NoSQL
Infrastruktura za elektronsko poslovanje

dr Miloš CVETANOVIĆ
dr Zaharije RADIVOJEVIĆ

SQL	Mongo (NoSQL)
database	database
table	collection
row	document or BSON document
column	field
index	index
table joins	embedded documents, linking
primary key	primary key (_id)
group by	aggregation framework

Osobine:

- Dokument orjentisana baza podataka
- Podrška za dinamičke upite
- Podrška za različite tipove indeksa
- Podrška za replikaciju i visoki nivo dostupnosti
- Podrška za horizontalnu skalabilnost
- Podrška za atomične modifikacije
- Podrška za map/reduce poslove
- Podrška za čuvanje velikih dokumenata

Relacioni pristup

```
|_media
      |_cds
          |_id, artist, title, genre, releasedate
      |_ cd_tracklists
          |_cd_id, songtitle, length
```

Ne relacioni pristup

```
|_media
      |_items
          |_<document>
```

```
{
"Type": "CD",
"Artist": "Nirvana",
"Title": "Nevermind",
"Genre": "Grunge",
"Releasedate": "1991.09.24",
"Tracklist": [
            { "Track" : "1",
              "Title" : "Smells Like Teen Spirit",
              "Length" : "5:02"},
            { "Track" : "2",
              "Title" : "In Bloom",
              "Length" : "4:15"}
            ]
}
```

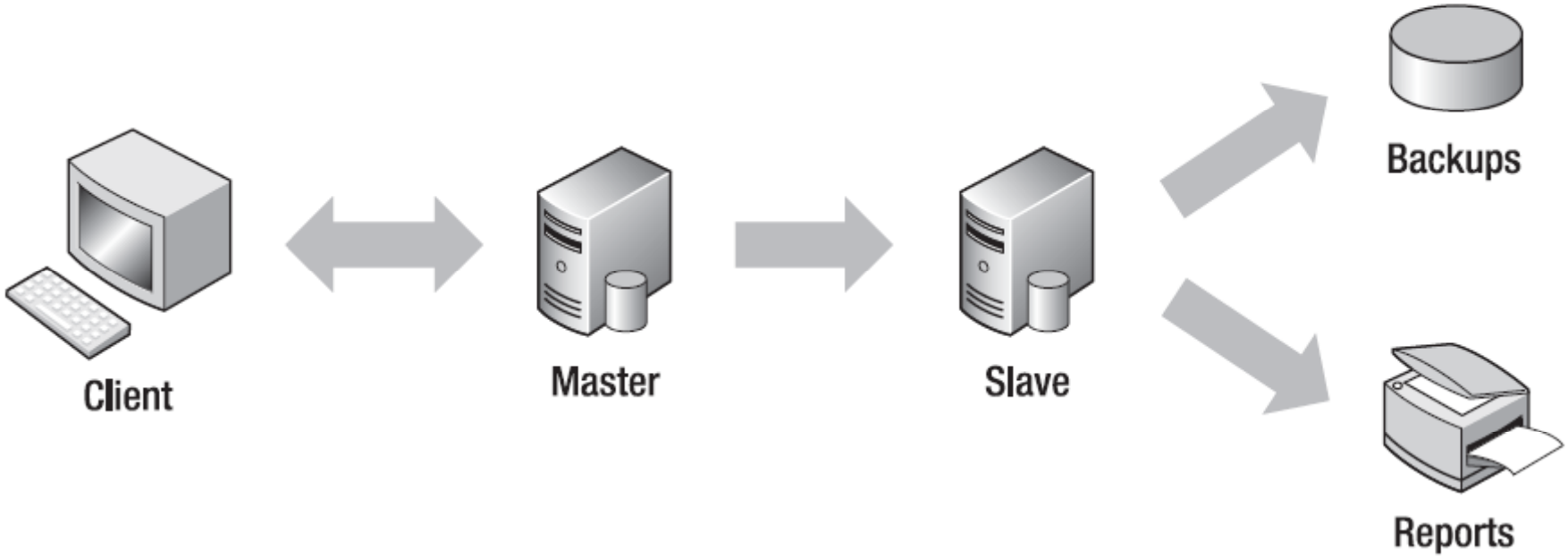
Binarni JSON (JavaScript Object Notation)→ BSON

