MongoDB

05

Presented by
Subhash EP

Subhash

mongoDB

{ name: mongo, type: DB }

Subhash

Logical operators

- $and

```
db.products.find({$and: [{price: {$lt: 30}},
{name: "Product 2"}]})
```

- $or

```
db.products.find({$or: [{price: {$gt: 40}},
{name: "Product 3"}]})
```

- $not

```
db.products.find({price: {$not: {$gt: 10}}})
```

$exists

- will return all documents that have the specified field in the query.
- db.products.find({sku: {$exists: true}})
  - In this example no documents will be returned as products collection do not have any fields by name sku
- db.products.find({partnership: {$exists: true}})
  - This will return all documents that have field partnership

$regex

- will return all values that match a regular expression
- db.products.find({name: {$regex: /2/}})
  - will return all documents having literal value ‘2’ in anywhere in name fields

$elemMatch

- operator will return all documents where the specified array field values have at least one element that match the query criteria conditions.
  - Example: 0126.txt

Other Operators

- $where
  - Matches documents that satisfy a JavaScript expression. (see example – 0127.txt)
- $all
  - Matches arrays that contain all elements specified in the query. (see example – 0128.txt)

Write Operations

- insert
- update
- remove

Writer interfaces

- db.document.insert
- db.document.update
- db.document.remove

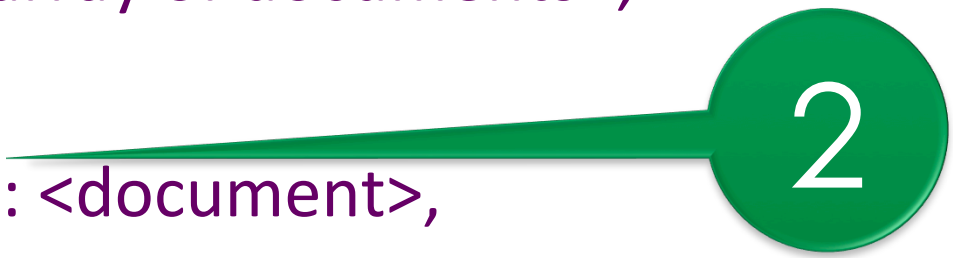
Write Operations

- The write operations in MongoDB
  - targeted to a specific collection
  - are atomic on the level of a single document.

Inserts

- The syntax:

```
db.collection.insert(
   <document or array of documents>,
   {
      writeConcern: <document>,
      ordered: <boolean>
   }
)
```



1. document or array of documents

- is either a document or an array
- with one or many documents
- that should be created in the targeted collection

2. writeConcern

- is a document expressing the write concern
- *(this will be explained in detail later)*

ordered

- a Boolean value,
  - if true, ordered process on the documents of the array
  - if there is an error in a document,
    - MongoDB will stop processing it.
  - if the value is false,
    - it will carry out an unordered process
    - it will not stop if an error occurs.
- By default, the value is true.

Example

- 0201.txt

Write concerns

- Traditionally, DB professionals are familiar with ACID.
- Let us talk about D in ACID

Durability

- Durability in database systems is a property that tells us
  - whether a write operation was successful,
  - whether the transaction was committed, and
  - whether the data was written on non-volatile memory in a durable medium, such as a hard disk.

NoSQL ≠ RDBMS

- Unlike relational database systems,
  - the response to a write operation in NoSQL databases is determined by the client.
- Once again, we have the possibility to make a choice on our data modeling, addressing the specific needs of a client.

WriteConcern

- In MongoDB, the response of a successful write operation can have many levels of guarantee.
- This is what we call a write concern.

WriteConcern

- The levels vary from weak to strong, and the client determines the strength of guarantee.
- It is possible for us to have, in the same collection, both a client that needs a strong write concern and another that needs a weak one.

