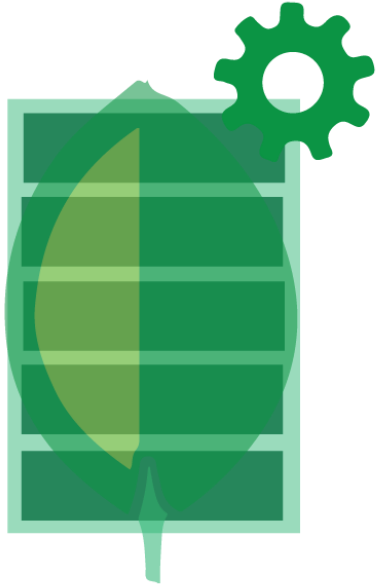


edureka!

MODULE-2
CRUD OPERATIONS

Course Topics

edureka!

- → Module 1
  - » Introduction and Overview
  - » No SQL
- → Module 2
  - » **CRUD Operations**
  - » **CRUD Concerns**
- → Module 3
  - » Schema Design and Data Modeling
  - » Comparison with Relational Systems
- → Module 4
  - » Administration
  - » Backup and Recovery
- → Module 5
  - » Scalability and Availability
  - » Replication and Sharding
- → Module 6
  - » Indexing and Aggregation Frame work
  - » Performance Tuning
- → Module 7
  - » Application Engineering and MongoDB® Tools
  - » Interface with Other Language
- → Module 8
  - » Project, Additional Concepts and Cases Studies

Objectives

At the end of this module, you will be able to

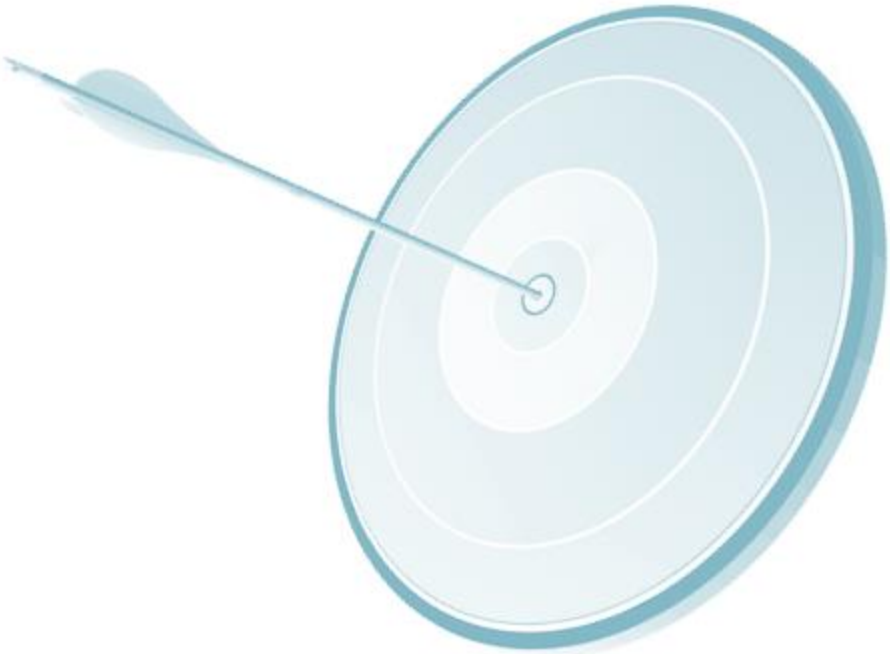
→ Understand MongoDB®'s development and production architecture

→ Understand read and write concepts of MongoDB®

→ Understand how Journaling works

→ Use mongo shell for CRUD operations

→ Understand different MongoDB® data types



Annie's Question

edureka!

What are typical usage for MongoDB® ?

Annie's Answer

edureka!

MongoDB® has a general-purpose design, making it appropriate for a large number of use cases. Examples include content management systems, mobile applications, gaming, e-commerce, analytics, archiving, and logging.

Annie's Question

edureka!

Where shouldn't MongoDB® used?

Annie's Answer

edureka!

MongoDB® shouldn't be used for systems that require SQL, joins, and multi-object transactions.

Annie's Question

edureka!

Does MongoDB® handle caching?

Annie's Answer

edureka!

Yes. MongoDB® keeps all of the most recently used data in RAM. If you have created indexes for your queries and your working data set fits in RAM, MongoDB® serves all queries from memory.

Annie's Question

What is the difference between mongo and mongos?

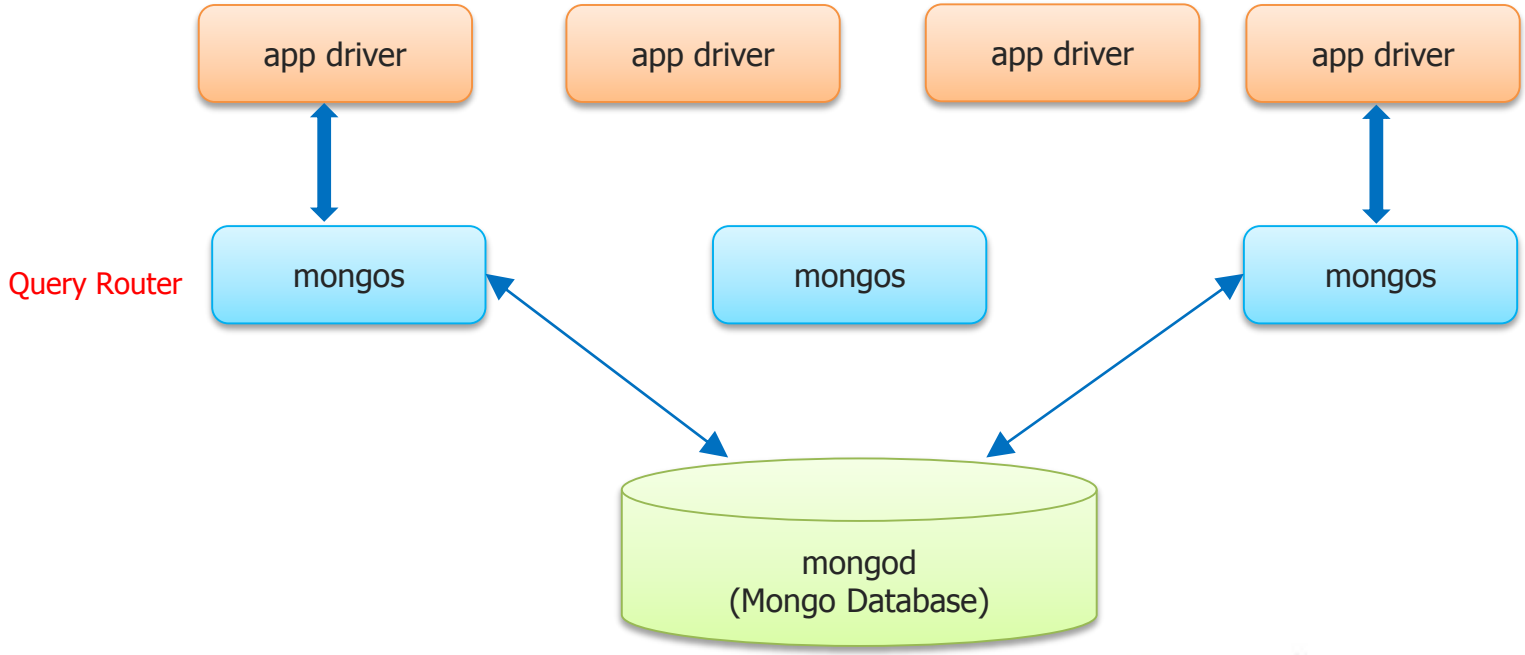
Annie's Answer

edureka!