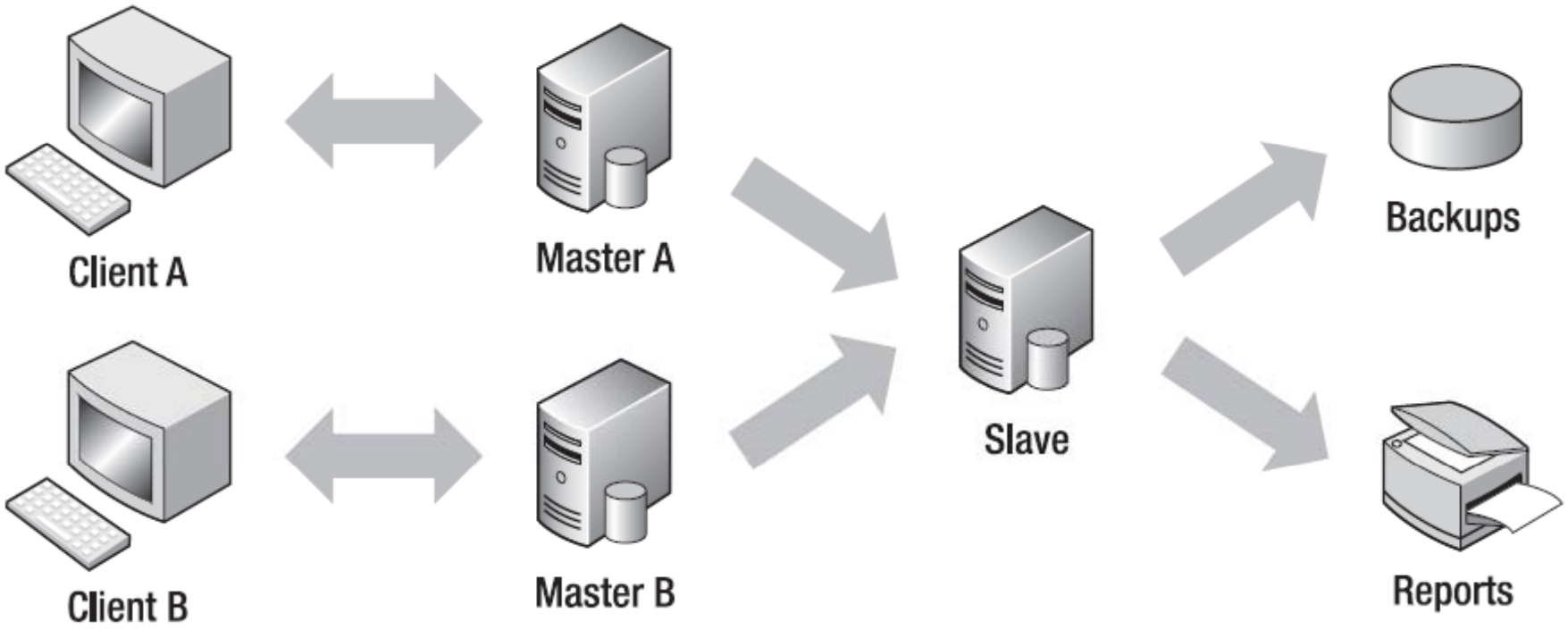
{_id: ObjectId(XXXXXXXXXXXX), Hello: “world”}

