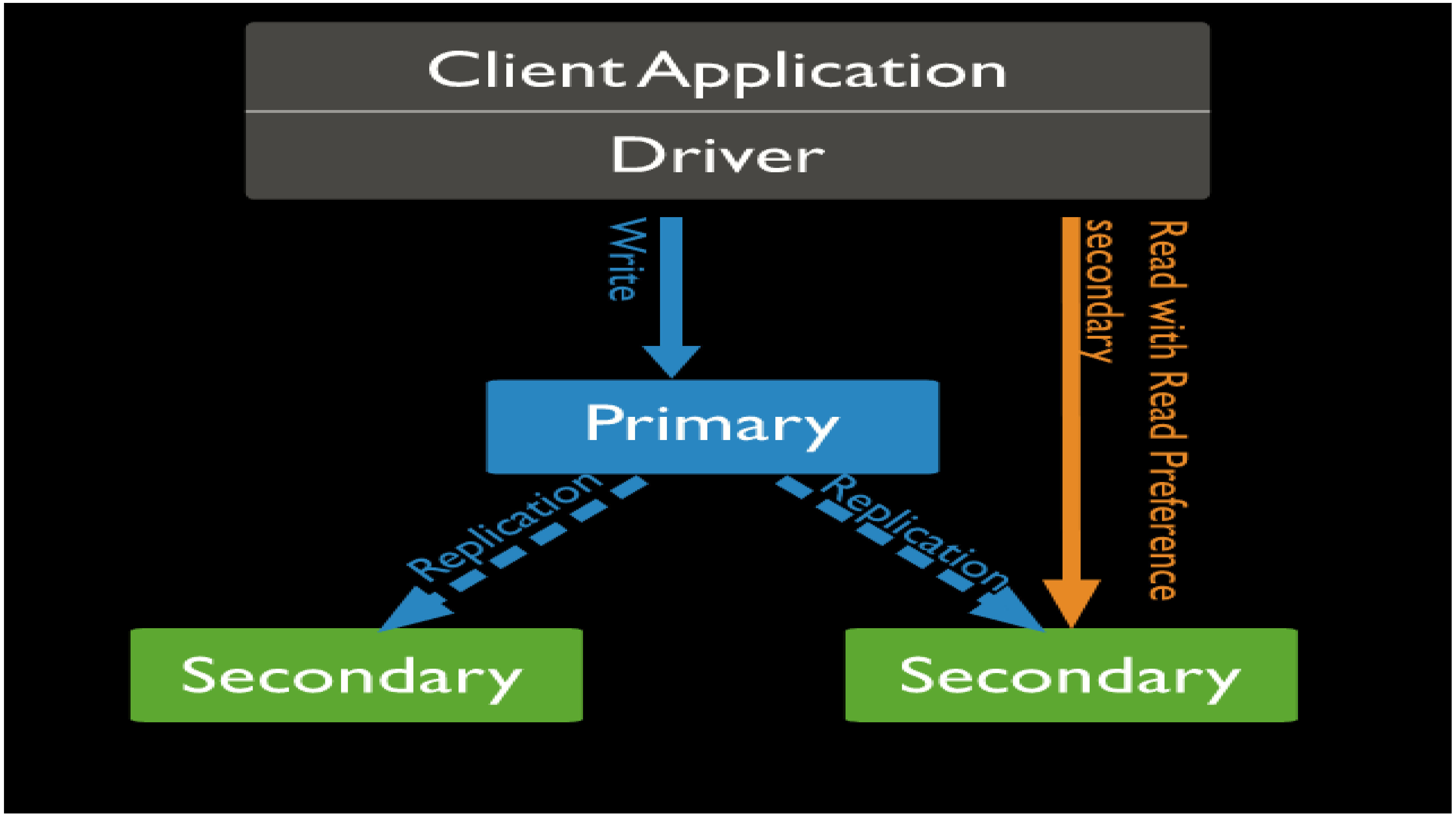
1. Write operations go to the Primary node
2. All changes are recorded into operations log
3. Asynchronous replication to Secondary
4. Secondaries copy the Primary oplog
5. Secondary can use sync source Secondary*

Automatic failover on Primary failure

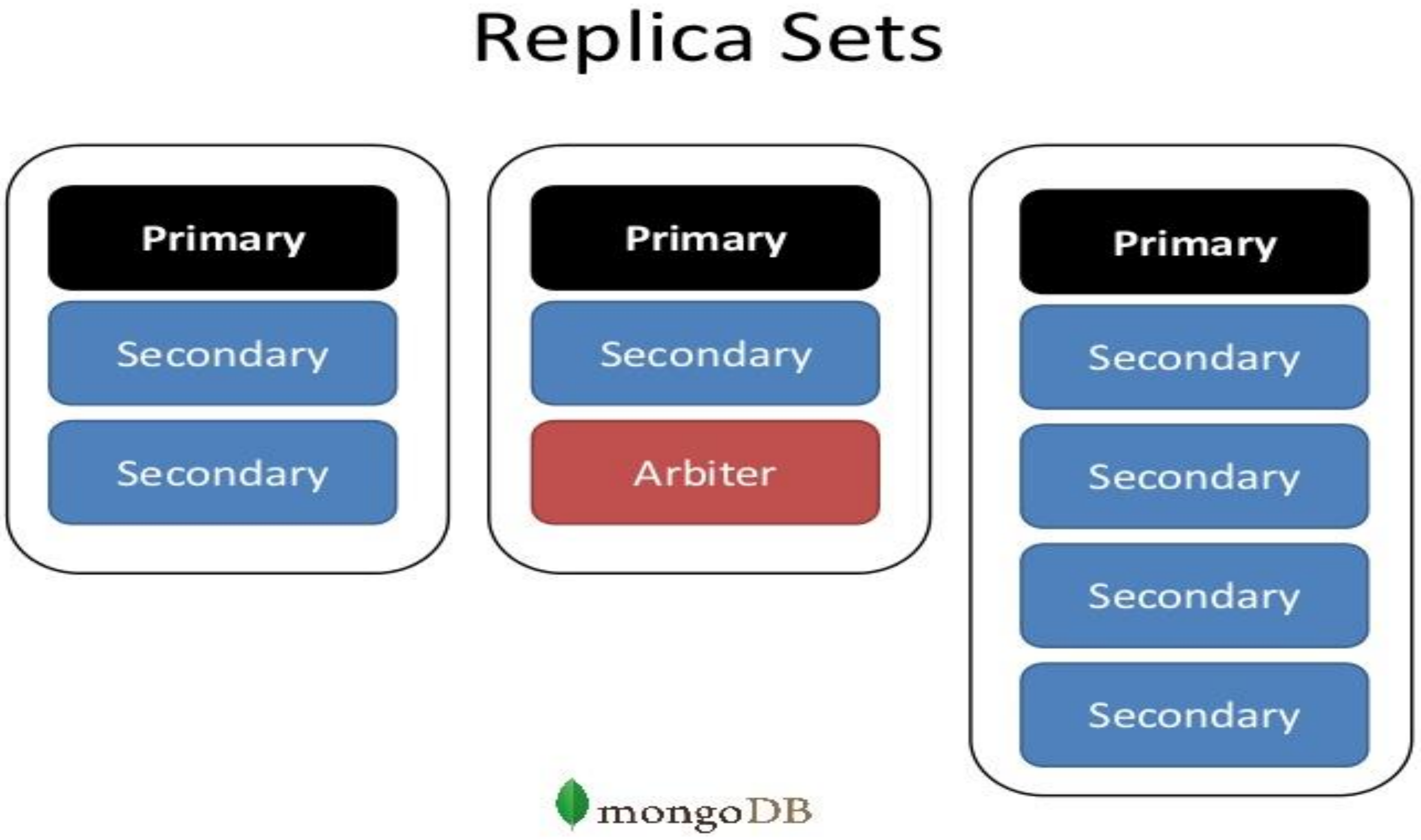
*settings.chainingAllowed (true by default)