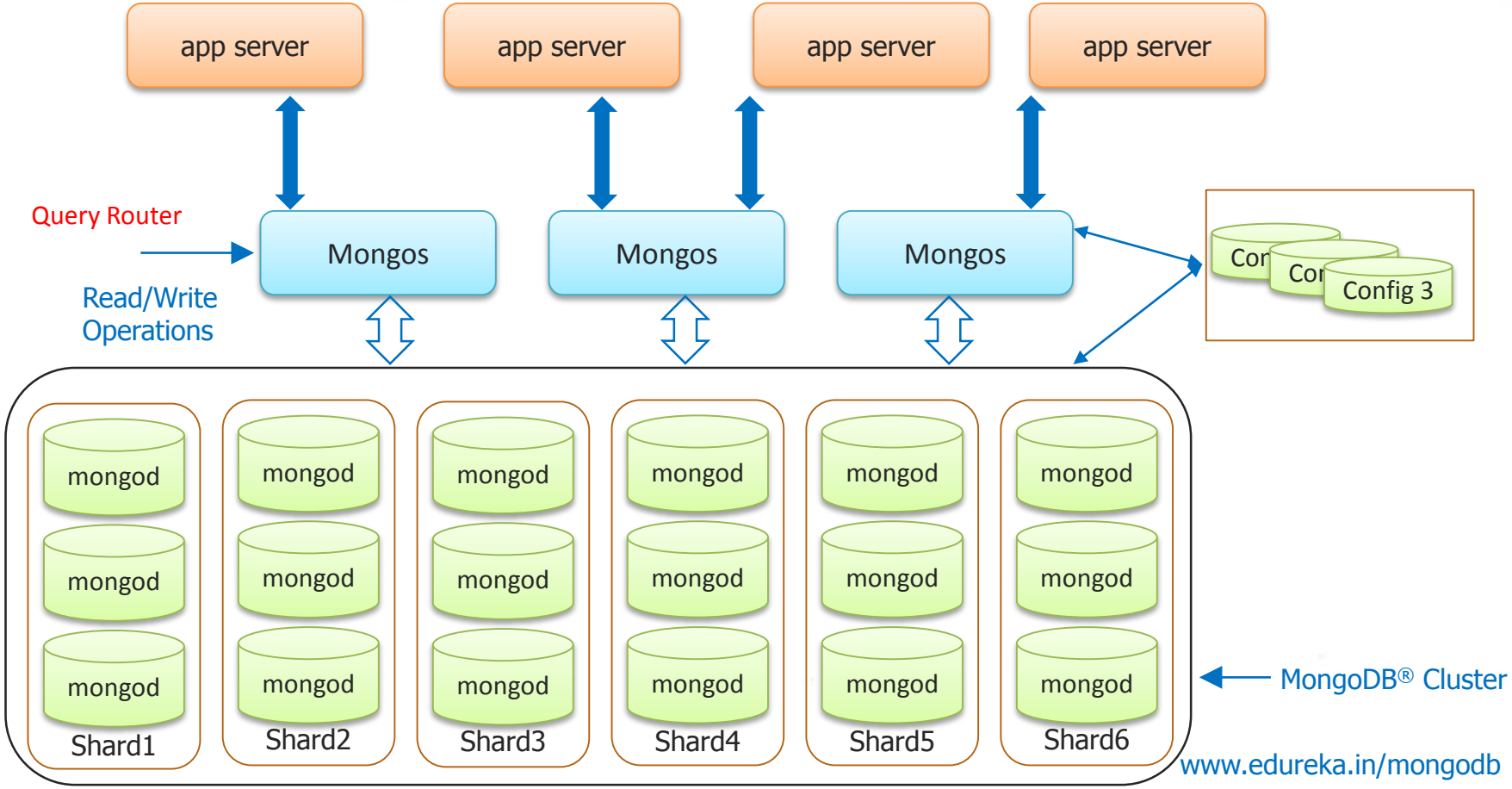
Mongo - This javascript shell acts as a client and connects to a mongod or mongos instance for general database operations.

Mongos - This is a daemon which acts as the query router between the client (mongo or application layer) and the Sharded Cluster (one or multiple mongod).

MongoDB® Development Environment



MongoDB® Production Overview



MongoDB® CRUD Introduction

edureka!

→ MongoDB® provides rich functionalities for reading and manipulating data.

→ CRUD stands for Create, Read, Update and Delete.