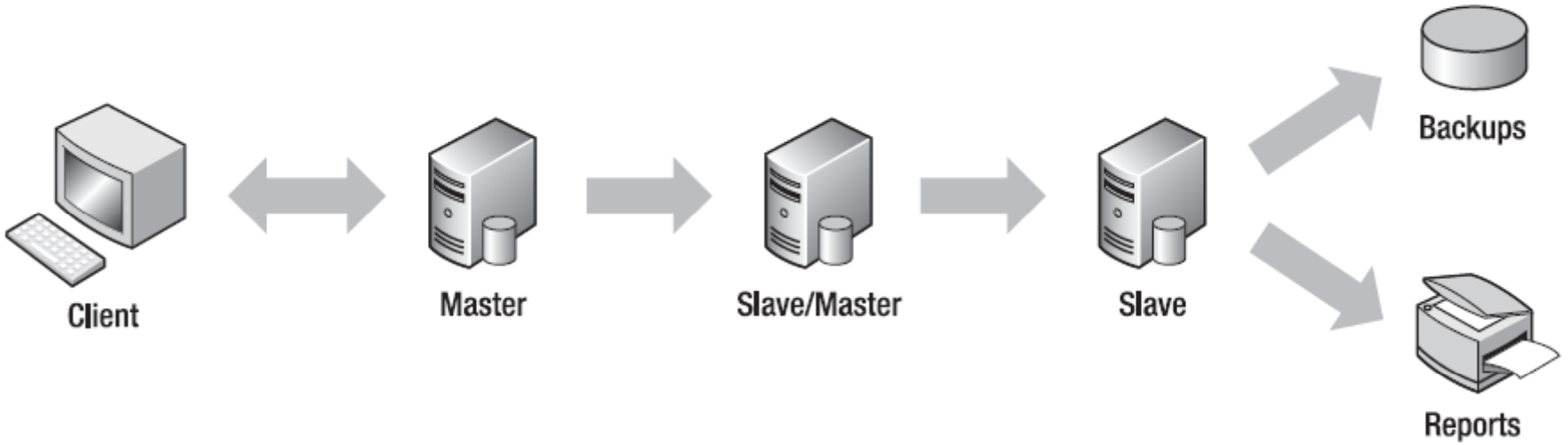
\x27 \x00 \x00 \x00 \x07 _ i d \x00
X X X X X X X X X X X X
\x02 h e l l o \x00
\x06 \x00 \x00 \x00 w o r l d \x00
\x00

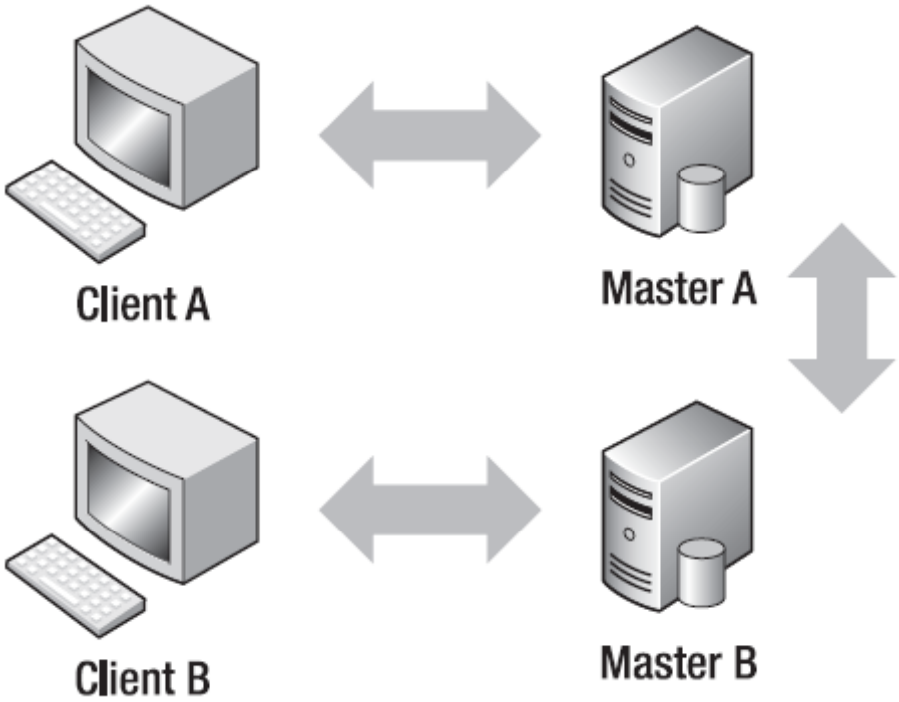
_id polje – 12B (automatski ili pozivom ObjectId())