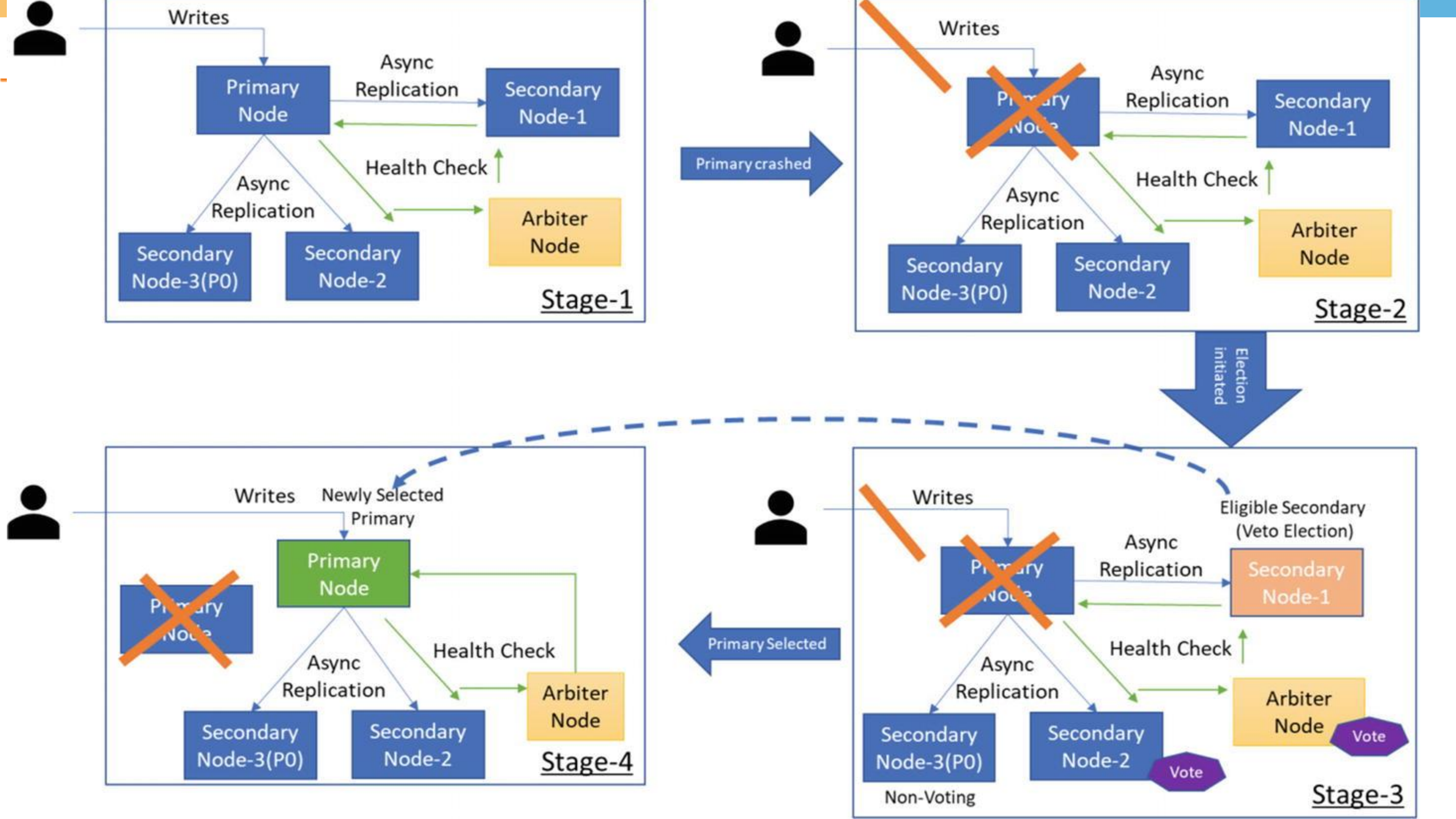


Replication

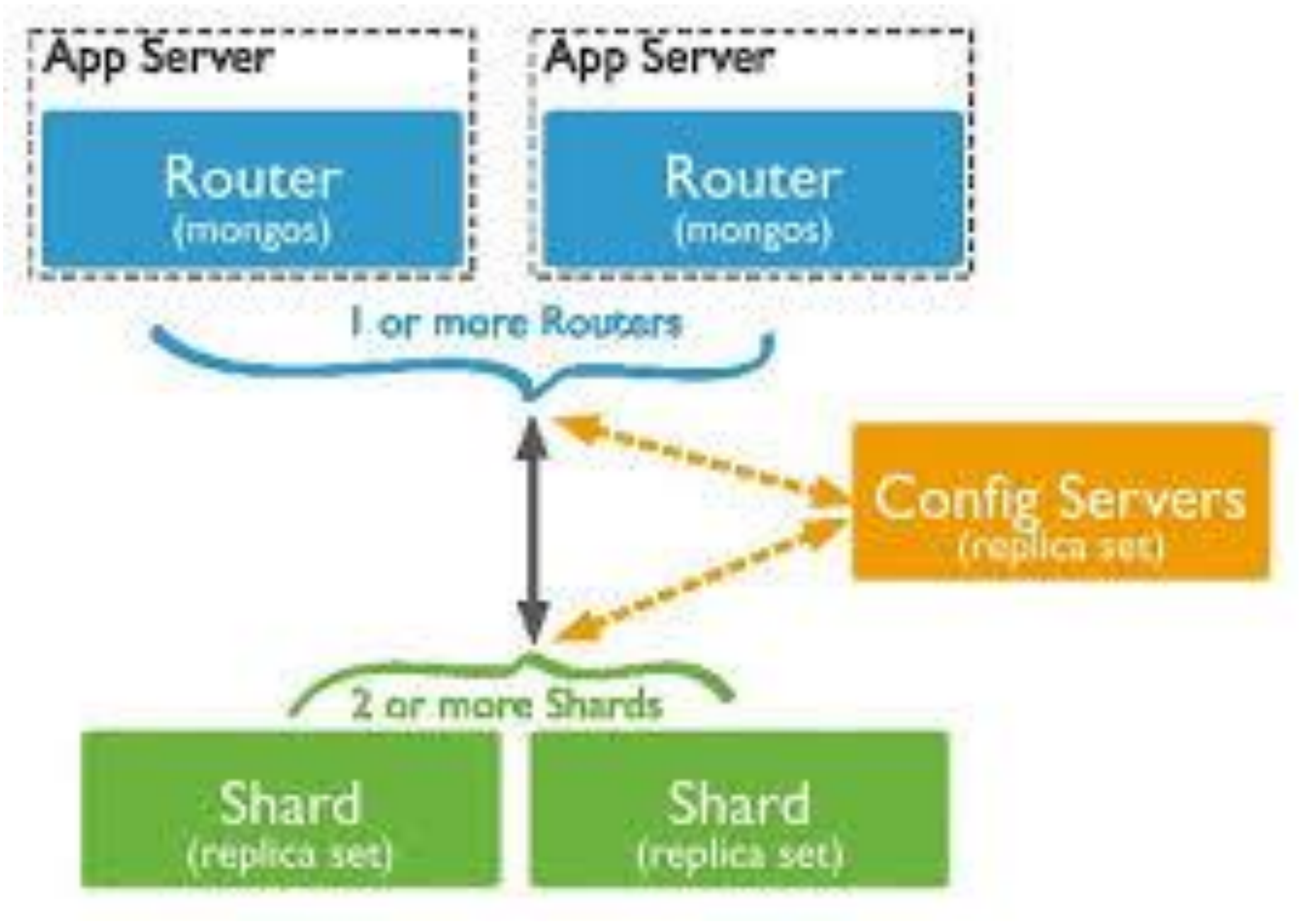


Replication

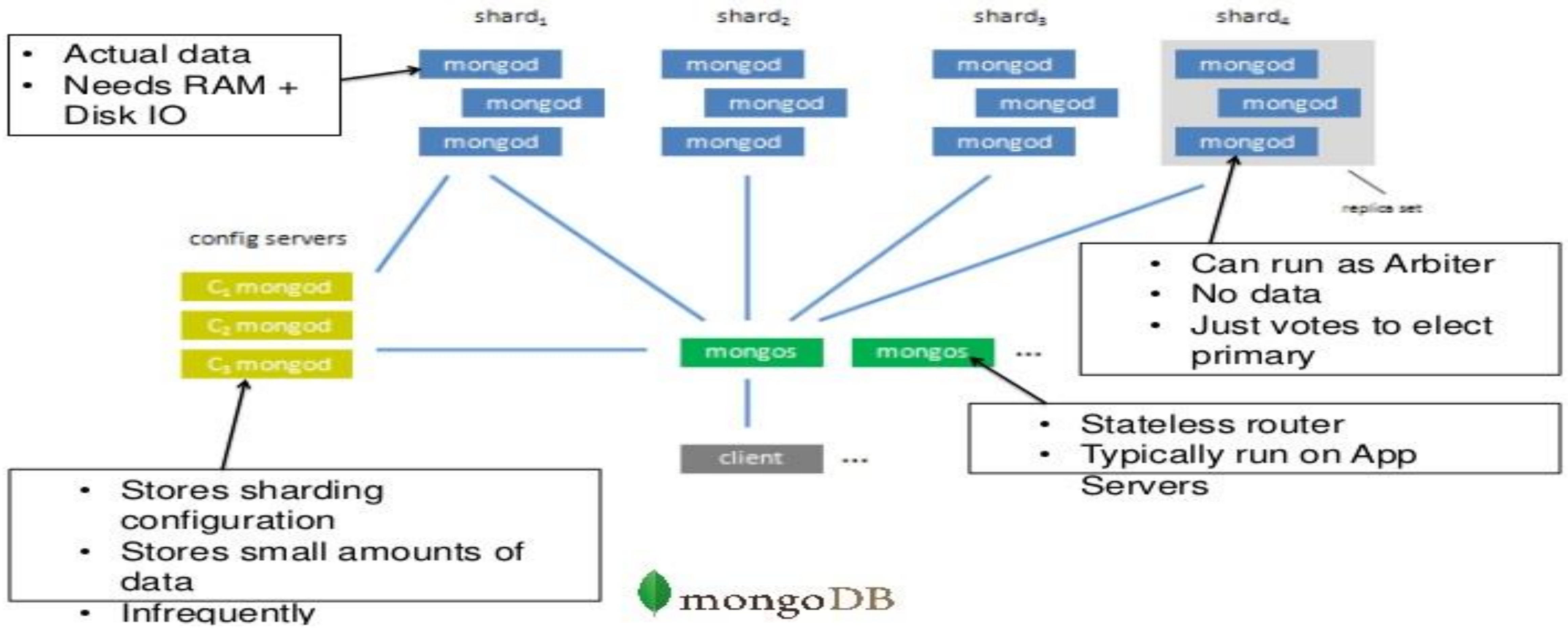



Writes
Primary Node
Async Replication
Secondary Node-1
Health Check
Async Replication
Secondary Node-3(P0)
Secondary Node-2
Arbiter Node
Stage-1
Primary crashed
Writes
Primary Node
Async Replication
Secondary Node-1
Health Check
Async Replication
Secondary Node-3(P0)
Secondary Node-2
Arbiter Node
Stage-2
Election initiated
Writes
Eligible Secondary (Veto Election)
Primary Node
Async Replication
Secondary Node-1
Health Check
Async Replication
Secondary Node-3(P0)
Non-Voting
Secondary Node-2
Vote
Arbiter Node
Vote
Stage-3
Primary Selected
Writes
Newly Selected Primary
Primary Node
Primary Node
Health Check
Async Replication
Secondary Node-3(P0)
Secondary Node-2
Arbiter Node
Stage-4

