MongoDB Specification in Maude

LAC 2025

BEATRIZ ALCAIDE

MongoDB

- DBMS
- NoSQL
- Distributed
  - Sharding
  - Replication
- Document-based
  - JSON-like documents

Why specify MongoDB?

- It dominates the No-SQL space:
  - EA
  - Toyota
  - Google
  - Uber
  - Adobe

Documents

Example:

```
{
   name:
   {
     first: “Jane”,
      last: “Doe”
   }
   age: 28,
   groups: [“A”, “B”, “C]
}
```

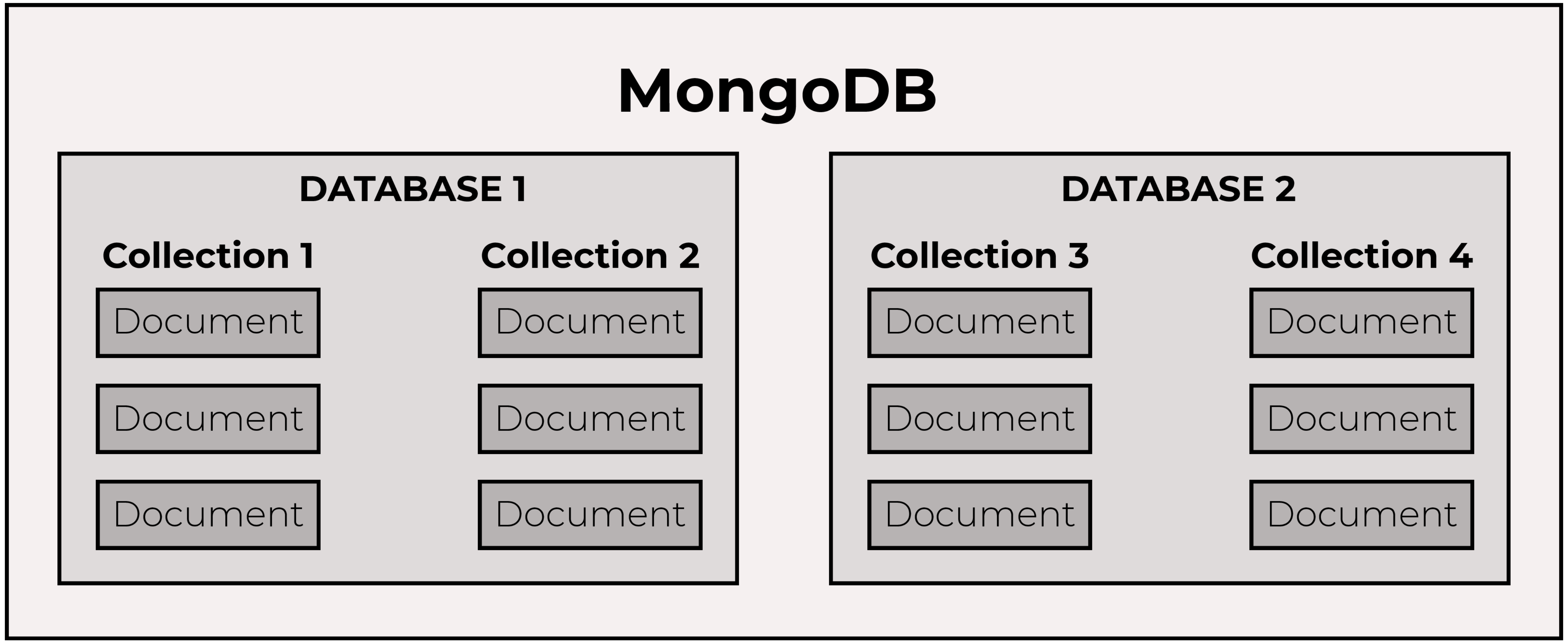
Quering Documents

MONGODB	SQL equivalence

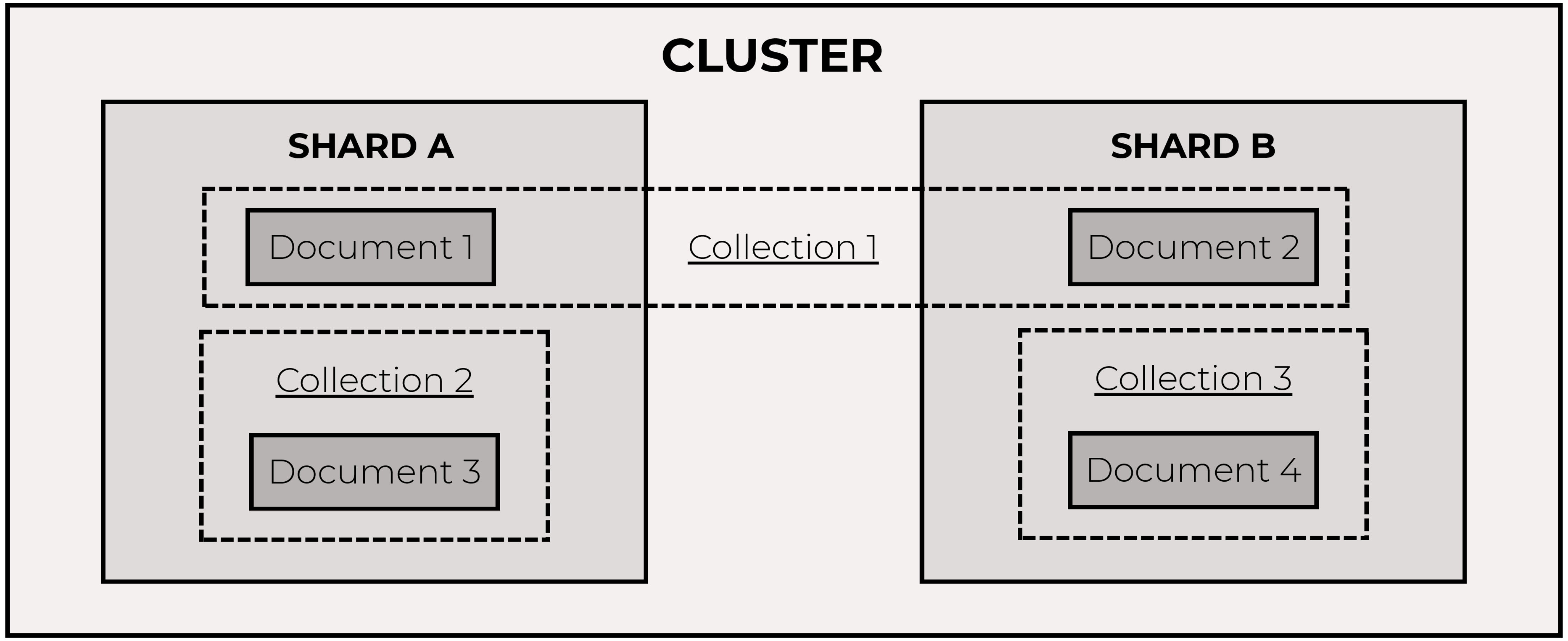
db.collection('**inventory**').find(
{
status: 'D'
}
);
	SELECT * FROM inventory
	WHERE
	status = "D"

db.collection('**inventory**').find(
{
status: 'A',
qty: { $lt: 30 }
}
);
	SELECT * FROM **inventory**
	WHERE
	status = "A"
	AND
	qty < 30