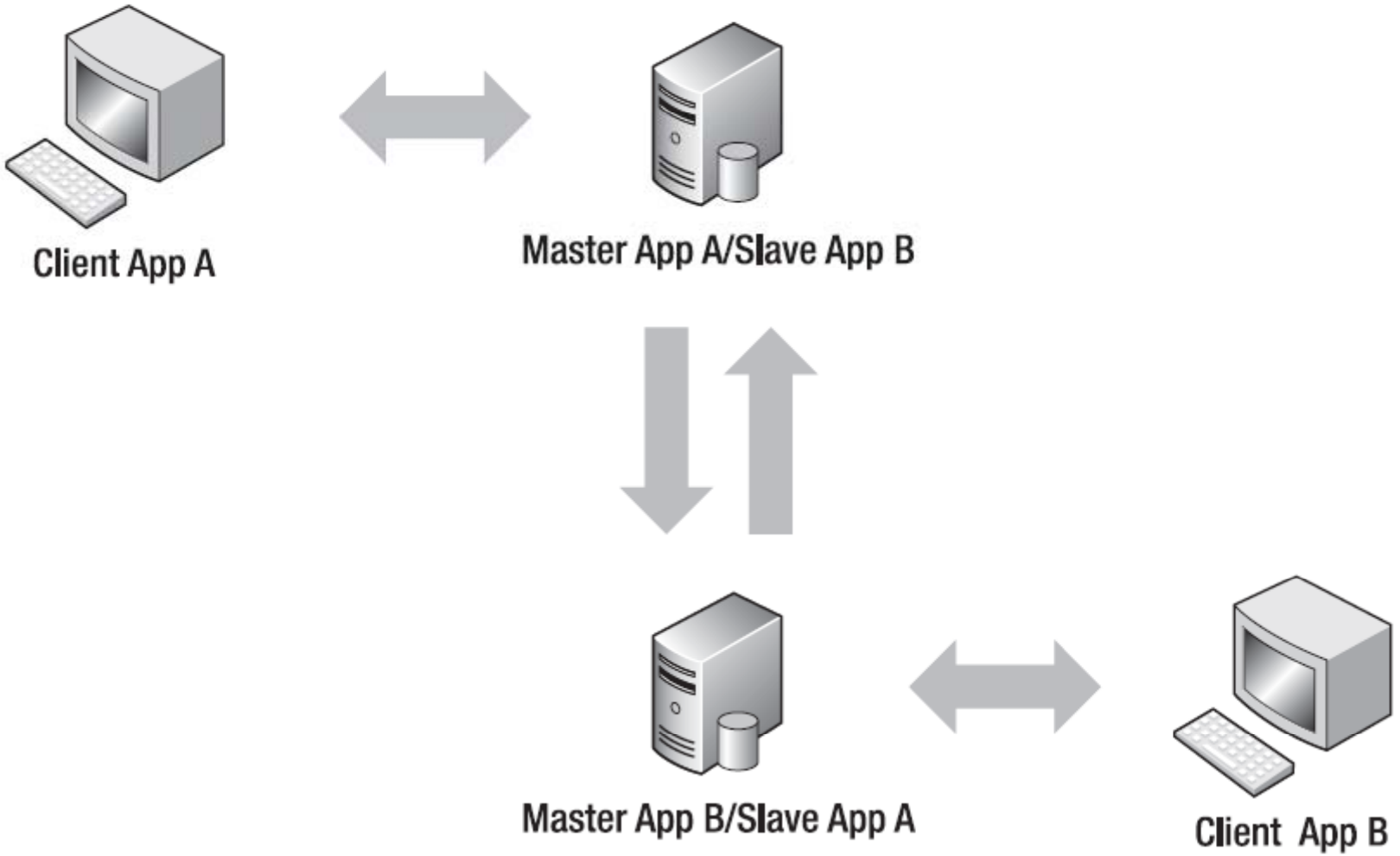
0	1	2	3	4	5	6	7	8	9	10	11
Vreme				Mašina			Proces		Brojač		