Sharding


App Server
Router
(mongos)
App Server
Router
(mongos)
1 or more Routers
Config Servers
(replica set)
2 or more Shards
Shard
(replica set)
Shard
(replica set)

MongoDB Components

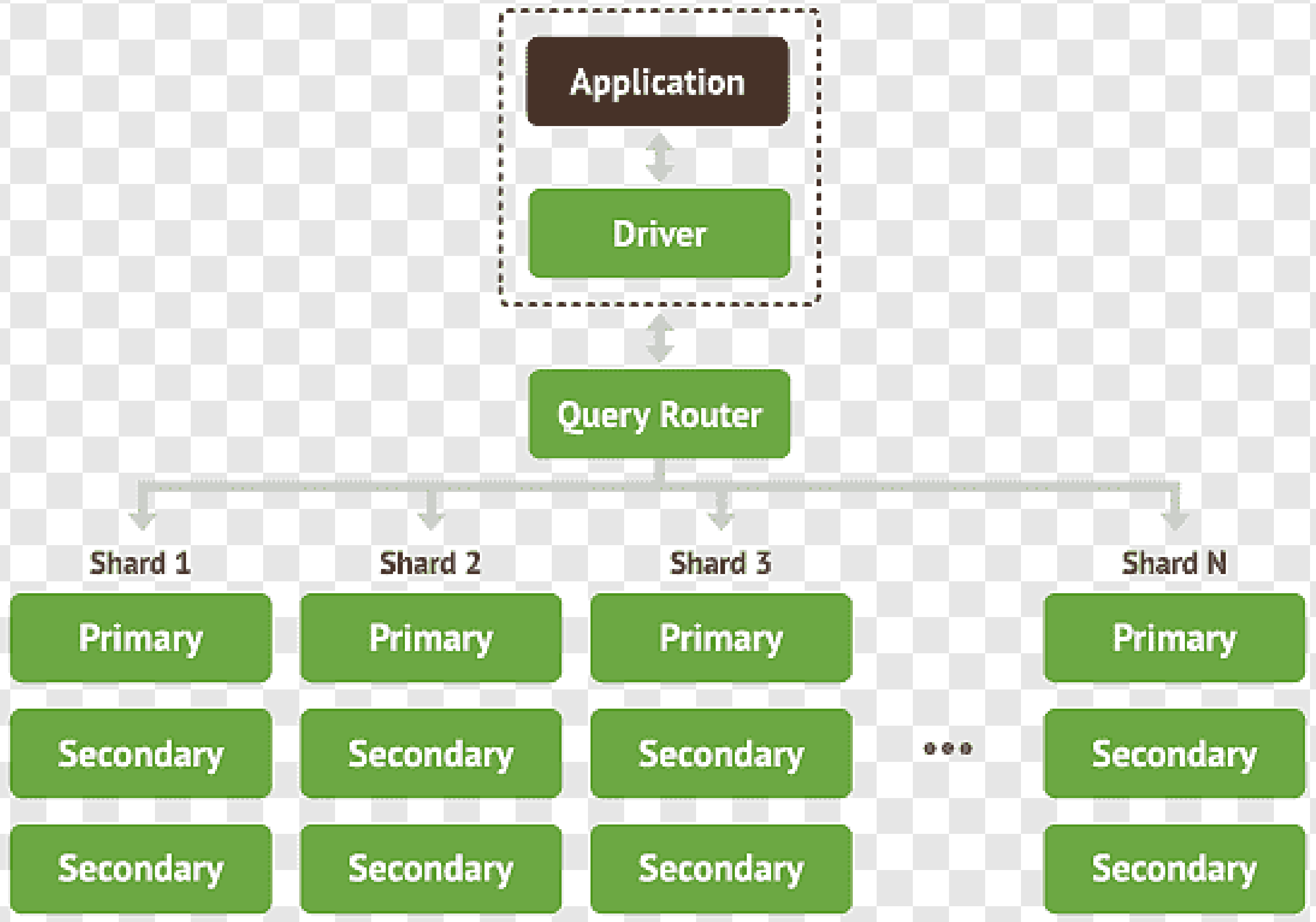


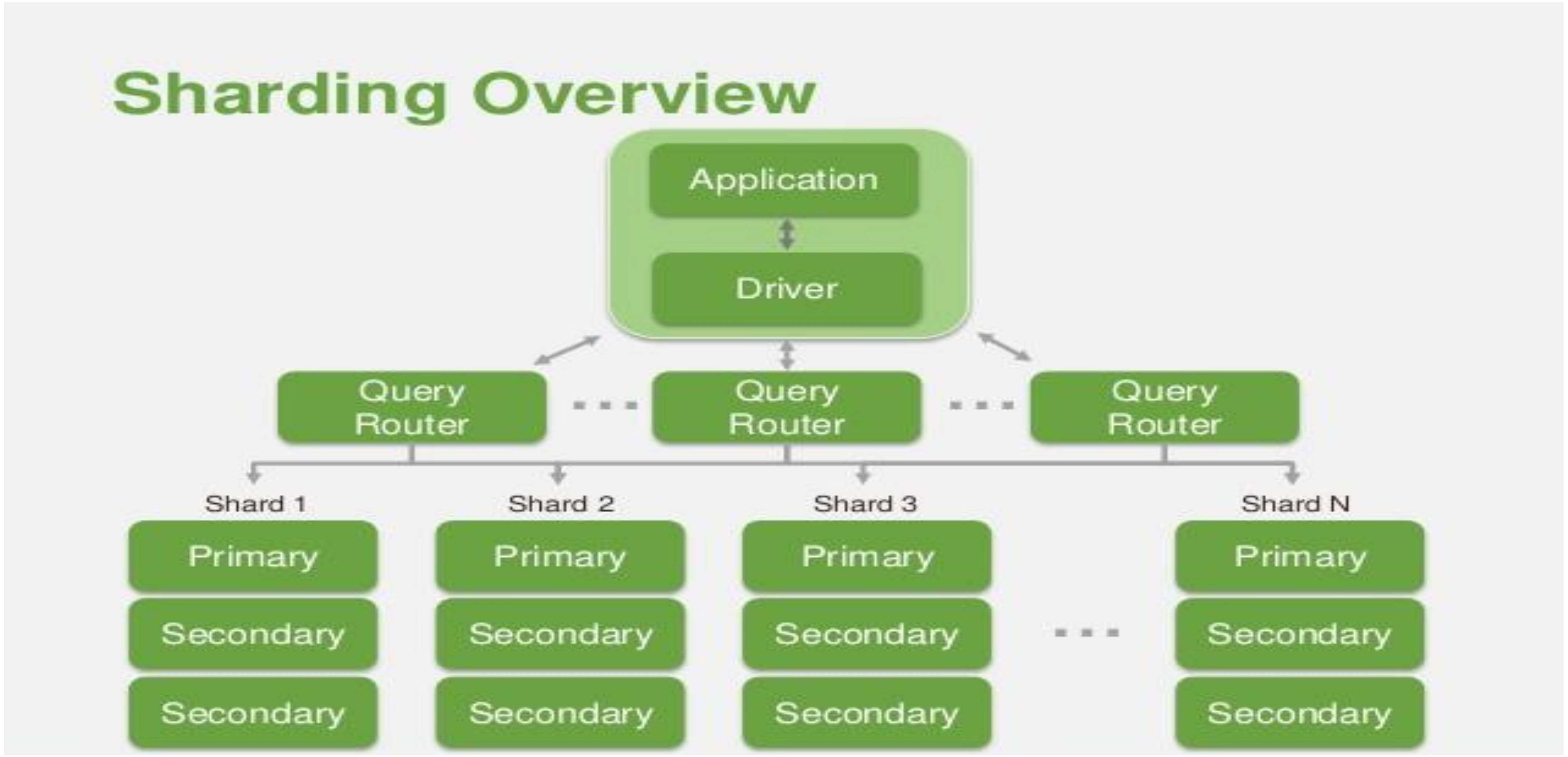
- Actual data
- Needs RAM + Disk IO