C → Create

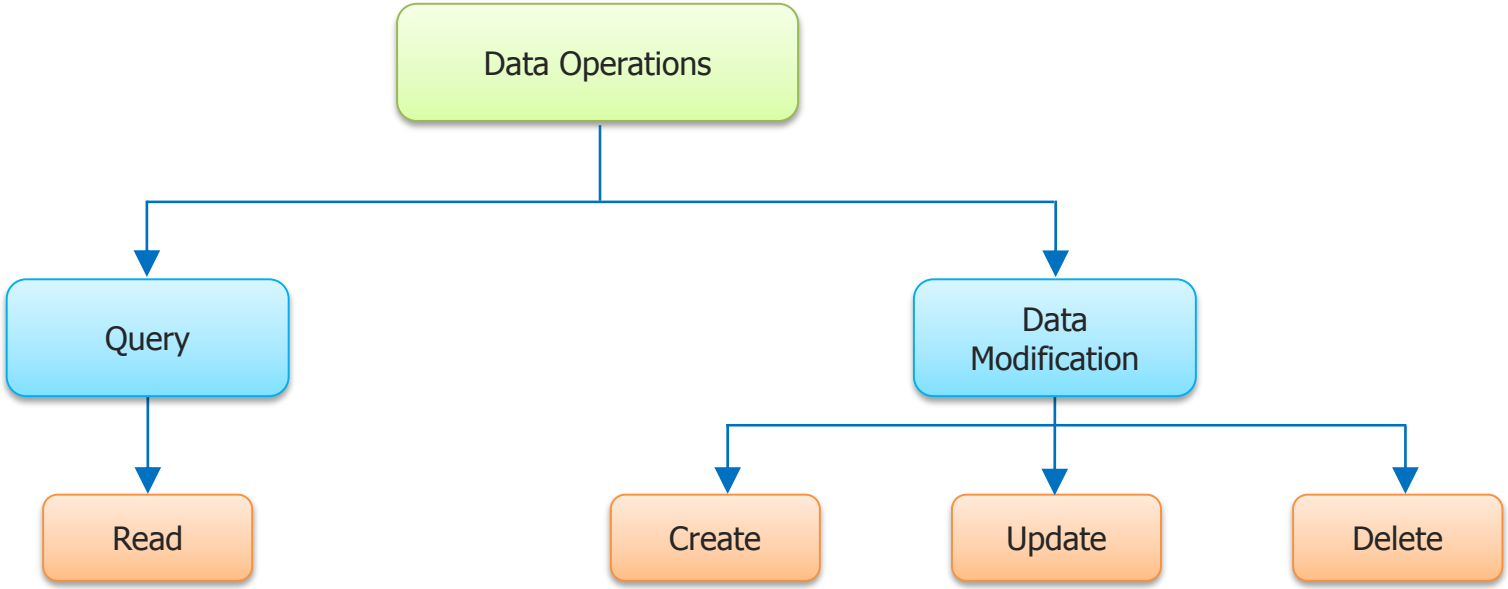
R → Read

U → Update

D → Delete

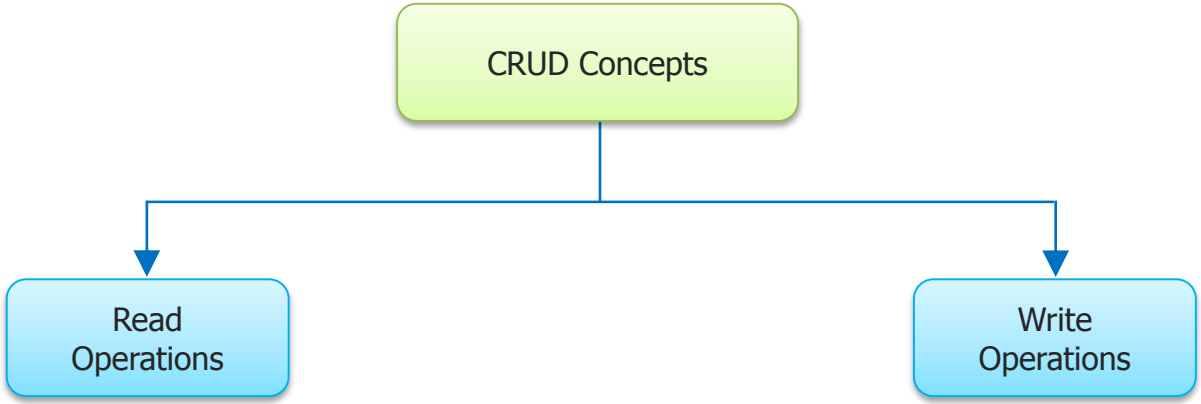
These terms are the basic stuffs for all interactions with the databases.

Database Operations

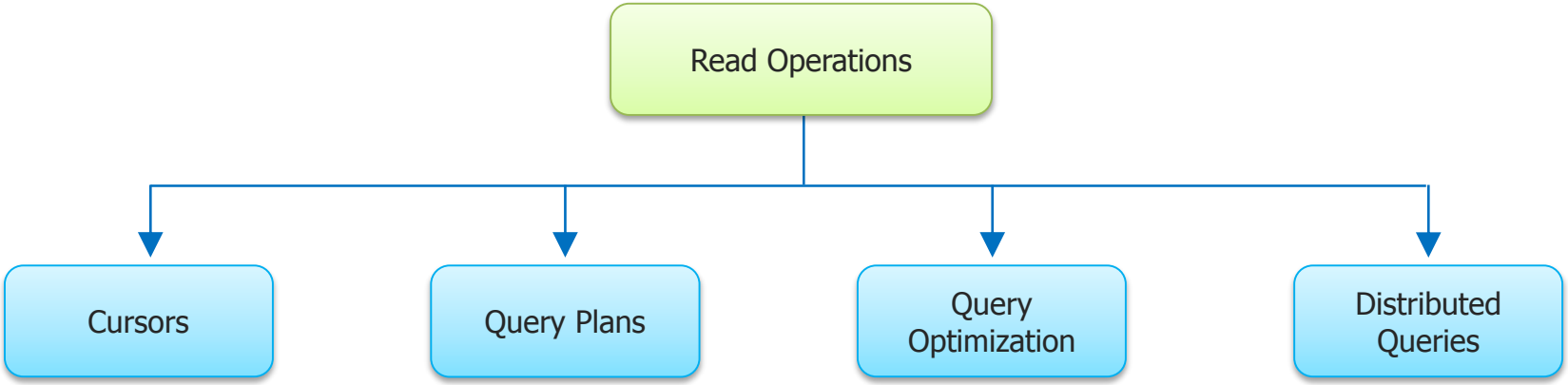


MongoDB® CRUD Concepts

edureka!



Read Operations



Query Interface

edureka!

→ A MongoDB® Query :

```
db . users . find(                          ← collection
    { age : { $gt : 18 } },                 ← query criteria
    { name : 1, address : 1 }               ← projection
) . limit (5)                               ← cursor modifier
```

→ An Equivalent SQL query:

```
SELECT   _id , name , address   ← projection
FROM     users                  ← table
WHERE    age > 18               ← select criteria
LIMIT    5                      ← cursor modifier
```

Query Behavior

edureka!

- All queries in MongoDB® address a single collection.
- You can modify the query to impose limits, skips, and sort orders.
- The order of documents returned by a query is not defined unless you specify a sort().
- Operations that modify existing documents (i.e. updates) use the same query syntax as queries to select documents to update.
- In aggregation pipeline, the $match pipeline stage provides access to MongoDB® queries.
- MongoDB® provides a db.collection.findOne() method as a special case of find() that returns a single document.

Projections

edureka!

→ Exclude One Field From a Result Set

```
db.records.find( { "user_id": { $lt: 42} }, { history: 0} )
```

→ Return Two Fields and the _id Field

```
db.records.find( { "user_id": { $lt: 42} }, { "name": 1, "email": 1} )
```

→ Exclude _id field

```
db.records.find( { "user_id": { $lt: 42} }, {“_id": 0 , "name": 1, "email": 1} )
```

CURSORS

edureka!

→ Queries return iterable objects, called cursors, that hold the full result set of the query request.

→ If the returned cursor is not assigned to a variable using the var keyword, then the cursor is automatically iterated up to 20 times.

→ db.collection.find() method queries a collection and returns a cursor to the returning documents. To access the documents, you need to iterate the cursor.

→ The db.serverStatus() will return cursor information. How many cursors are open, timed out etc. can be identified.