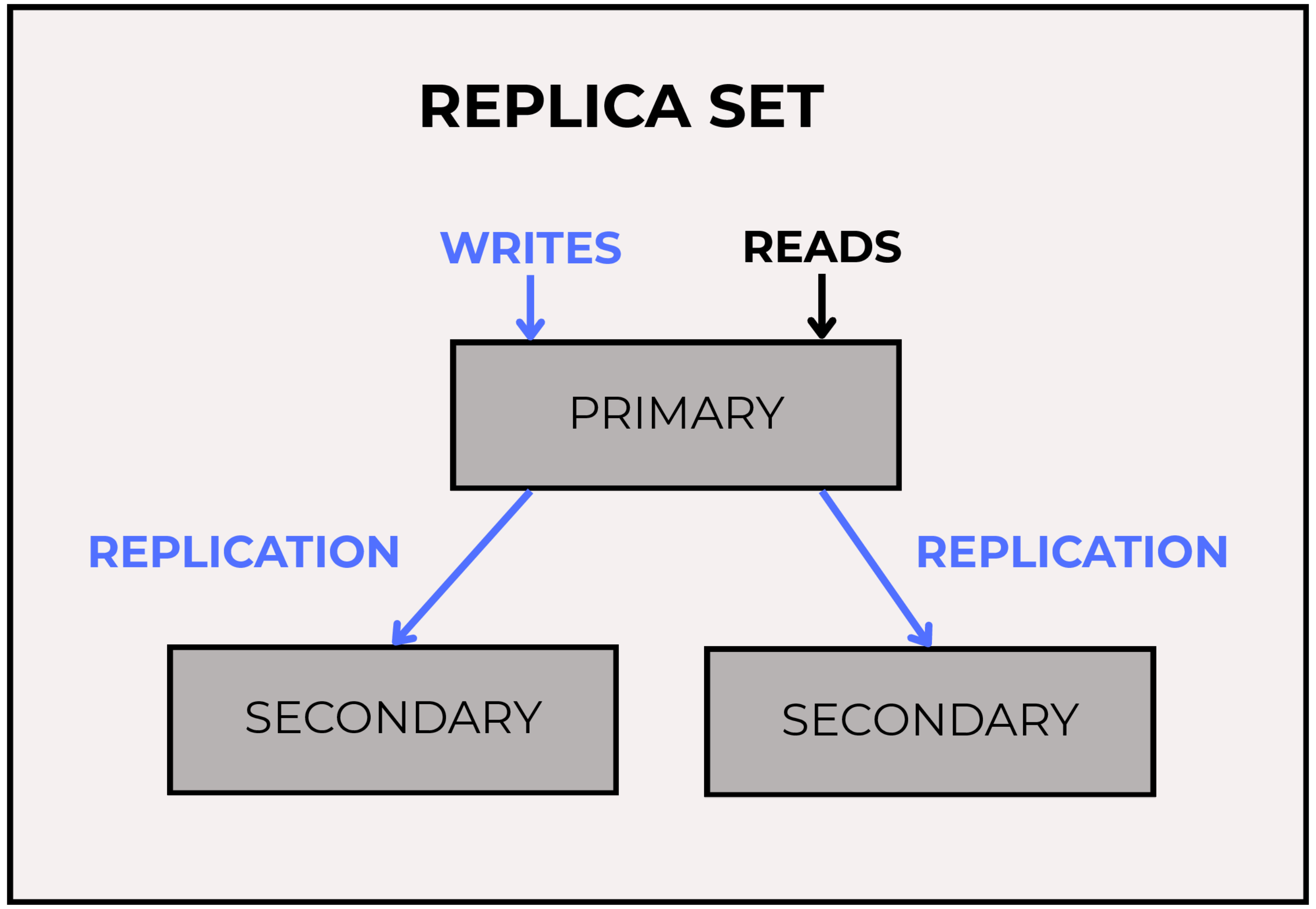
MongoDB's Data Model



Sharding


CLUSTER
SHARD A
SHARD B
Document 1
Collection 1
Document 2
Collection 2
Document 3
Collection 3
Document 4