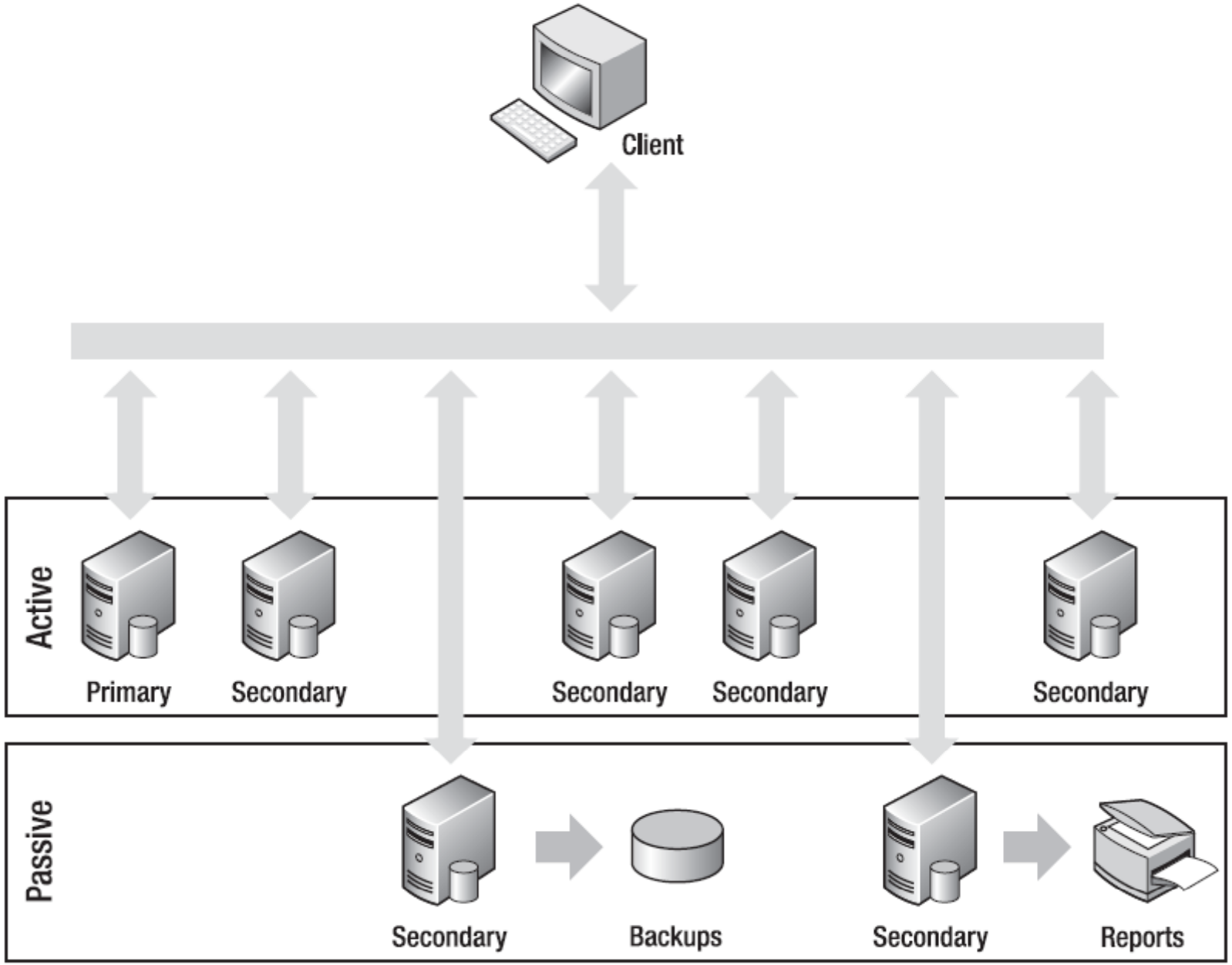

Client
Master
Slave
Backups
Reports