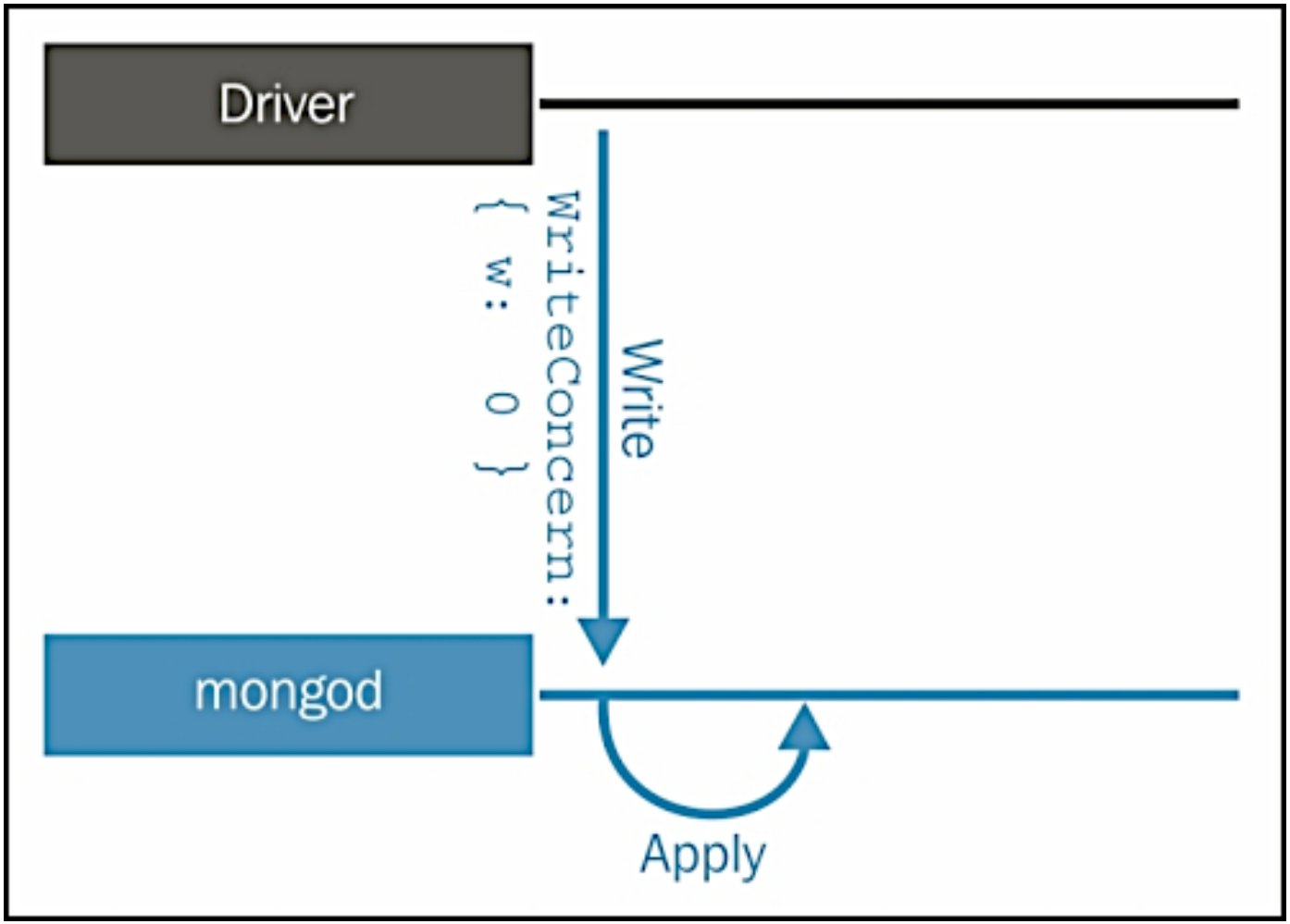
WriteConcern Levels

- Unacknowledged
- Acknowledged
- Journaled
- Replica acknowledged

Unacknowledged

- the client will not attempt to respond to a write operation.
- If this is possible, only network errors will be captured