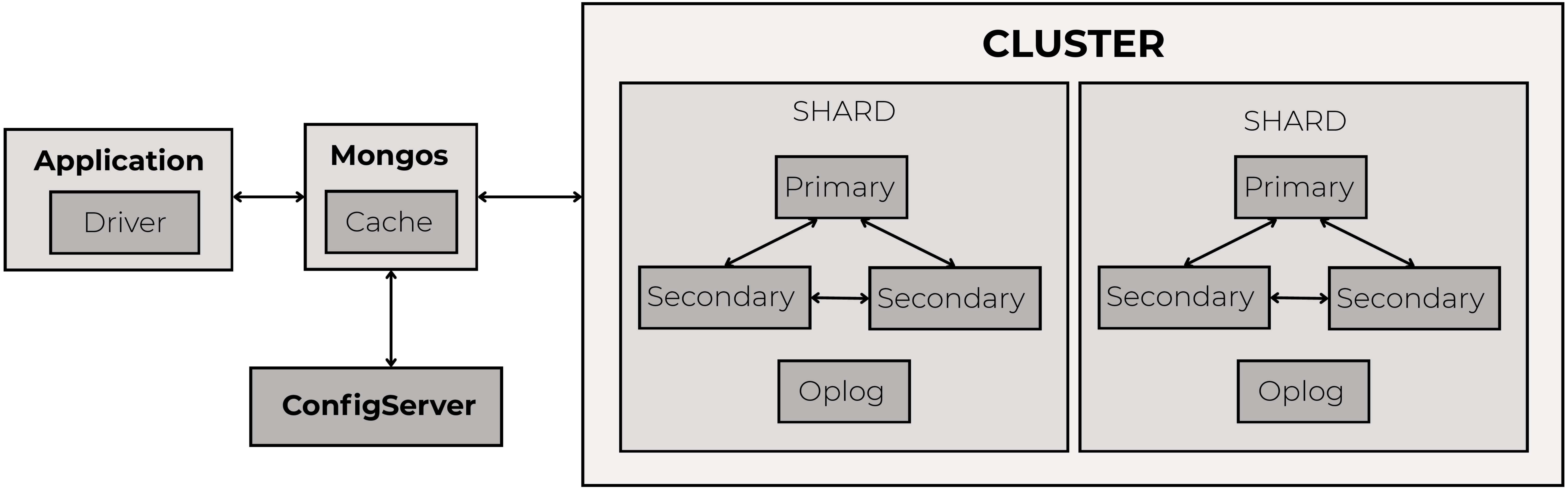
Replication