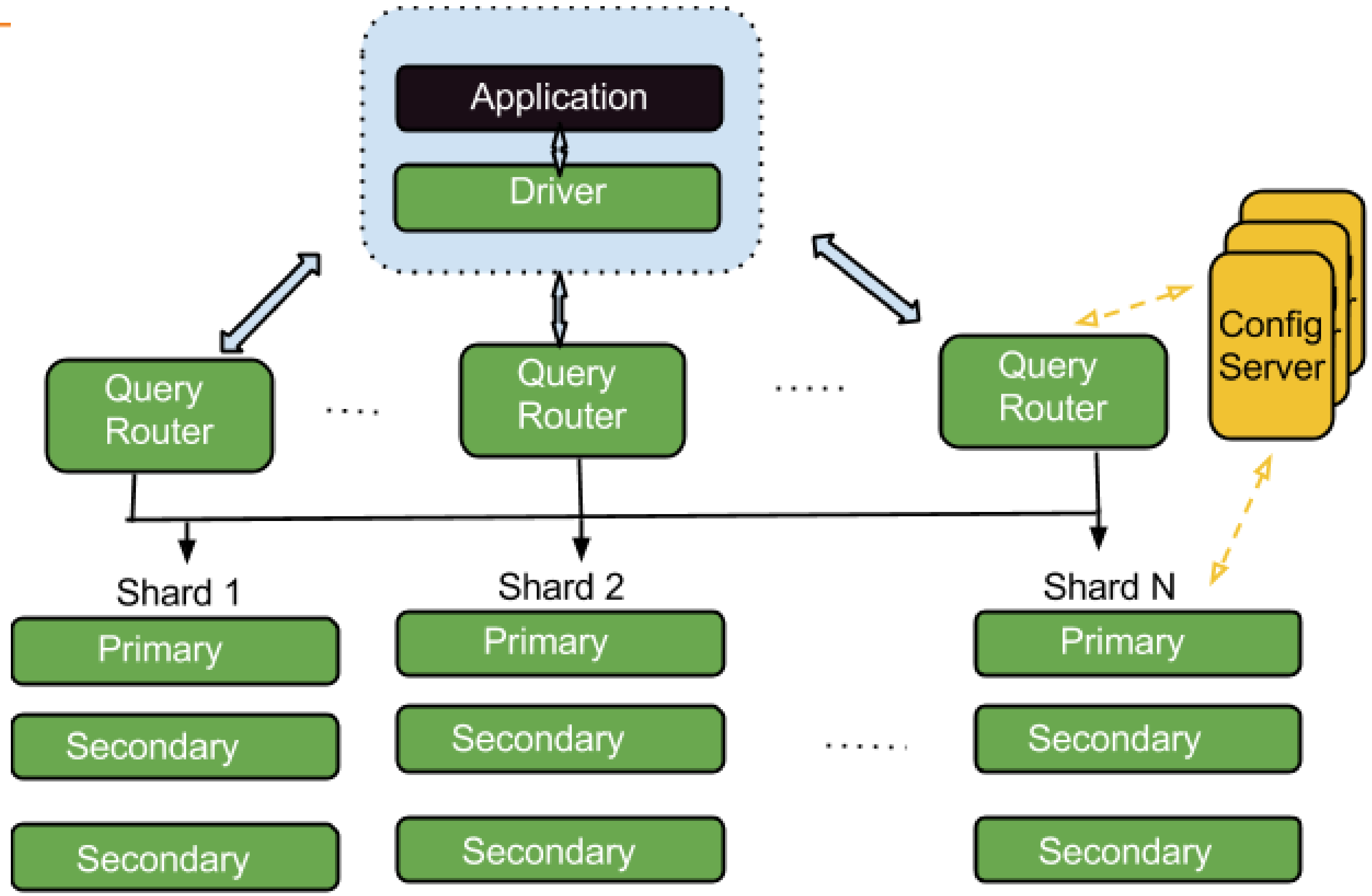
- Can run as Arbiter
- No data
- Just votes to elect primary

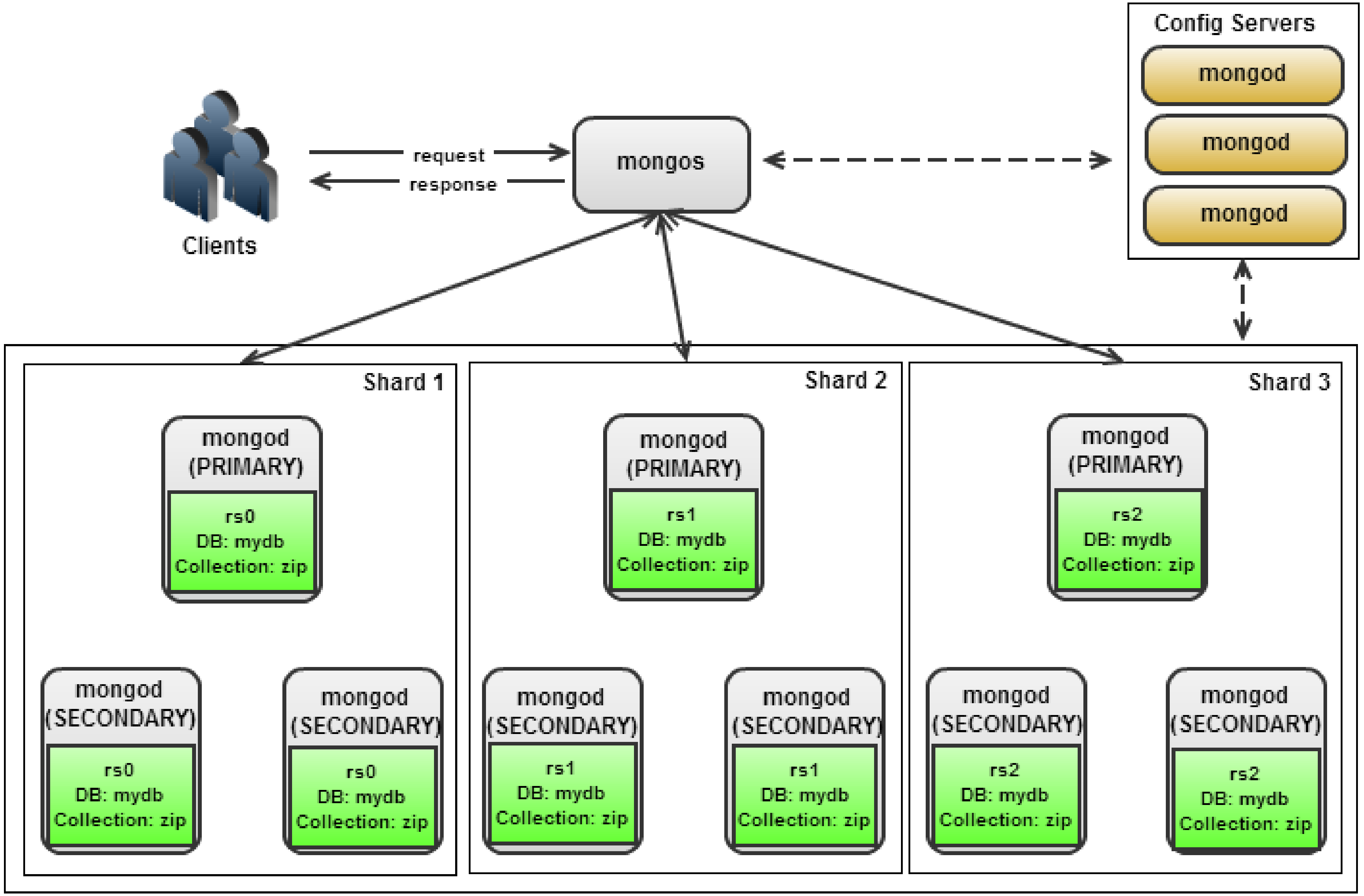
- Stateless router
- Typically run on App Servers

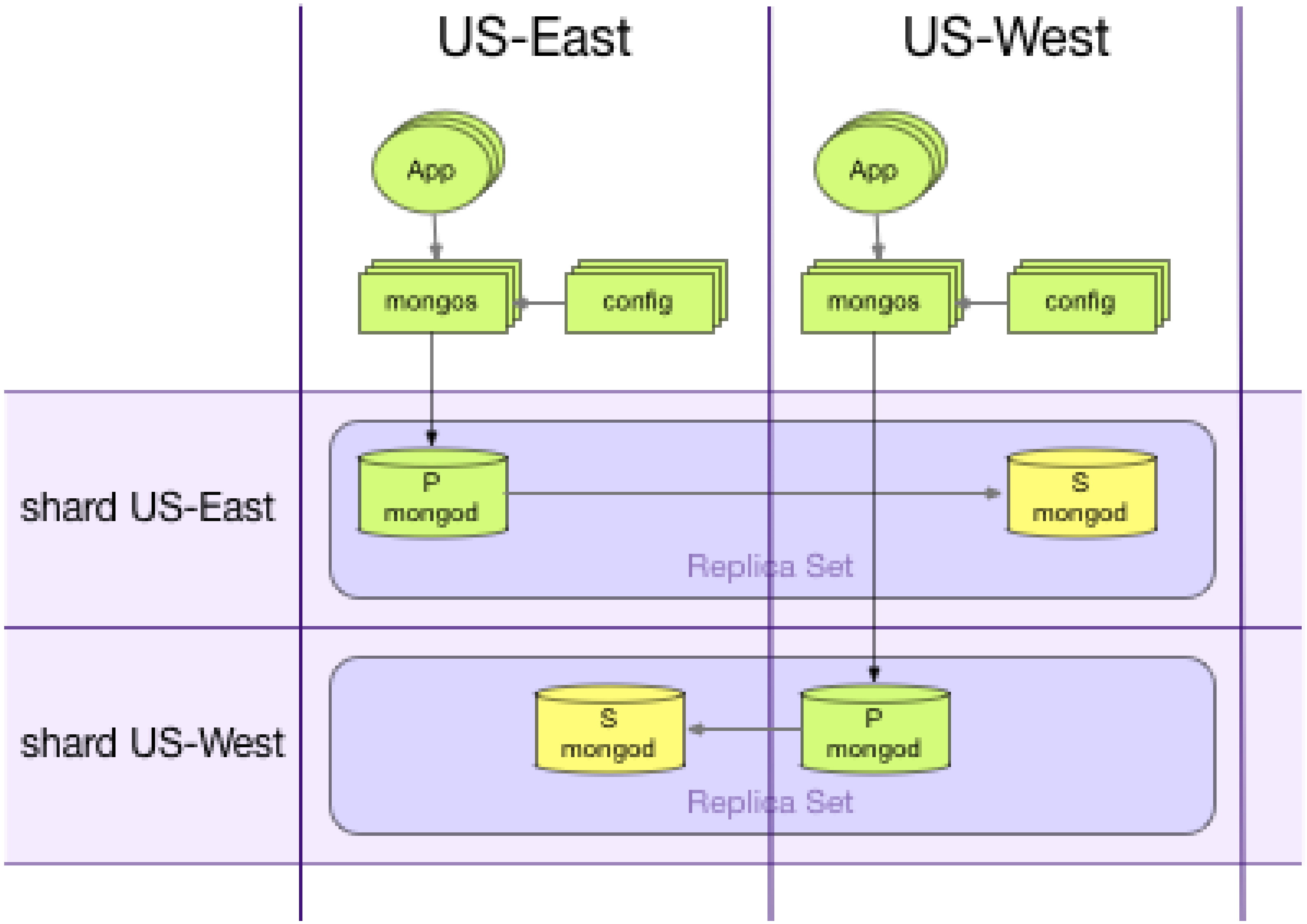
- Stores sharding configuration
- Stores small amounts of data
- Infrequently