Iterate through the Cursor

edureka!

```
var myCursor = db.inventory.find( { type: 'food' } );

myCursor
```

```
var myCursor = db.inventory.find( { type: 'food' } );

myCursor.forEach(printjson);
```

```
var myCursor = db.inventory.find( { type: 'food' } );

var documentArray = myCursor.toArray();

var myDocument = documentArray[3];
```

Query Optimization

edureka!

→ Create an Index to Support Read Operations.

→ Query Selectivity – the inequality operators $nin and $ne are not very selective

→ Covering a Query – all the fields in the query and returned are part of the index

→ Query Plans – query optimizer caches query plan

Annie's Question

When Query Plan Revision happens?

Annie's Answer

As collections change over time, the query optimizer deletes the query plan and re-evaluates after any of the following events:

- The collection receives 1,000 write operations.
- The reIndex rebuilds the index.
- You add or drop an index.
- The mongod process restarts.

Annie's Question

How do I isolate cursors from intervening write operations?

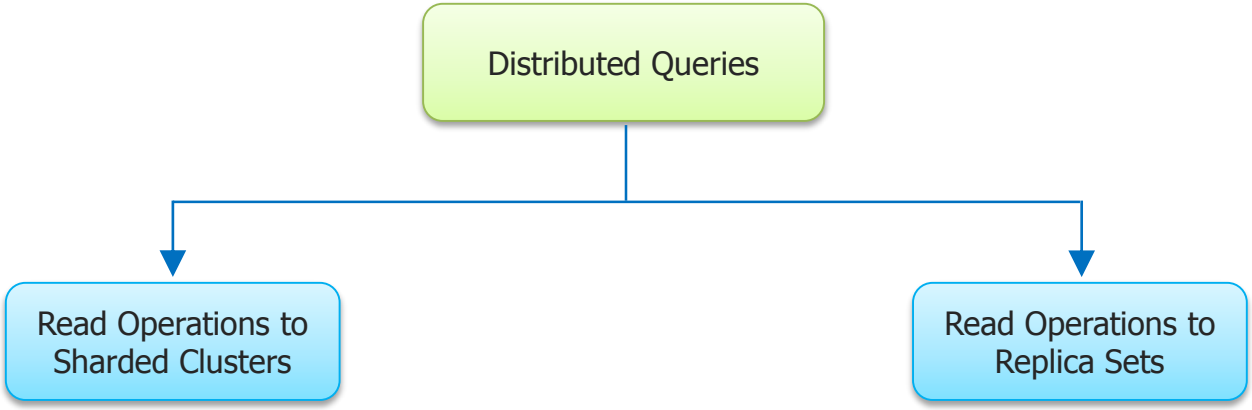
Annie's Answer

edureka!

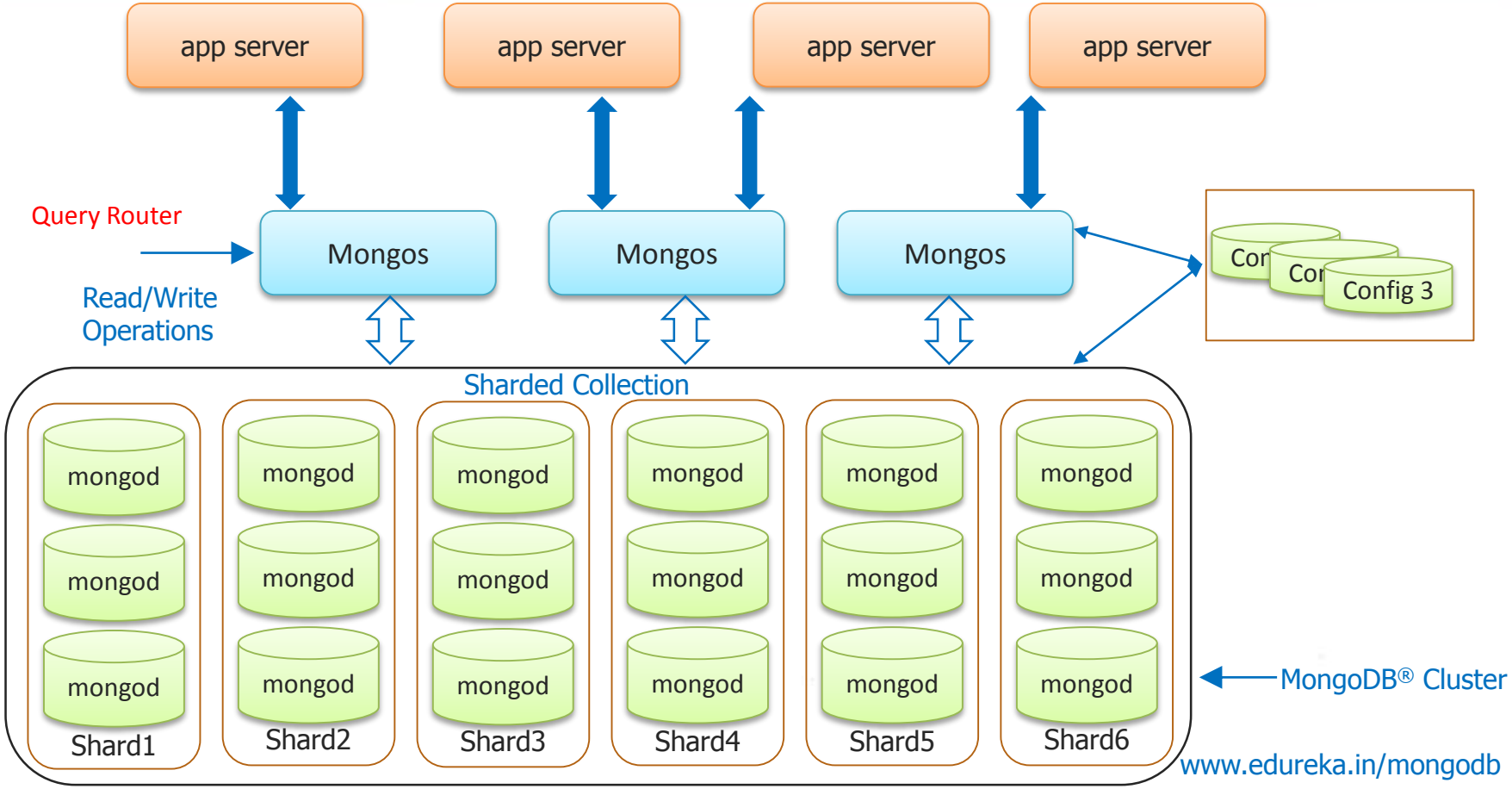
MongoDB® cursors can return the same document more than once in some situations.

You can use the snapshot() method on a cursor to isolate the operation for a very specific case.