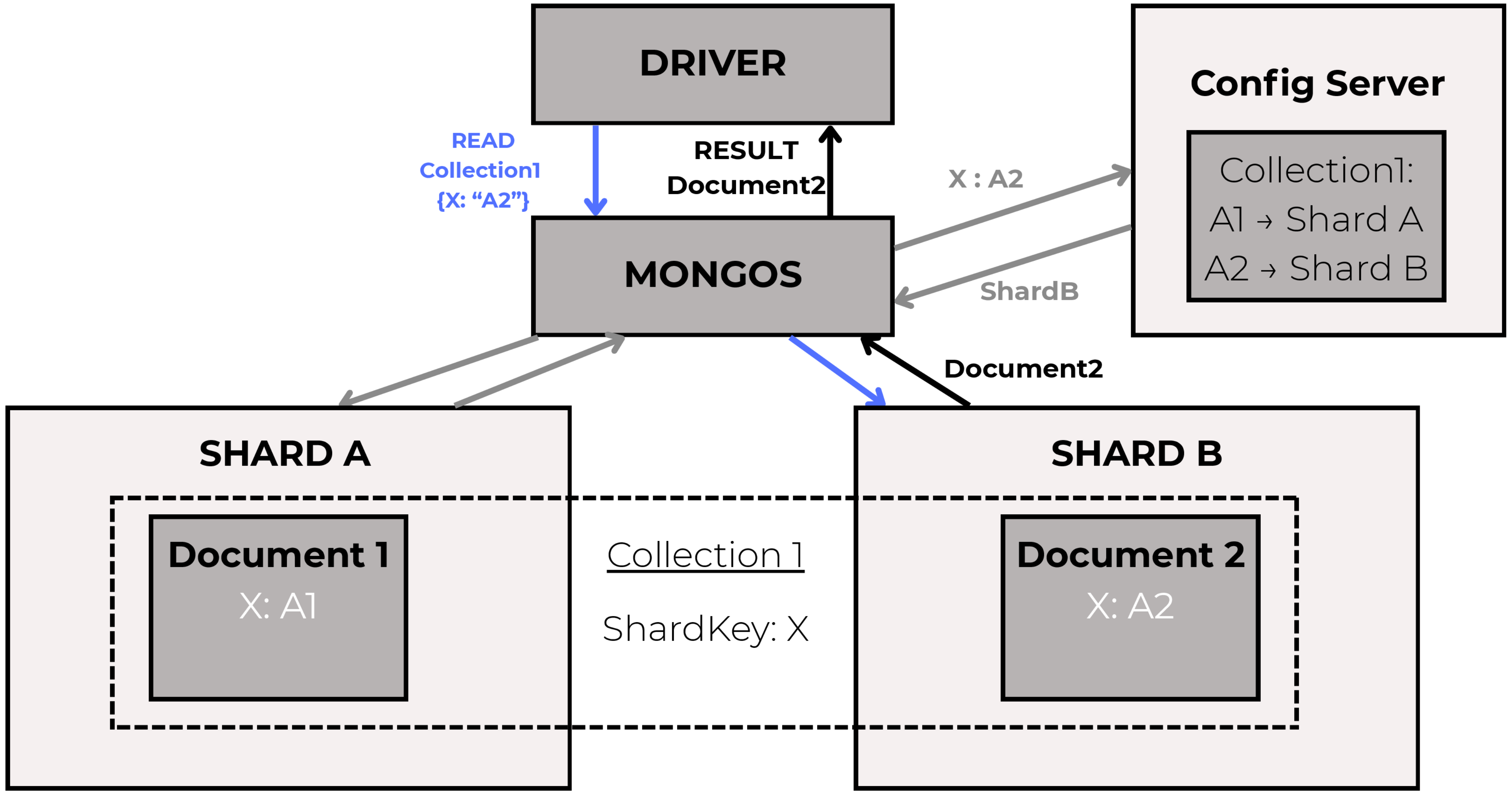
The **primary node** receives all the write operations