Application
Driver
Query Router
Shard 1
Shard 2
Shard 3
Shard N
Primary
Secondary
Secondary
•••


Sharding Overview
Application
Driver
Query Router
Query Router
Query Router
Shard 1
Shard 2
Shard 3
Shard N
Primary
Secondary
Secondary
Primary
Secondary
Secondary
Primary
Secondary
Secondary
Primary
Secondary
Secondary


Application
Driver
Query Router
Query Router
Query Router
Config Server
Shard 1
Shard 2
Shard N
Primary
Secondary
Secondary
Primary
Secondary
Secondary
Primary
Secondary
Secondary


Config Servers
mongod
mongod
mongod
Clients
request
response
mongos
Shard 1
mongod (PRIMARY)
rs0
DB: mydb
Collection: zip
mongod (SECONDARY)
rs0
DB: mydb
Collection: zip
mongod (SECONDARY)
rs0
DB: mydb
Collection: zip
Shard 2
mongod (PRIMARY)
rs1
DB: mydb
Collection: zip
mongod (SECONDARY)
rs1
DB: mydb
Collection: zip
mongod (SECONDARY)
rs1
DB: mydb
Collection: zip
Shard 3
mongod (PRIMARY)
rs2
DB: mydb
Collection: zip
mongod (SECONDARY)
rs2
DB: mydb
Collection: zip
mongod (SECONDARY)
rs2
DB: mydb
Collection: zip


US-East
US-West
App
App
mongos
config
mongos
config
shard US-East
P mongod
S mongod
Replica Set
shard US-West
S mongod
P mongod
Replica Set

Replication

MongoDB uses replica sets to create database replication. A replica set performs the following actions:

- Distributes data across various MongoDB nodes called shards for redundancy.
- Automates failover when server or network outages occur.
- Scales database reads.

A replicaset contains primary node and one or more secondary nodes, the primary node supports both read and write, whereas the secondary supports read.

Replication

MongoDB uses replica sets to create database replication. A replica set performs the following actions:

- Distributes data across various MongoDB nodes called shards for redundancy.
- Automates failover when server or network outages occur.
- Scales database reads.

A replicaset contains primary node and one or more secondary nodes, the primary node supports both read and write, whereas the secondary supports read.

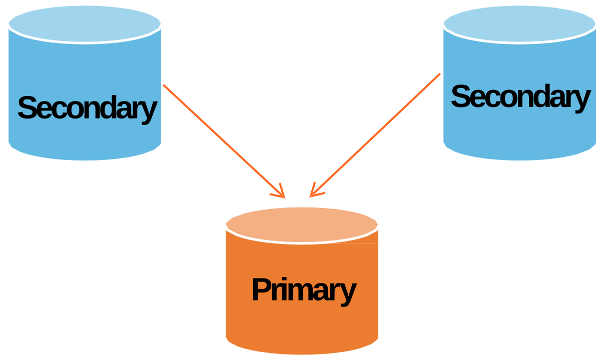
Replication

MongoDB uses replica sets to create database replication. A replica set performs the following actions:

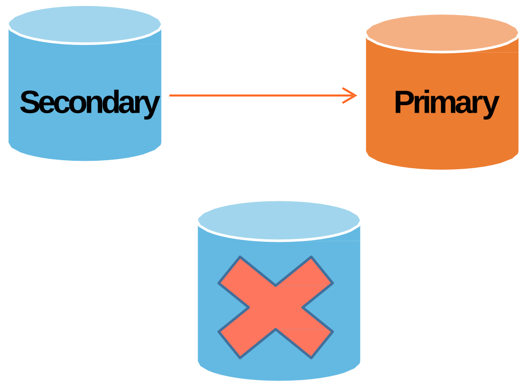
- Distributes data across various MongoDB nodes called shards for redundancy.
- Automates failover when server or network outages occur.
- Scales database reads.

A replicaset contains primary node and one or more secondary nodes, the primary node supports both read and write, whereas the secondary supports read.

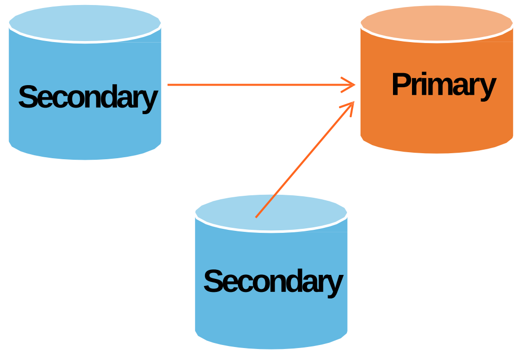
Replication (contd.)



A Working Replica Set



Secondary node is converted to primary when the primary fails



Original primary comes back as secondary

Replica Set

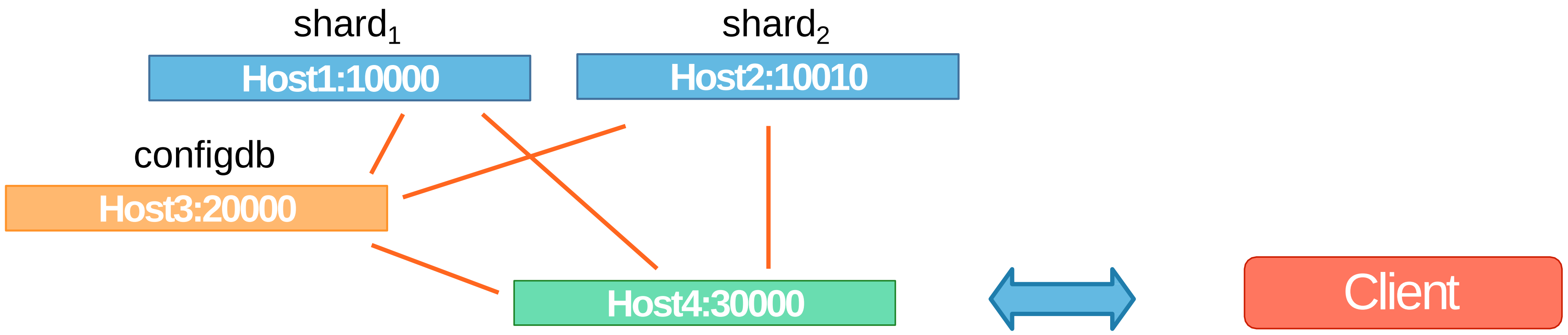
The replica set feature in MongoDB facilitates redundancy and failover with the following actions:

- When a master node of the replica set fails, another member of replica set is converted to the master.
- Allows choosing a master or slave depending on whether you want a strong or delayed consistency.
- Keeps replicated data on the nodes belonging to different data centres to protect the data in the case of natural disaster.

Auto Sharding

MongoDB uses sharding for horizontal scaling. For automatic sharding, choose a shard key that determines the distribution of the collection data.

MongoDB is spread over multiple servers and performs load balancing and/or data duplicating to keep the system up and running in case of hardware failure.