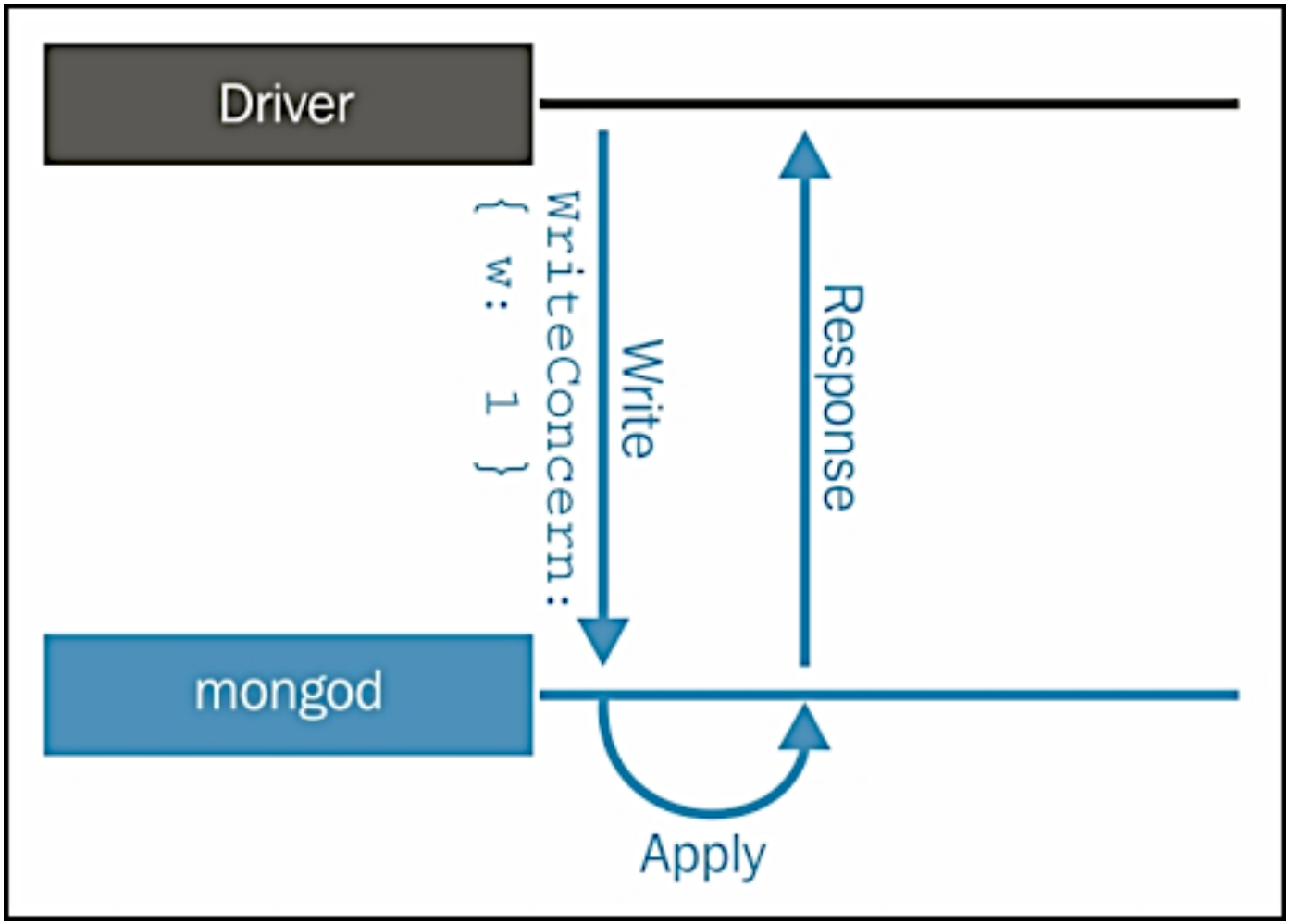
Unacknowledged



Acknowledged

- the client will have
  - an acknowledgement of the write operation,
  - and see that it was written on the in-memory view of MongoDB.
- In this mode,
  - the client can catch,
  - among other things,
  - network errors and duplicate keys