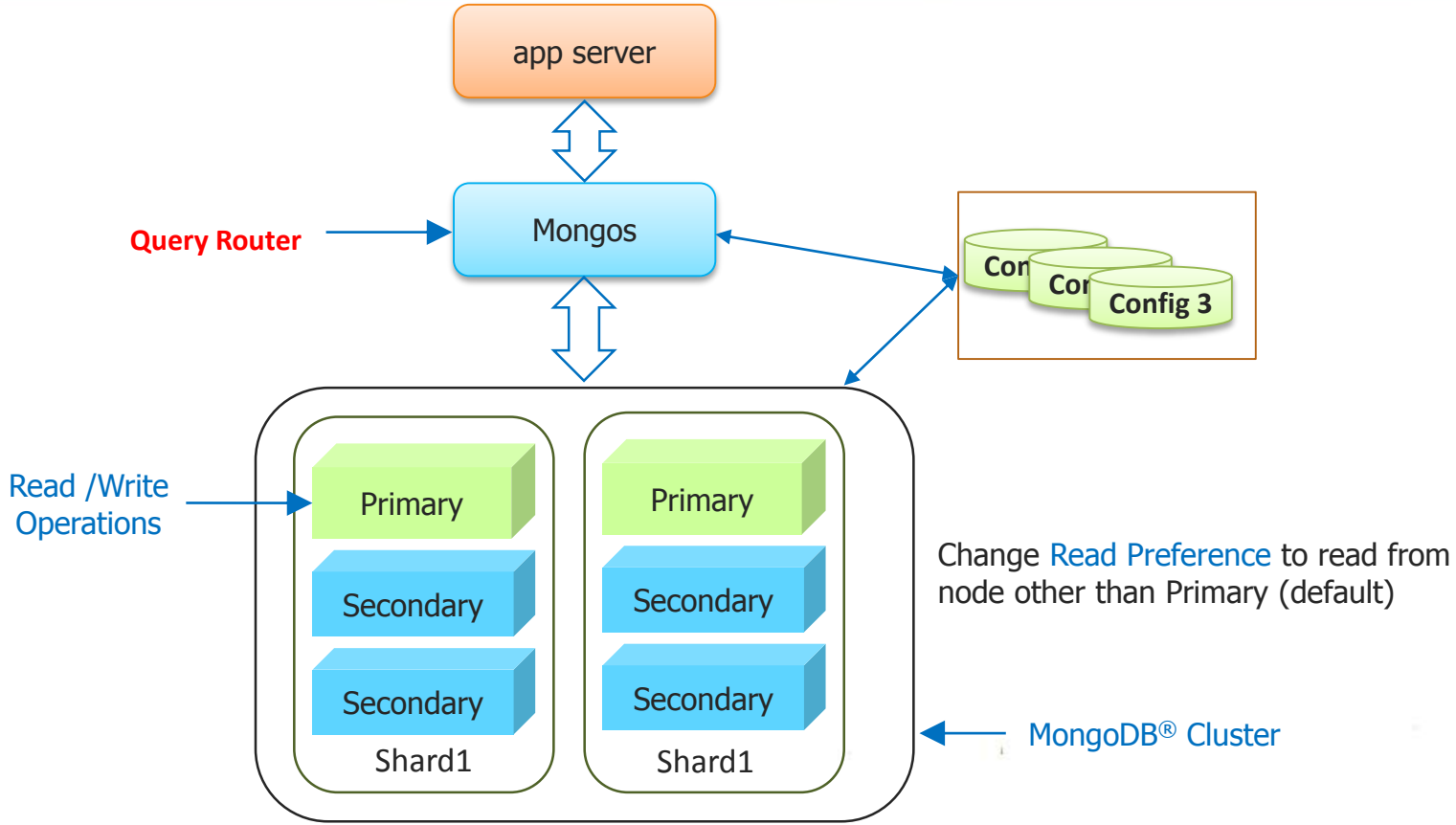
Distributed Queries



Read Operations to Sharded Clusters



Read Operations to Replica Set



Write Operations Overview

edureka!

Create in MongoDB® :

```
db . users . insert(          ← collection
    {
        name : "sure",        ← field value
        age: 26,              ← field value   } Documents
        status: "A"           ← field value
    }
)
```

Update in MongoDB® :

```
db . users . update(                          ← collection
    { age : { $gt : 18 } } ,                  ← update criteria
    { $set : { status: "A" } } ,              ← update action
    { multi : true} ,                         ← update option
)
```

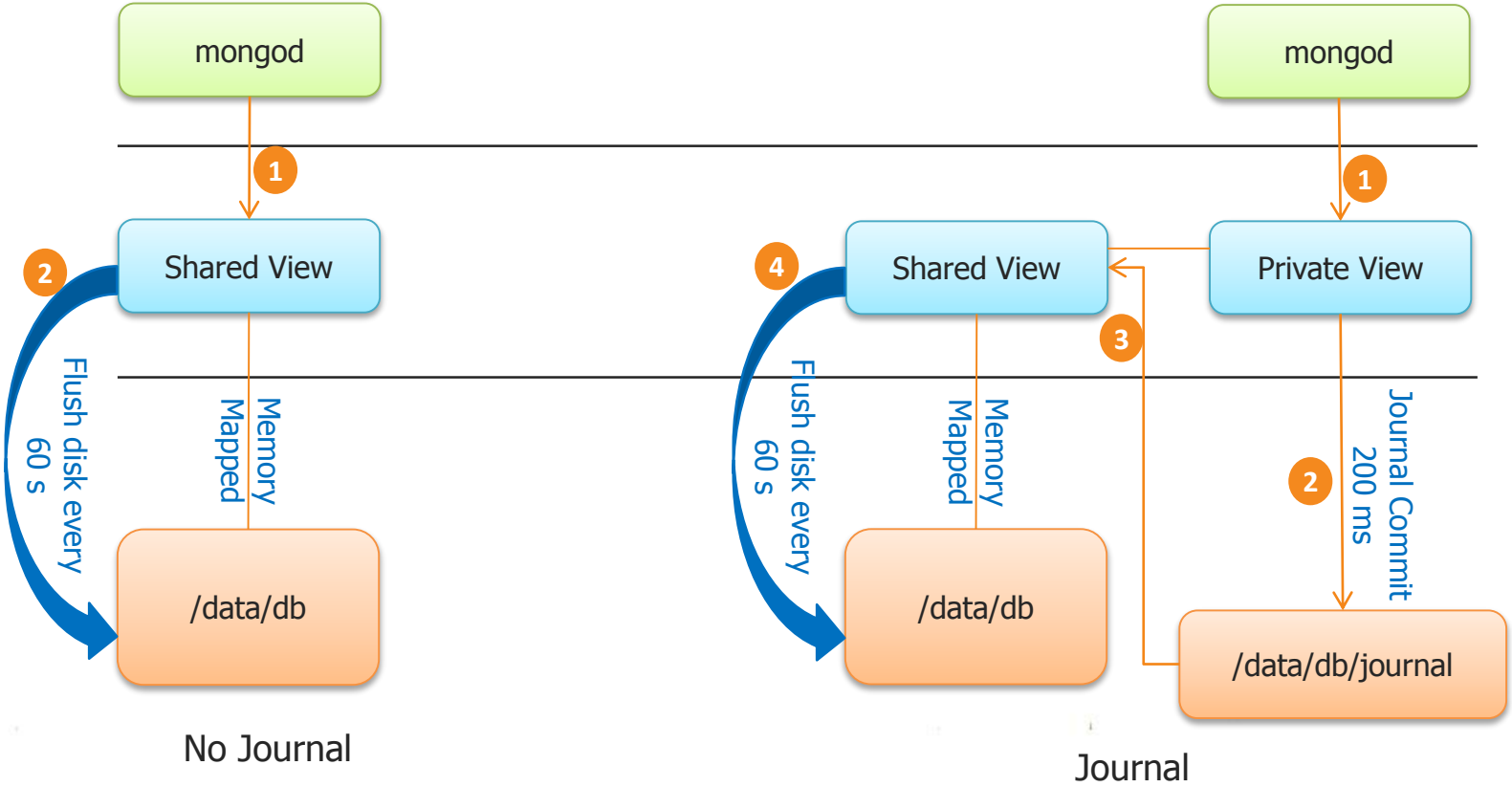
Create in SQL:

```
INSERT INTO  users                  ← table
             (name, age, status) ,  ← columns
VALUES       ("sure", "26", "A") ,  ← values/row
```