Secondary nodes read the write operations from an operation log (**Oplog**)

MongoDB's Architecture



Routing



MongoDB uses **Shard Keys** to route sharded queries, which can be:

- Hashed-Based
- Range-Based

Problem

An overload on the config servers can create a **bottleneck**.

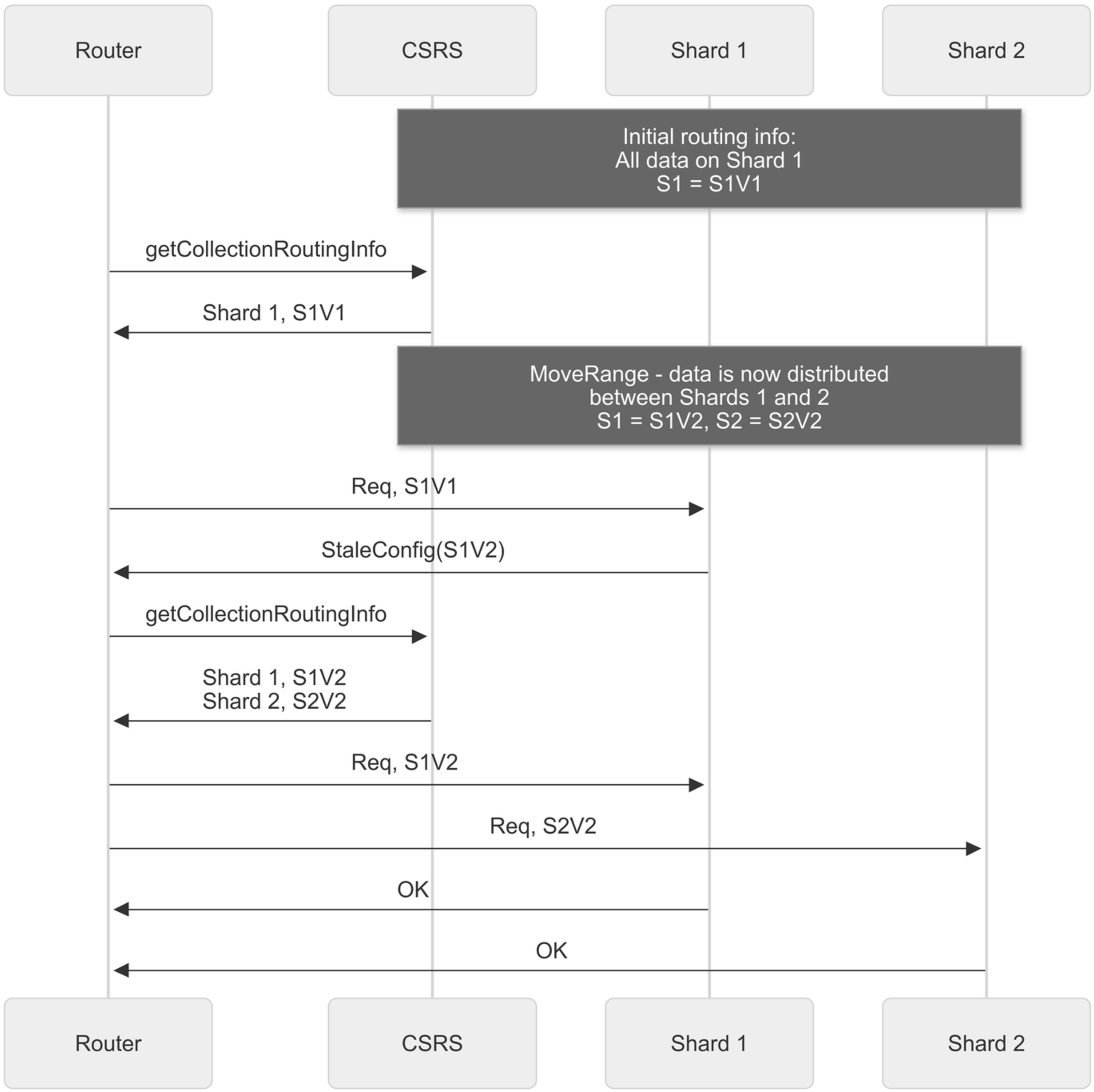
Solution

Mongos uses a **cache** memory to locally store cluster metadata.

How do we know that the metadata in the **cache** memory is **up-to-date**?