Acknowledged



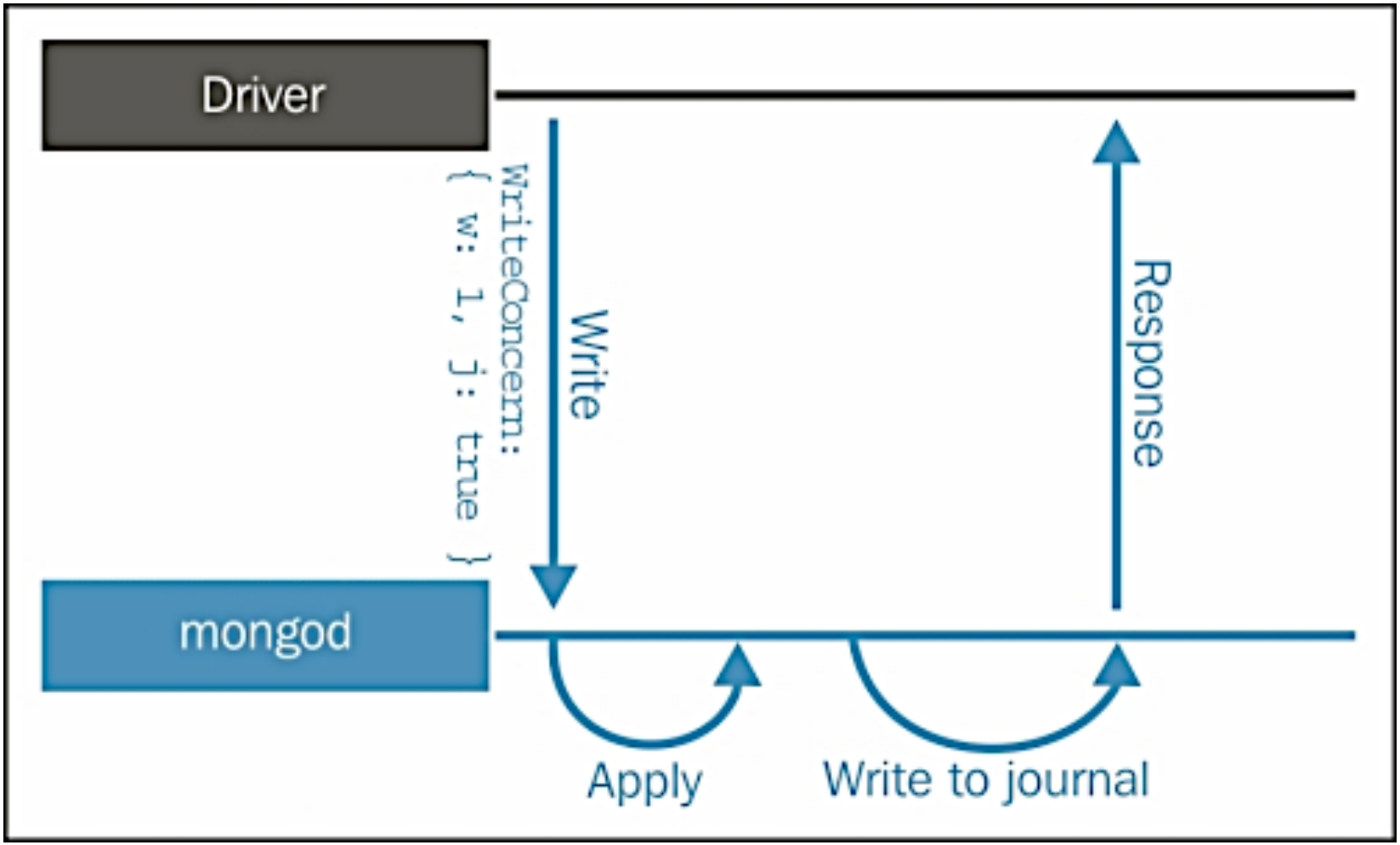
Journaled

- client will receive confirmation that the write operation was committed in the journal.
- Thus, the client will have a guarantee that the data will be persisted on the disk,
  - even if something happens to MongoDB

Journaled

- To reduce the latency when we use a journaled write concern,
- MongoDB will reduce the frequency in which it commits operations to the journal from the default value of 100 milliseconds to 30 milliseconds