Update in SQL:

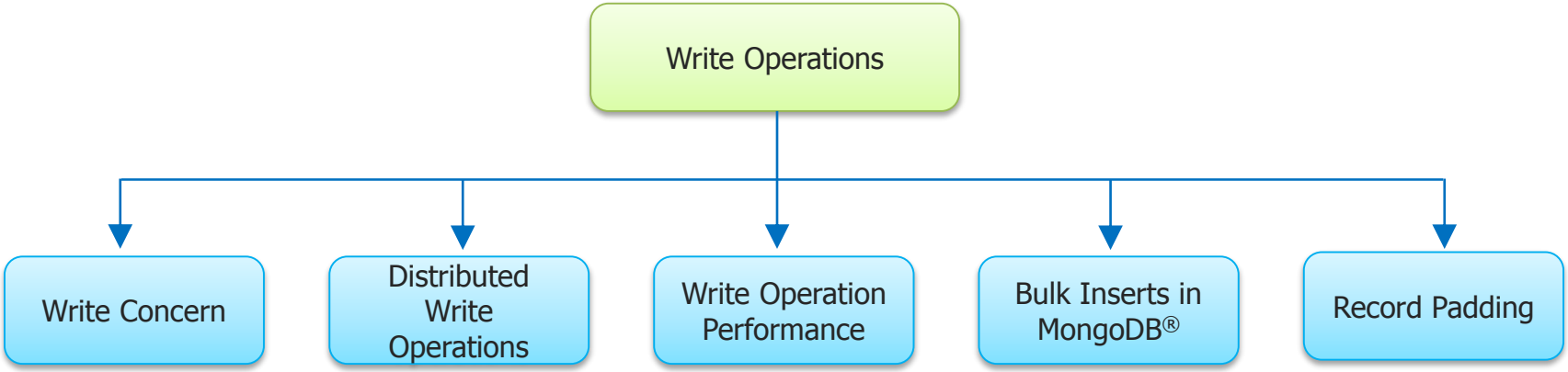
```
UPDATE  users          ← table
SET     status = "A",  ← update action
WHERE   age > 18       ← update criteria
```

Journaling Mechanics



Write Operations

edureka!

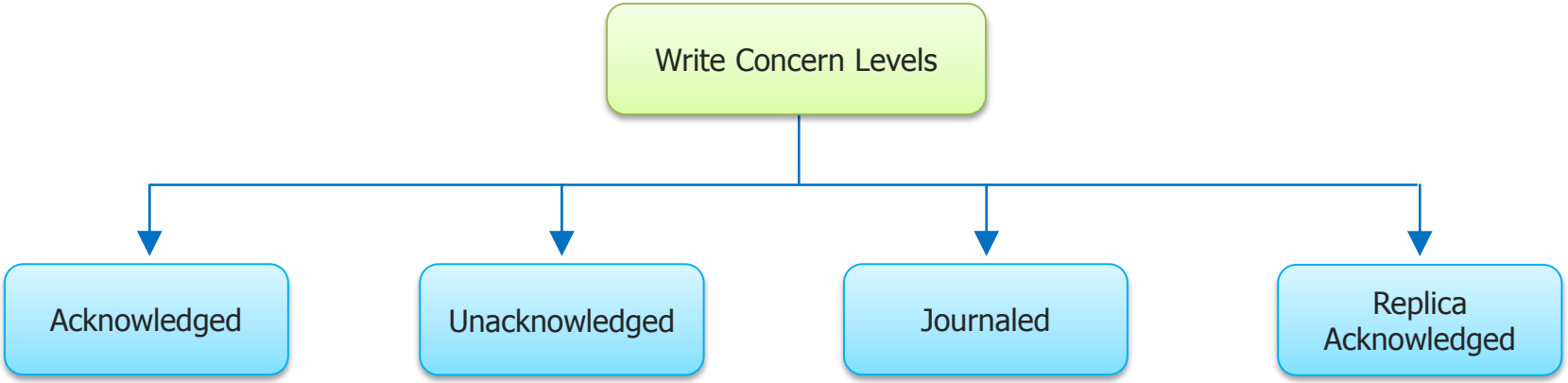


Write Concern

edureka!

- → Write concern describes the guarantee that MongoDB® provides when reporting on the success of a write operation.
- → The strength of the write concerns determine the level of guarantee.
- → When inserts, updates and deletes have a weak write concern, write operations return quickly.
- → In some failure cases, write operations issued with weak write concerns may not persist.
- → With stronger write concerns, clients wait after sending a write operation for MongoDB® to confirm the write operations.

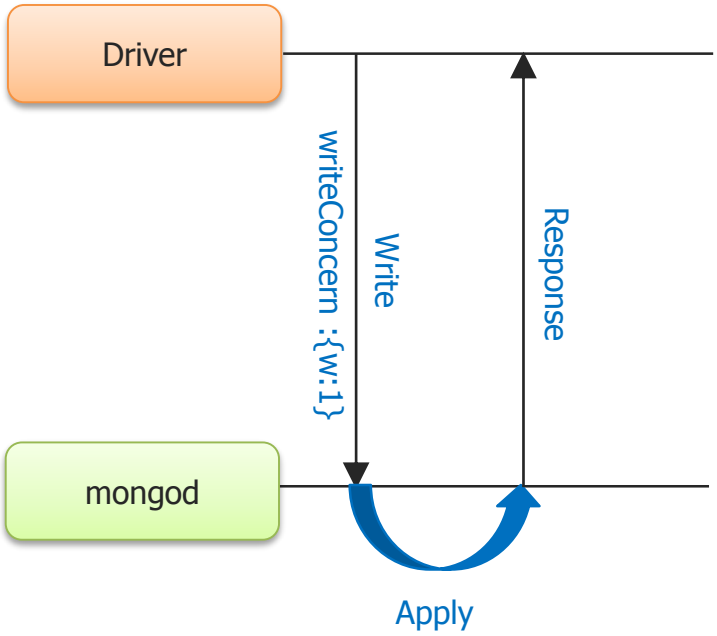
Write Concern Levels



Acknowledged

edureka!