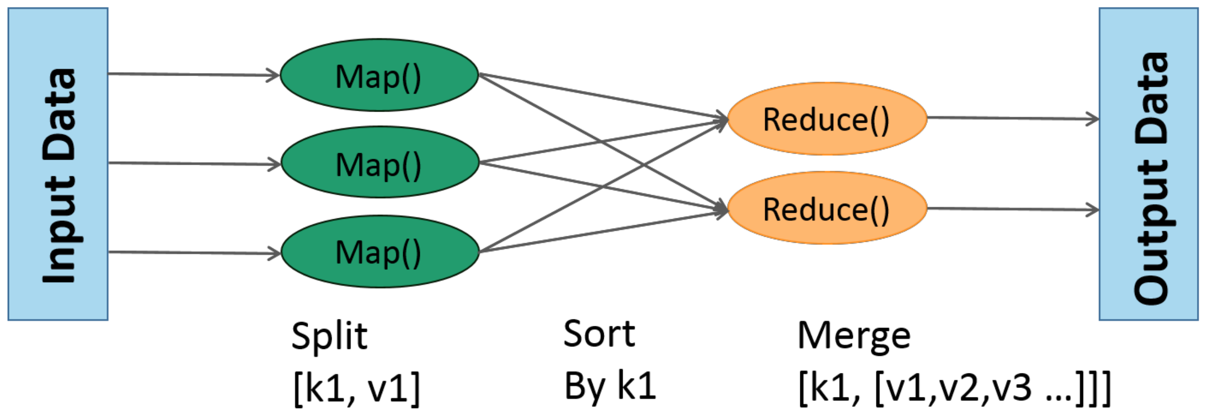


Aggregation and MapReduce

Aggregation operations analyze data sets and return calculated results.

A mapreduce operation consists of two phases:

- **Map**: Documents are processed and one or more objects are produced for each input document.
- **Reduce**: The outputs of the map operation are combined.



! Map reduce can define a query condition to select the input documents, sort and limit the results.

Collection and Database

A collection is a group of documents not restricted by any schema. MongoDB groups collections into databases. A single instance of MongoDB can host several independent databases.

Having different types of documents in the same collection can be cumbersome for developers and administrators. Developers need to ensure that their queries retrieve specific documents or their application codes handle documents of different structure.

! Storing data of a single application in the same database is a good practice.

Schema Design and Modeling

The collections in MongoDB allows flexibility in document structure that enables easy mapping of documents to an entity or an object. All documents in a collection share a similar structure.

The key challenge in data modeling involves maintaining a balance between:

- The needs of the application
- The performance capability of the database engine
- Data retrieval patterns

! When designing a data model consider the application usage of the data, such as queries, updates, and data processing along with its inherent structure.

Reference Data Model

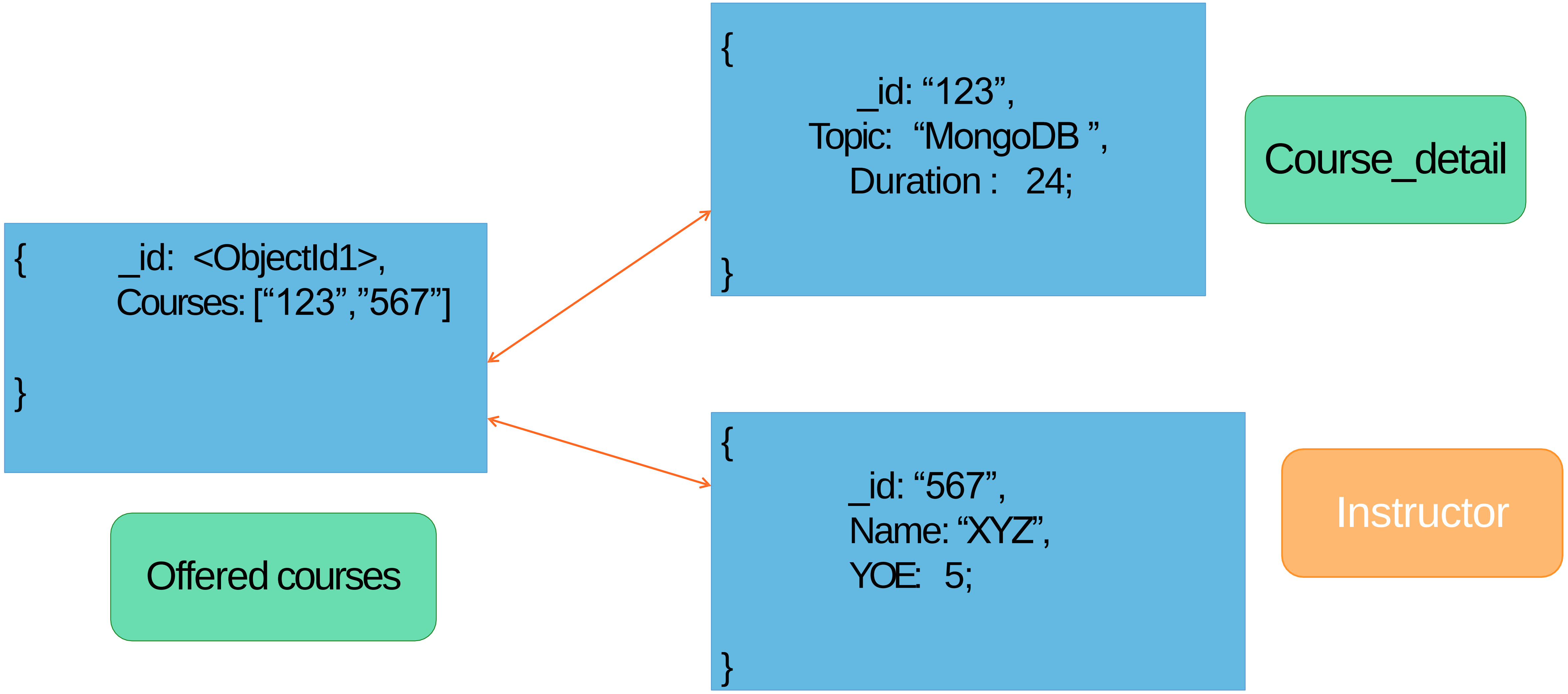
Data model design requires two important things—the structure of documents and representation of the data relationships. References and embedded documents allow applications to represent data relationship.

References use links to store data relationships and applications use these references to access the related data. These are normalized data models, which you can use:

- When embedding document results in data duplication does not give sufficient performance benefit.
- To represent complex many-to-many relationships.
- To model large hierarchical data sets.

! Normalized data models can send more database queries from client to the database server.

Reference Data Model Example



Features of MongoDB

Following are some key features of MongoDB:

Ad hoc queries	Performs search functions by field, range queries, and regular expression searches.
Querying	Uses rich query language and complex conditions to retrieve documents.
Fast In-Place Updates	Allows write semantics, enable journaling, and control the speed and durability.
Server-side JavaScript execution	Uses JavaScript in queries and aggregation functions such as MapReduce, which are sent to the database for execution.
Capped Collections	Maintains insertion order and once the specified size has been reached, behaves like a circular queue.

Embedded Data Model Example

```
{
        _id: “123”,
          coursedetail:
        {
                Topic:   “MongoDB ”,
                Duration : 24,

        }

        instructor:
          {
                name: “XYZ”,
                YOE: 5,
        }

}
```

Embedded Data Model

Embedded documents store related data in a single document structure and thus capture relationships between data. MongoDB allows denormalized data models to permit retrieval and manipulation of related data in a single database operation.

Embedded data models can be used when the following relationships exist between entities:

- ‘contains’
- one-to-many

! Embedded data models allow related data update in a single atomic write operation.

Data Types

MongoDB supports the following data types:

- **Null:** Used to represent both a null value and a nonexistent field

  {"x”: null}

- **Undefined:** Used in documents

  {"x" : undefined}

- **Boolean:** Used for the values 'true' and 'false'

  {"x" : true}

- **32-bit integer:** Cannot be represented on the shell
- **64-bit integer:** The shell cannot represent these
- **64-bit floating point number:** All numbers in the shell will be of this type

Data Types (contd.)

Some more data types supported by MongoDB are:

- **Maximum value:** Contains a special data type that represents the largest possible value. The shell does not support this type.
- **Minimum value:** Contains a special data type that represents the smallest possible value. The shell does not support this type.
- **ObjectId:** Unique, fast to generate and ordered, these consists of 12 bytes where the first four bytes represent the ObjectId creation time.
- **String**: Are UTF-8 compliant. When serializing and deserializing BSON, programming languages convert language strings to UTF-8 format.
- **Symbol**: Not supported by the shell. When the shell gets a symbol, it converts it into a string.
- **Timestamps:** BSON offers a special timestamp for internal MongoDB use that is not associated with the regular Date type.
- **Date:** A 64-bit integer that denotes the number of milliseconds since the UNIX epoch.
- **Regular expression:** Documents contain JavaScript's regular expression syntax.

```
{"x" : //i}
```

Data Type (contd.)

Some more data types are:

- **Code:** Documents can contain JavaScript code. For example:

  {"x" : function() { /* ... */ }}

- **Binary data:** A string of arbitrary bytes that cannot be manipulated from the shell.
- **Array:** Sets or lists of values can be represented as arrays. For example:

  {"courses" : ["PMP", "Cloud", " MongoDB "]}

- **Embedded document:** Documents can contain entire documents, embedded as values in a parent document. For example:

  {"course_duration" : {"MongoDB " : "24 Hrs"}}

Core Servers of MongoDB

The core database server of MongoDB can be run on executable process called mongod or mongodb.exe on Windows.

A mongod process can be run on several modes:

- Replica set: Configurations comprise two replicas and an arbiter process that reside on a third server.
- Per-shard replica sets: The auto-sharding architecture of MongoDB consist of mongod processes configured as per-shard replica sets.
- Mongos: A separate routing server is used to send requests to the appropriate shard.

! Mongos queries from the application layer and locates the data in the sharded cluster to complete its operations.