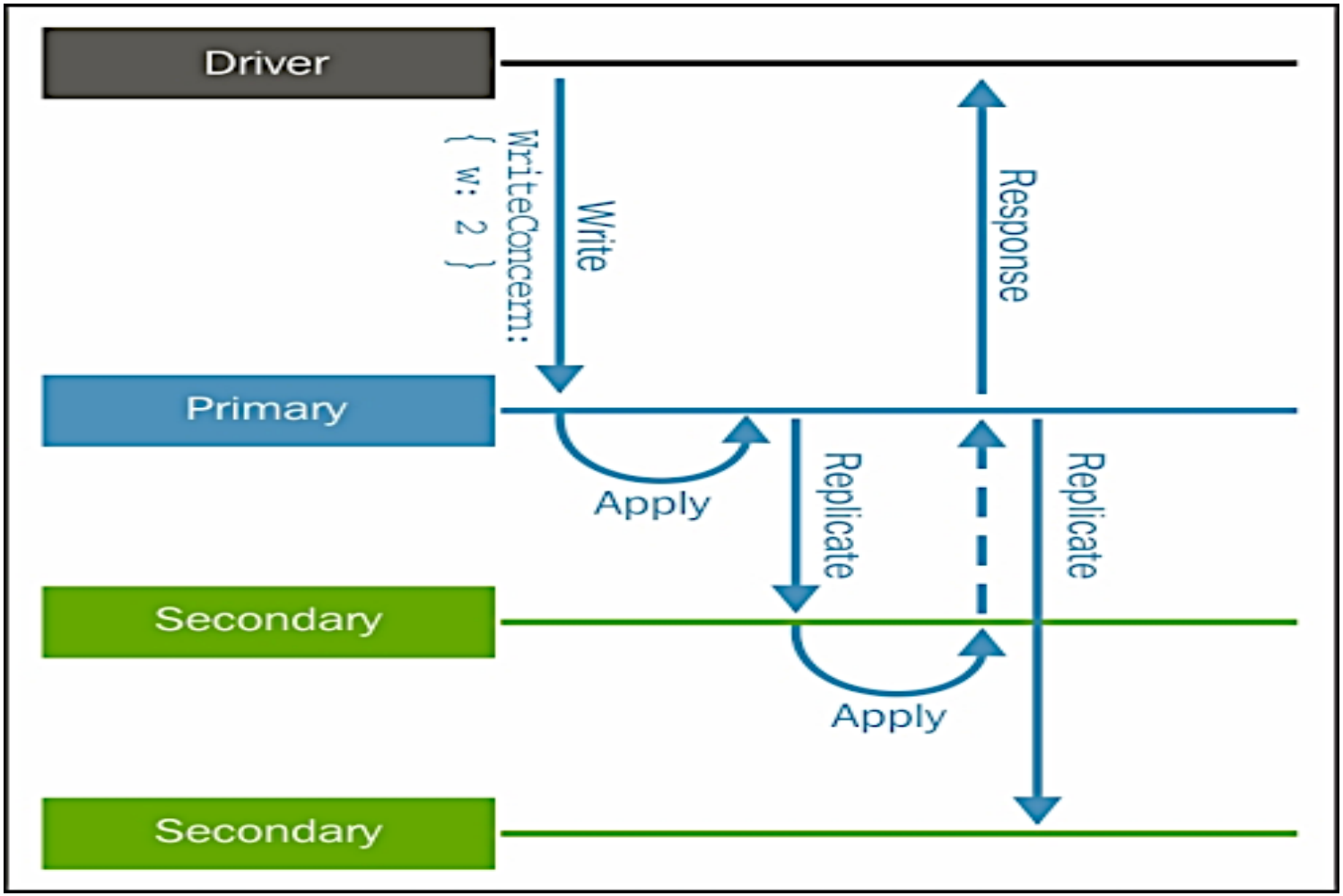
Journaled


Driver
WriteConcern:
{ w: 1, j: true }
Write
Response
mongod
Apply
Write to journal

Replica acknowledged

- When we are working with replica sets,
  - it is important to be sure that a write operation was successful not only in the primary node,
  - but also that it was propagated to members of the replica set.
- For this purpose, we use a replica acknowledged write concern.

Replica Acknowledged



Bulk writing documents

- Sometimes it is quite useful to
  - insert, update, or delete
  - more than one record of your collection.
- MongoDB provides us with the capability to perform bulk write operations.
- A bulk operation works in a single collection,
  - and can be either ordered or unordered.

Bulk writing documents

- As with the insert method,
  - the behavior of an ordered bulk operation is to process records serially,
  - and if an error occurs, MongoDB will return without processing any of the remaining operations.

Example

```
db.customers.insert(
[
{username: "customer3", email: "customer3@customer.com",
password: hex_md5("customer3paswd")},

{username: "customer2", email: "customer2@customer.com",
password: hex_md5("customer2paswd")},

{username: "customer1", email: "customer1@customer.com",
password: hex_md5("customer1paswd")}
]
)
```

That's all

End of Session

©

subhash.ep@gmail.com
linkedin.com/in/subhashep