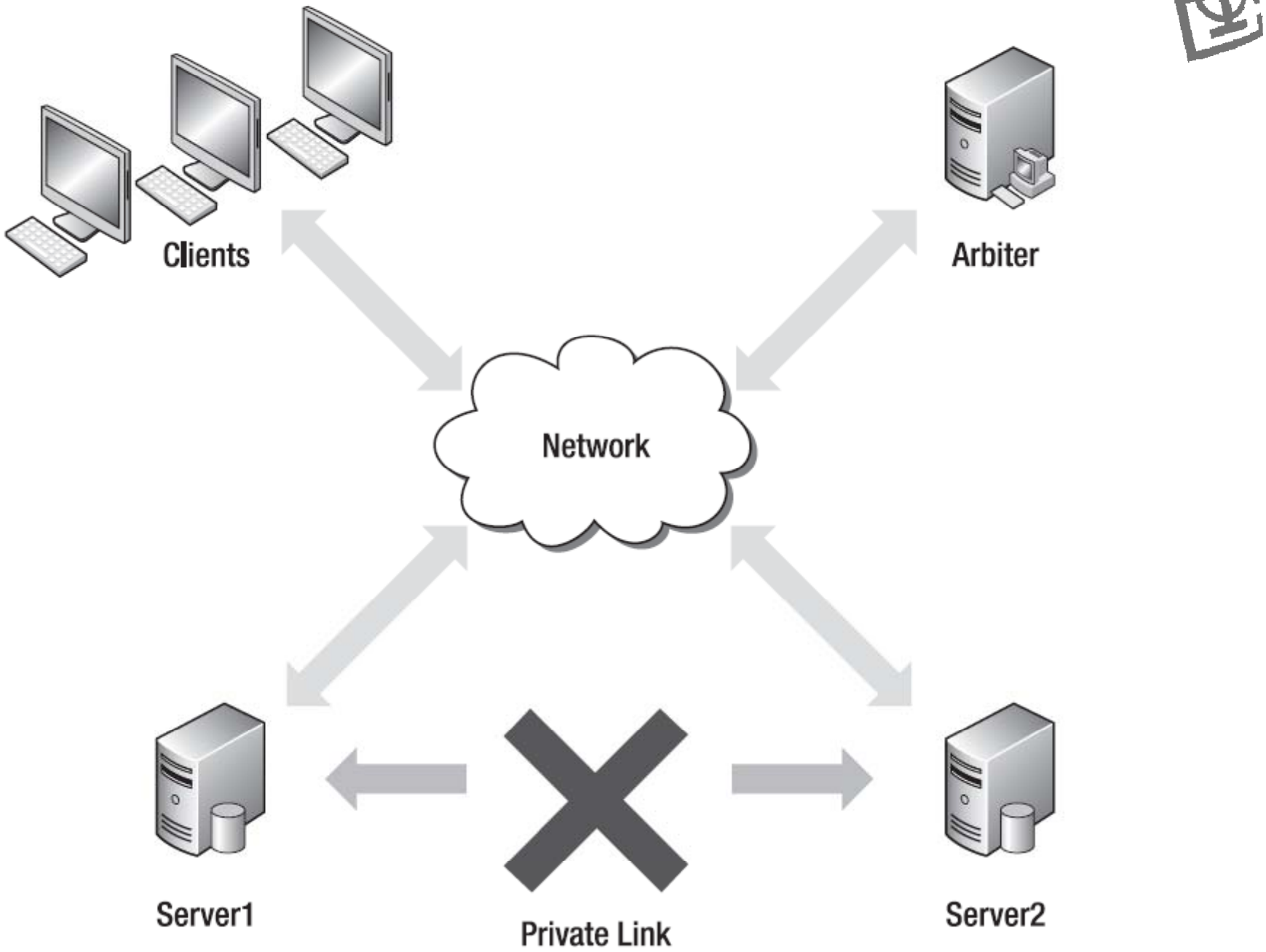

Client A
Client B
Master A
Master B