MongoDB's Tools

MongoDB tools consists of the following:

- **The JavaScript shell:** The MongoDB command shell is JavaScript-based used for administering the database and manipulating data.
- **Database drivers :** Provides an Application Program Interface (API) that matches the syntax of the language used.
- **Command-line tools:**
  - mongodump and mongorestore: Standard utilities that helps backup and restore a database.
  - mongoexport and mongoimport: Used to export and import JSON, comma separated value (CSV), and Tab separated Value (TSV) data.
  - Mongosniff: A wire-sniffing tool used for viewing operations sent to the database.
  - Mongostat: Provides helpful statistics, such as inserts, queries, updates, deletes, and so on.

Use Cases

Some use cases of MongoDB are:

Personalization

- Allows personalization of customer experience in real time to predict the wants and needs of customers.

Mobile

- Allows scaling of mobile applications to cater to millions of online users.

Internet of Things (IOT)

- Manages huge volumes of data generated by IOT. Allows building of your own IOT suite to manage big data on your own.

Real-time Analytics

- Allows real-time data analysis, minute by minute and second by second.

Use Cases (contd.)

Some more use cases of MongoDB are:

Web Application

- Helps Web application manage data efficiently through rich data structures, such as documents.

Content Management

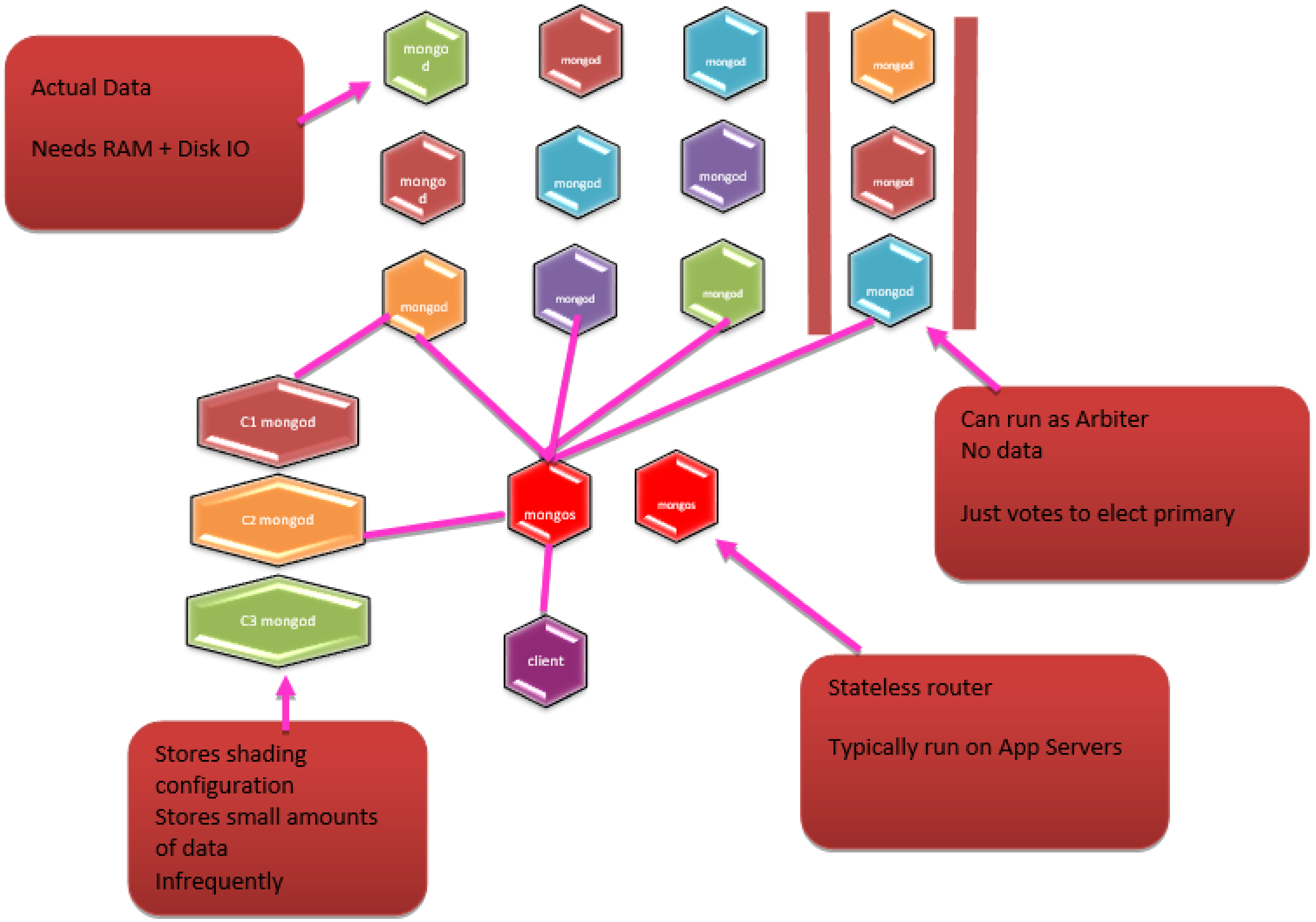
- Allows storing and presenting of any type of content, build new features, incorporate all kinds of data in a single database.

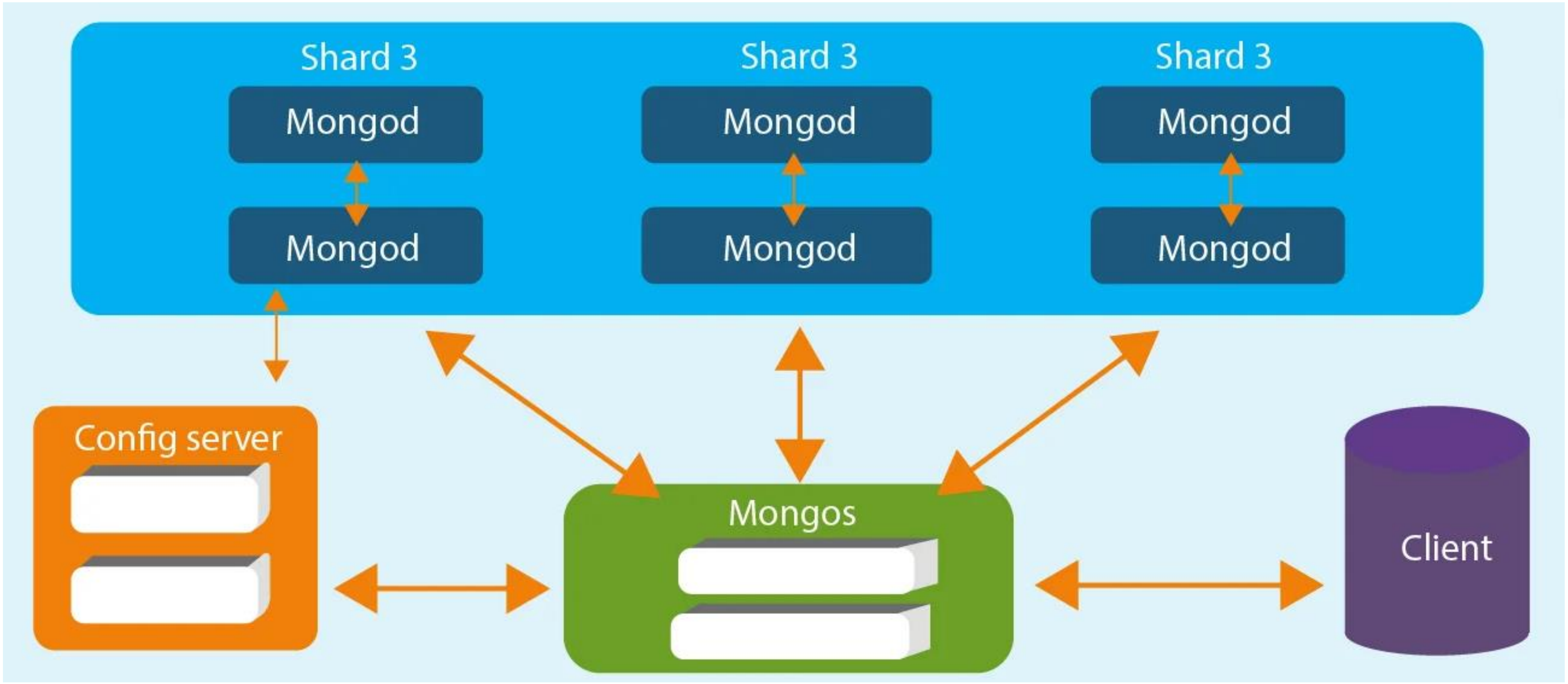
Catalog

- Allows access to customer data to provide online shoppers a highly personalized and smooth shopping experience.