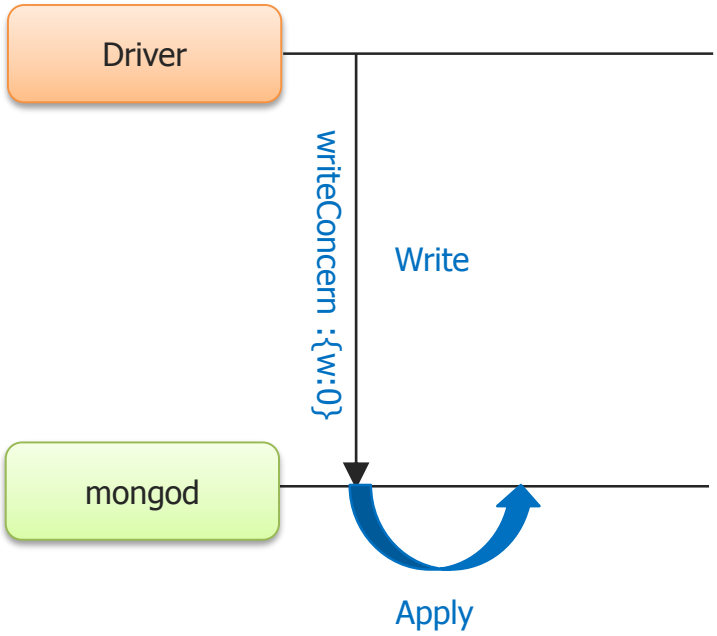
- With a receipt acknowledged write concern, the mongod confirms the receipt of the write operation.
- Acknowledged write concern allows clients to catch network, duplicate key, and other errors.
- To set acknowledged write concern, specify w values of 1 to your driver.
- MongoDB® uses acknowledged write concern by default, after the releases outlined in Default Write Concern Change.
- Write operation to a mongod instance with write concern of acknowledged the client waits for acknowledgment of success or exception.



Unacknowledged

edureka!

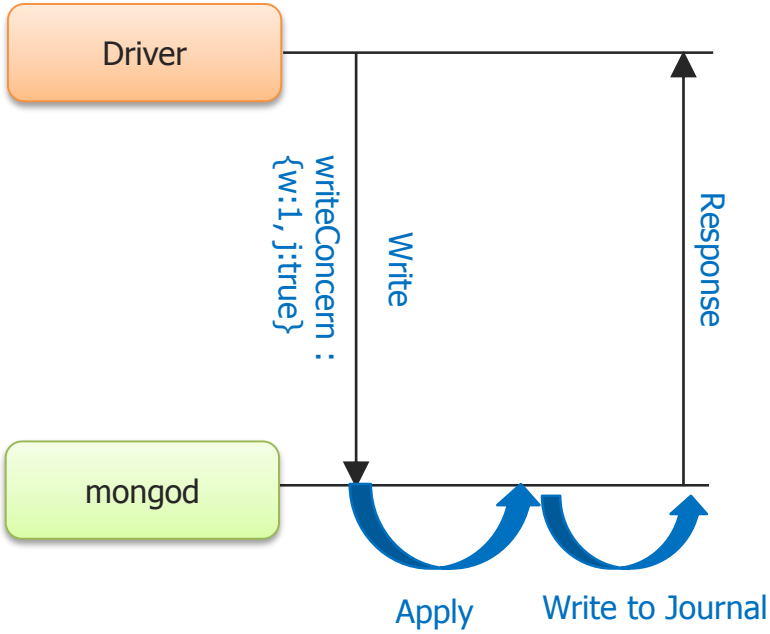
- With an unacknowledged write concern, MongoDB® does not acknowledge the receipt of write operation.
- Unacknowledged is similar to errors ignored; however, drivers attempt to receive and handle network errors when possible.
- Write operation to a mongod instance with write concern of unacknowledged the client does not wait for any acknowledgment.



Journaled

edureka!

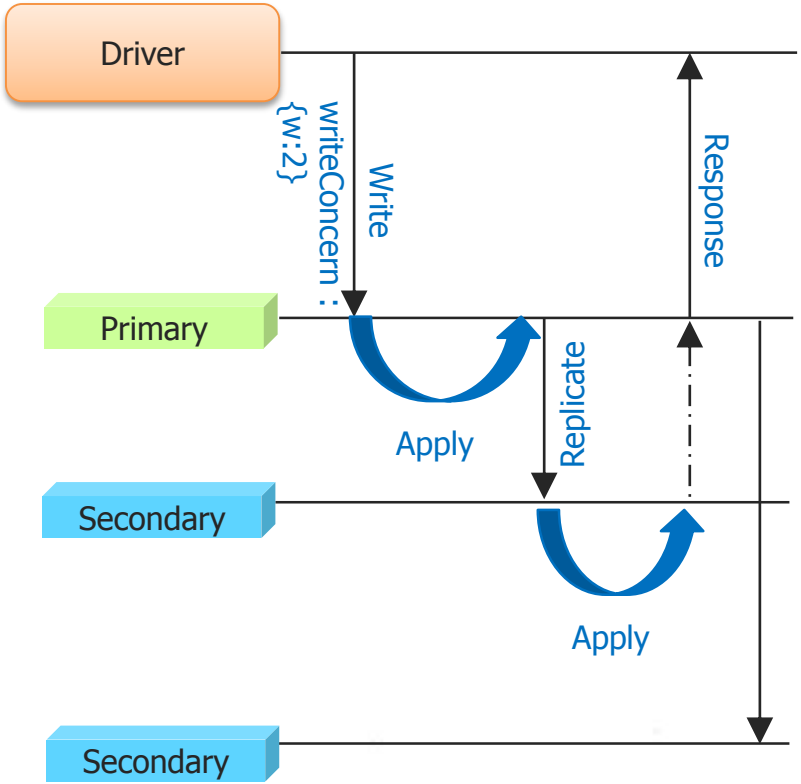
- With a journaled write concern, the mongod acknowledges the write operation only after committing the data to the journal.
- This write concern ensures that MongoDB® can recover the data following a shutdown or power interruption.
- To set a journaled write concern, specify w values of 1 and set the journal or j option to true for your driver.
- With a journaled write concern, write operations must wait for the next journal commit.
- To reduce latency for these operations, you can increase the frequency that MongoDB® commits operations to the journal.
- The mongod sends acknowledgment after it commits the write operation to the journal.