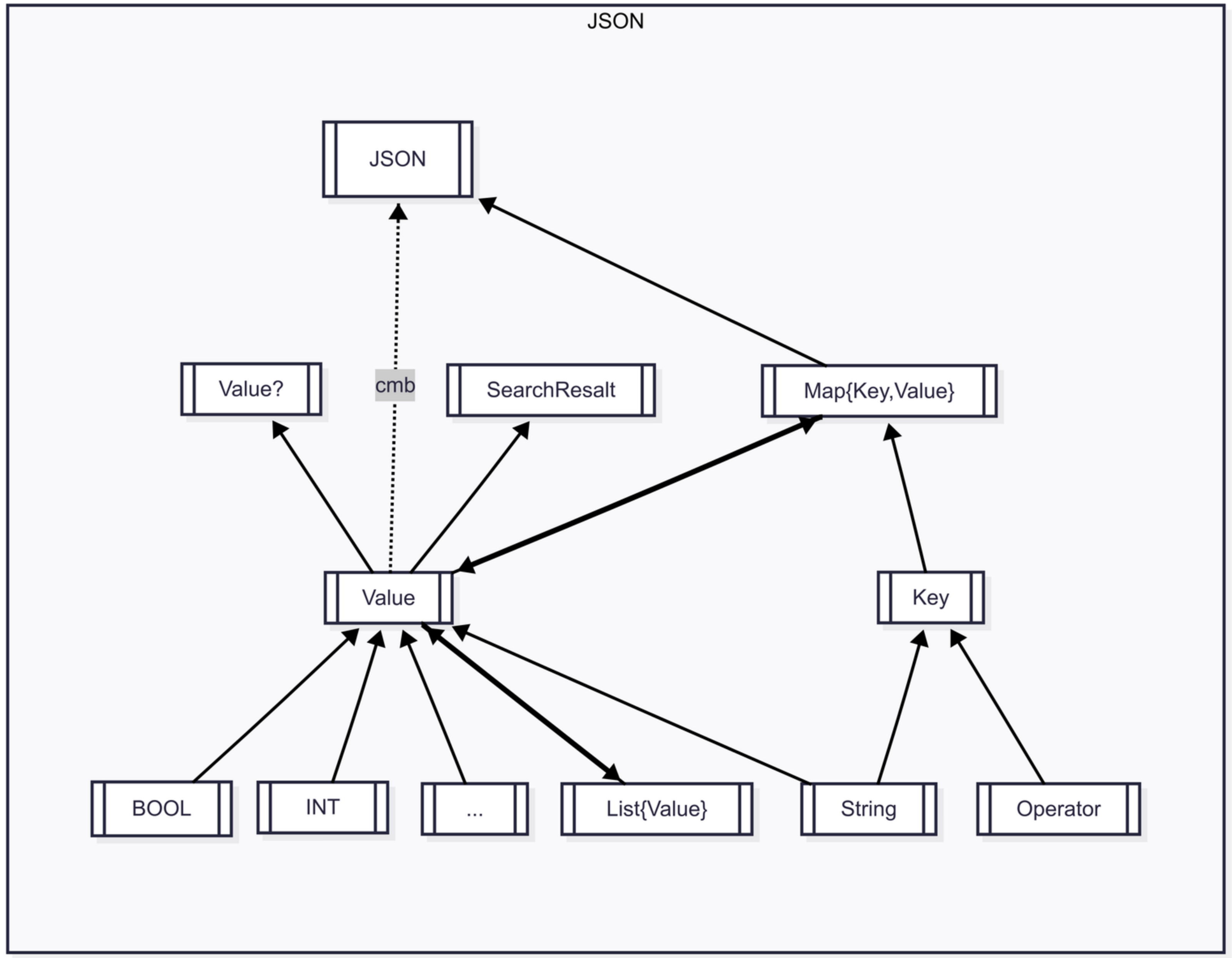
Versioning Protocol

Versioning Protocol



The **routing info** tells the router the set of shards that should be contacted.

JSONs


JSON
JSON
Value?
cmb
SearchResalt
Map{Key,Value}
Value
Key
BOOL
INT
...
List{Value}
String
Operator

Collection

sort UnshardedCollection ShardedCollection Collection .
subsorts UnshardedCollection ShardedCollection **<** Collection .

op UC : Map{Nat, JSON} Nat Nat -> UnshardedCollection [ctor] .
***nat1 = objectID counter, nat2 = version

op SC : Map{Nat, JSON} Nat -> ShardedCollection [ctor] .

Database

fmod DATABASE is
pr MAP{String, Collection} * (sort Map{String, Collection} to Database) .
endfm

fmod DATABASEMAP is
pr MAP{String, Database} * (sort Map{String, Database} to DatabaseMap) .
endfm

Node

ops primary secondary : -> NodeType [ctor] .

op node : NodeType DatabaseMap Nat -> Node [ctor] .
***nat = counter

ReplicaSet

op RS : Node NodeList UnshardedCollection -> ReplicaSet [ctor].

fmod SHARDMAP is
 pr MAP{Nat, ReplicaSet} * (sort Map{Nat, ReplicaSet} to ShardMap) .
endfm

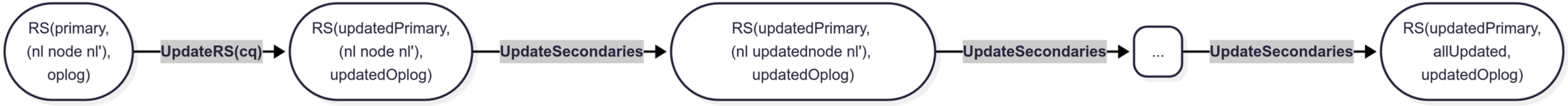
Cluster

op CL : Nat ShardMap -> Cluster [ctor] . ***Nat = idCounter

ShardKey

op hashBased rangeBased : -> TypeShardKey [ctor] .
op SK : TypeShardKey ShardValues -> ShardKey [ctor] .

Update Replica Set



Asynchronous message-passing communication model

Cache

Namespace:

Database name and Collection name

fmod METADATA-CACHE is

pr MAP{NameSpace, Routing-Info} * (sort Map{NameSpace, Routing-Info} to Cache) .