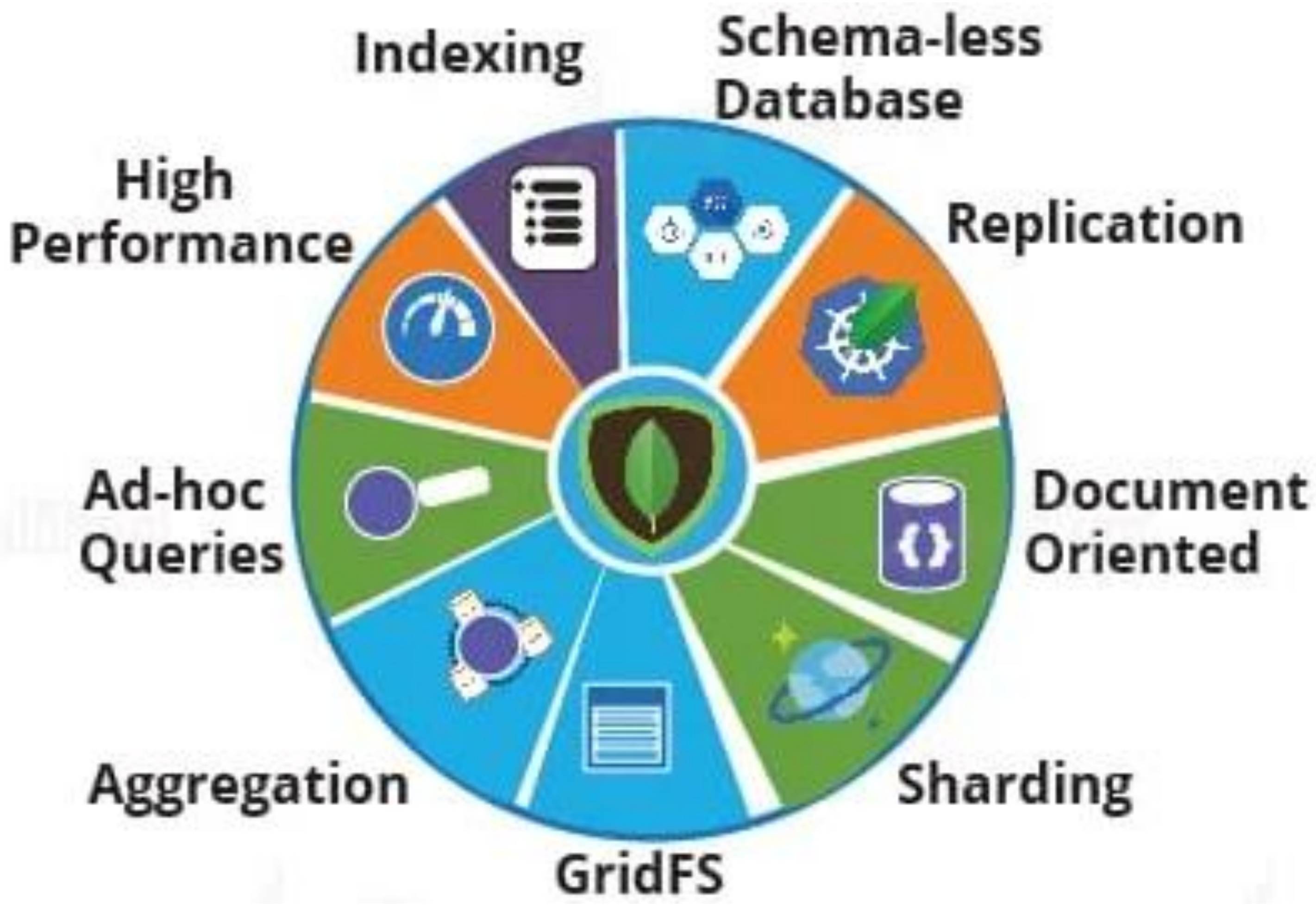
Single View

- Allows building of a single view of all your business data.


Actual Data
Needs RAM + Disk IO
mongo d
mongod
mongod
mongod
mongo d
mongod
mongod
mongod
mongod
mongod
mongod
mongod
C1 mongod
C2 mongod
C3 mongod
mongos
mongos
client
Can run as Arbiter
No data
Just votes to elect primary
Stateless router
Typically run on App Servers
Stores shading configuration
Stores small amounts of data
Infrequently


Shard 3
Mongod
Mongod
Shard 3
Mongod
Mongod
Shard 3
Mongod
Mongod
Config server
Mongos
Client

Important Features of MongoDB



Knowledge Check

Case Study

Scenario Analysis Solution

ABC HR Services Private Limited manages human resources of various organizations. Therefore, the employee data is huge and hierarchical. The data is growing fast and also there is a frequent change in the schema of the data. The organization uses RDMBS which stores the complex hierarchical information data in standard format that prevents sharing data to its clients. Moreover, as the schema of data changes a lot, it has created a huge issue with their existing RDBMS database. Hence, the organization has established a team to set up a new database.

Click ***Analysis*** *to know the team's next move.*

Case Study

Scenario Analysis Solution

The team explored various technologies and picked MongoDB as a new database for its following features:

1. It supports variable Schema.
2. It is flexible to store complex data that has hierarchical structure.
3. It stores the data in JSON format, which is standard serialization format. So, data can be easily shared without any extra work.
4. It can handle huge data if we use its cluster mode and it can also scale linearly.

Click ***Solution*** *to know the steps to explore MongoDB database.*

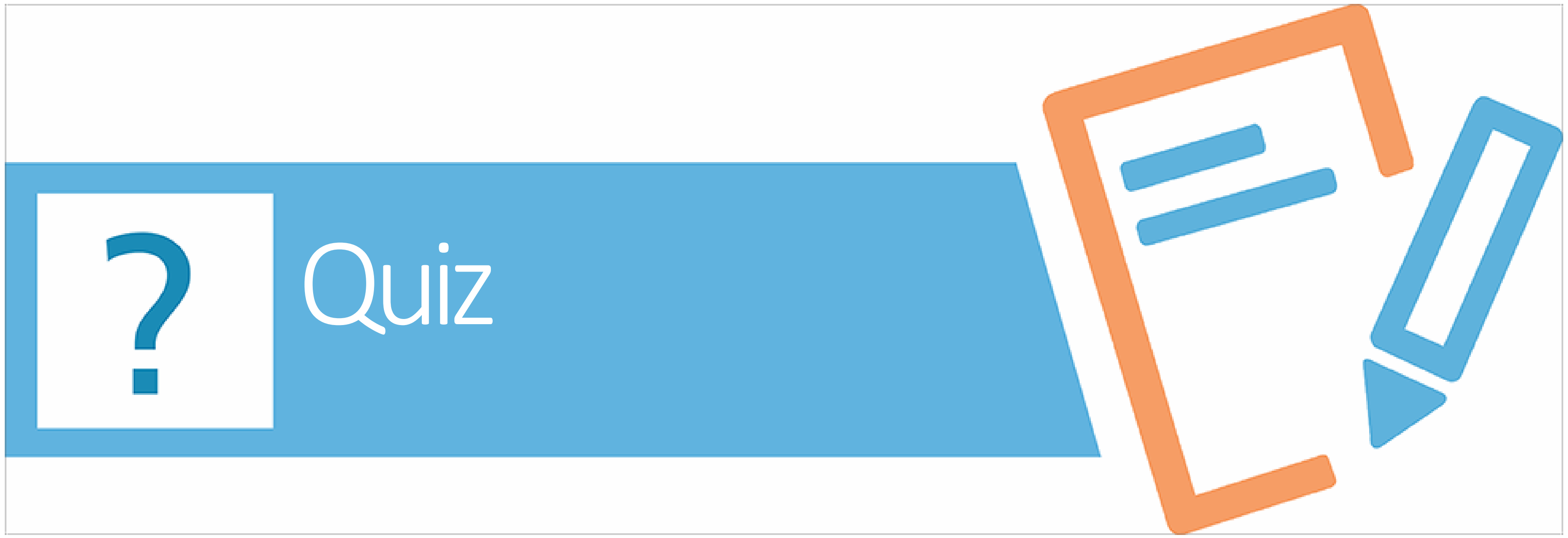
Case Study

Scenario Analysis Solution

The team takes the following steps to explore MongoDB database:

1. Consolidate a sample JSON data
2. Insert data to MongoDB using mongo import tool provided by MongoDB
3. Check data on MongoDB server
4. Export the same data to a file to test its export ability
5. Test the same data using Dump so that metadata is captured as well
6. Restore the data using the dump file into a new collection
7. Recheck the data in MongoDB new collection

*Click **Demo** to know the query execution process.*


?
Quiz

QUIZ 1	In which of the following format does MongoDB store data?

a. JSON

b. BSON

c. XML

d. HTML

QUIZ 1

In which of the following format does MongoDB store data?

a. JSON

b. BSON

c. XML

d. HTML

The correct answer is b.

Explanation: MongoDB stores the data into BSONformat.

QUIZ 2

Which among the following is the core server of MongoDB ?

a. mongos

b. Config server

c. Routing server

d. mongod

QUIZ 2

Which among the following is the core server of MongoDB ?

a. mongos

b. Config server

c. Routing server

d. mongod

The correct answer is **d**.

Explanation: mongod is the core server of the MongoDB.

QUIZ

3

Which among following is analog of an RDBMS row in MongoDB?

a. Database

b. Schema

c. Collection

d. Index

QUIZ 3

Which among following is analog of an RDBMS row in MongoDB?

a. Database

b. Schema

c. Collection

d. Index

The correct answer is **c**.

Explanation: In MongoDB, collection is analog of an RDBMS row.

Summary

Here is a quick recap of what was covered in this lesson:

- MongoDB is a document-based database that represents a complex hierarchical data relationship using embedded document model or using reference model.
- MongoDB uses JSON as the data format, which is based on:
  - A collection of name/value pairs
  - An ordered list of values
- A collection in MongoDB is a table and view structure that groups structurally or conceptually similar documents.
- MongoDB supports auto-sharding to manage data distribution across multiple nodes.
- MongoDB uses replica sets to create redundancy and automates failover when server or network outages occur.