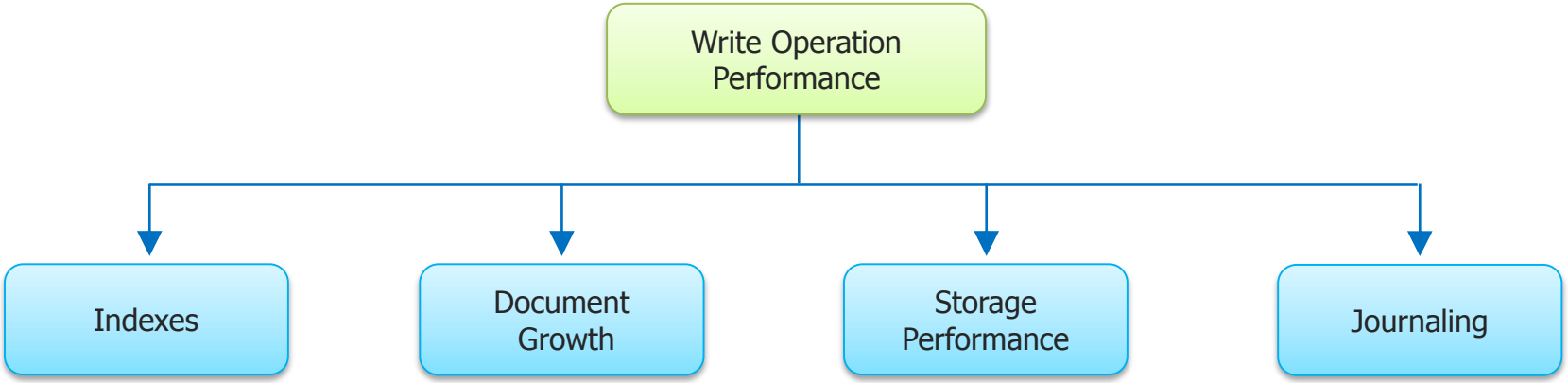
Replica Acknowledged

edureka!

- → Replica sets add several considerations for write concern.
- → Basic write concerns affect write operations on only 1 mongod instance.
- → With replica acknowledged you can guarantee that the write operation propagates to the members of a replica set.
- → To set replica acknowledged write concern, specify w values greater than 1 to your driver.



Write Operation Performance



Annie's Question

What is record padding?

Annie's Answer

edureka!

Padding is a process to minimize document movements. Every document in MongoDB® is stored in a record which contains the document itself and extra space, or padding, which allows the document to grow as the result of updates.

MongoDB® Data Types

edureka!

String: Used to store the string data. String in MongoDB must be UTF-8 valid. (2)	**Arrays**: This type is used to store arrays or list or multiple values into one key. (4)	**Object**: This data type is used for embedded documents.	**Code**: To store java script code into document. (13)	**Time Stamp**: 64 bit value. 1st 32 bit seconds since epoch. 2nd 32 bit is incremental. (17) new Timestamp()
Integer: For numerical value. Integer can be 32 bit or 64 bit depending upon your server. (16/18)	**Boolean**: For a Boolean (true/ false) value. (8)	**Double**: For floating point values. Default type. (1)	**Regular expression**: To store regular expression. (11)	**Null**: This type is used to store a Null value. (10)
Date: For current date or time in UNIX time format. (9) Date() new Date() ISODate()	**Min/ Max keys**: To compare a value against the lowest and highest BSON elements. (-1 / 127)	**Symbol**: It's generally reserved for languages that use a specific symbol type. (14)	**Object ID**: 12 byte BSON. 4 byte timestamp, 3 byte machine, 2 byte process id and last 3 byte counter. (7)	**Binary data**: To store binary data. (5)

db.collection.find({ field: { $type: 7} })

MongoDB® CRUD Syntax and Queries

edureka!

- → Insert Documents
- → Query Documents
- → Limit Fields to Return from a Query
- → Iterate a Cursor in the mongo Shell
- → Analyze Query Performance
- → Modify Documents
- → Remove Documents

MongoDB® CRUD Operation

edureka!

LIVE DEMO

Annie's Question

What is two phase commit?

Annie's Answer

This is used to ensure immediate consistency and reliability.

Assignment

edureka!

→ How to iterate a Cursor in the mongo shell?

→ How to close Inactive Cursors?

→ What is Tailable Cursor?

→ How can you create an Auto-incrementing Sequence Field?

→ What is Record Padding?

→ How many methods are there to insert document in Collection?

→ What is _id field in MongoDB® ?

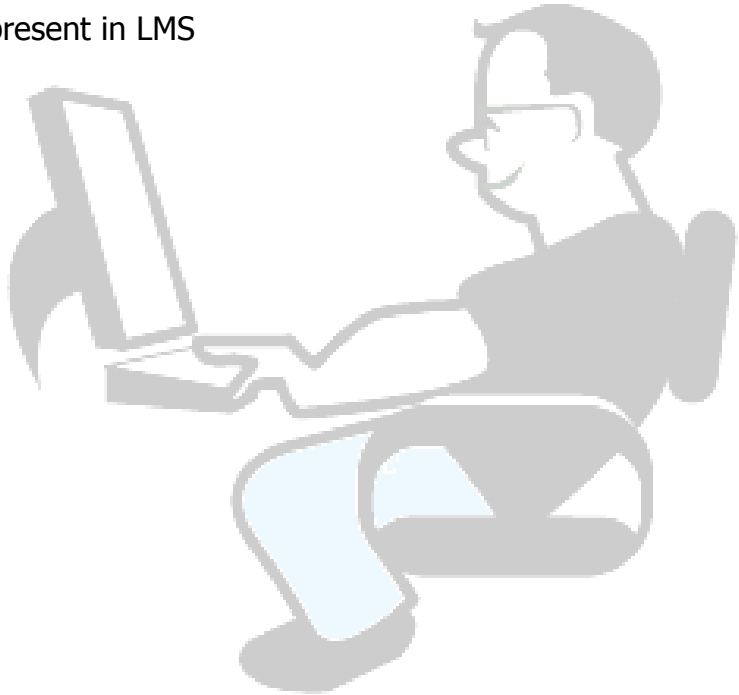
→ What all the operators are there in MongoDB®

→ What is capped collection?

→ What is CRUD?

Pre-work

- Generate Test Data on MongoDB® Database
- Execute all Module2 Script present in LMS
- Read Module 2 FAQ
- Attempt Module 2 Quiz
- Complete assignment



Agenda for Next Class

edureka!

- → Data Modeling Concepts
- → Type of Data Modeling
- → Why Data Modeling ?
- → Data Modeling Approach
- → Analogy Between RDBMS & MongoDB® Data Model
- → MongoDB® Data Model
- → Challenges for Data Modeling in MongoDB®
- → Data Model Examples and Patterns
- → Data Model References
- → Use Case of Data Modeling

Survey

edureka!

Your feedback is important to us, be it a compliment, a suggestion or a complaint. It helps us to make the course better!

Please spare few minutes to take the survey after the webinar.

edureka!