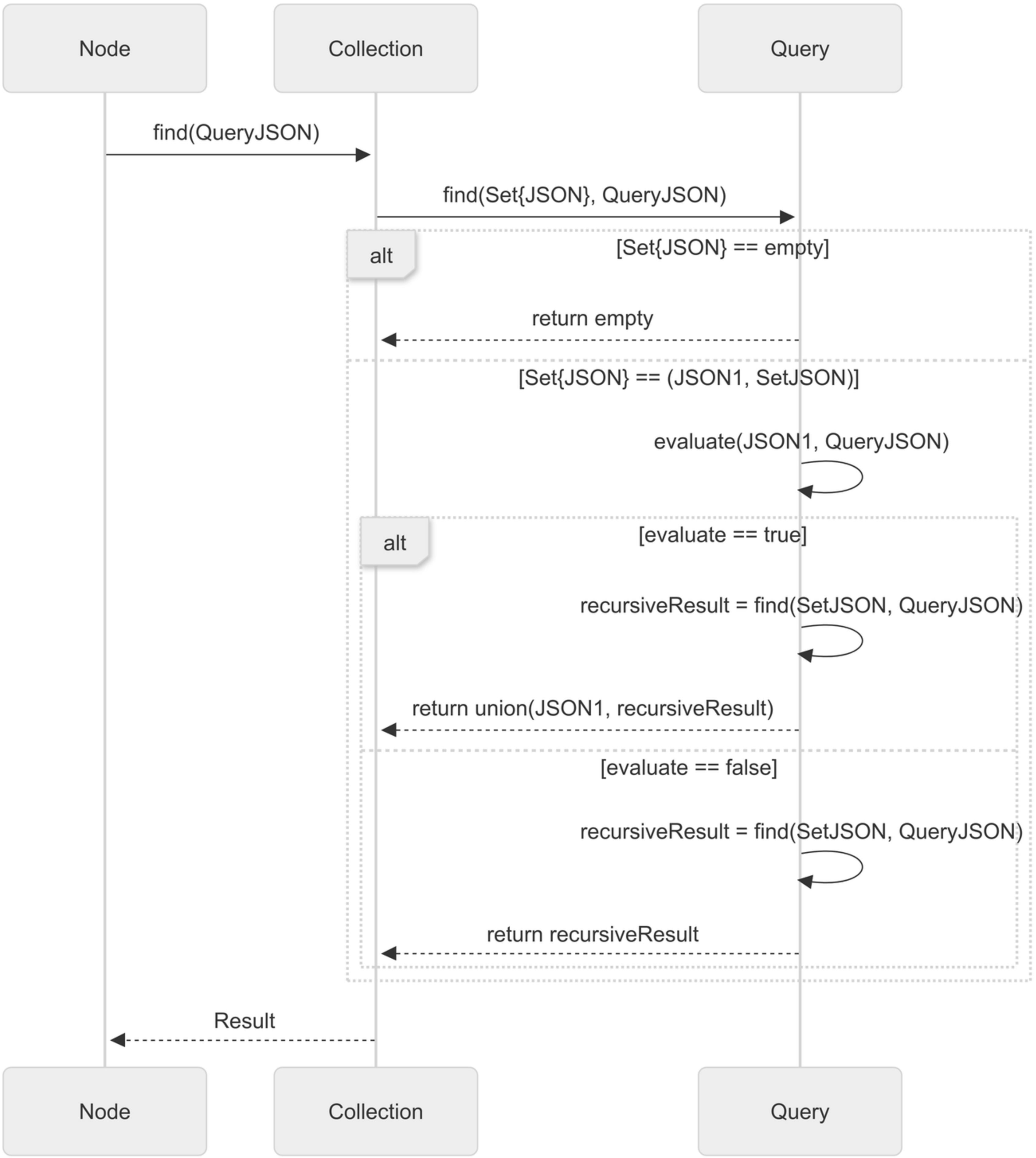
endfm

Mongos

op MGS : Cache ReplicaSet -> Mongos [ctor] . ***replicaset = configServer

Queries

The find function receives a **JSON-like** object that **defines** the **conditions** documents must meet to be included in the query result.



Future work

- Conformance testing.
- Consensus algorithm.
- Expand query operators.
- Indexes.
- Model check.

Thank you!