Slave
Backups
Reports

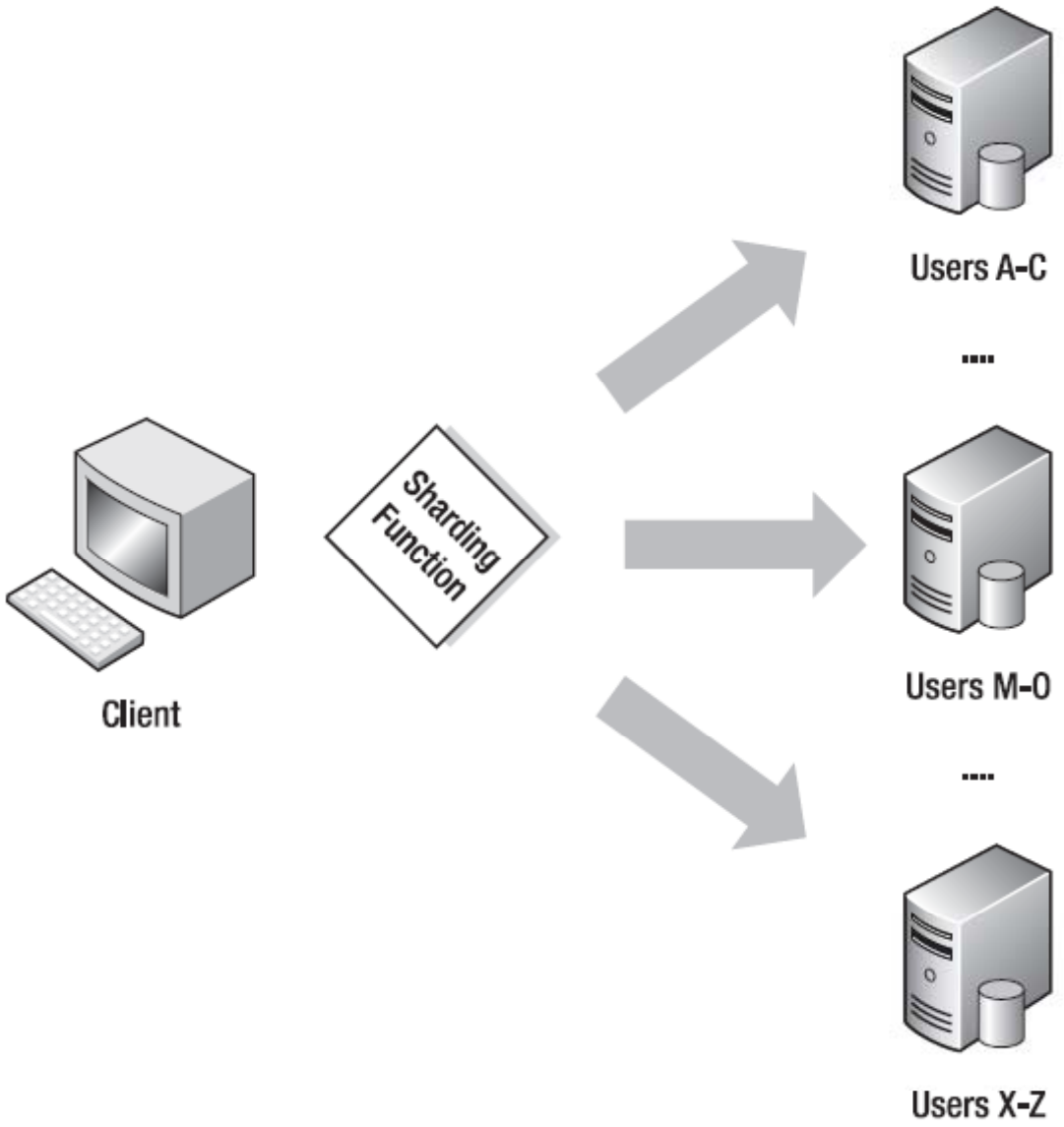

Client
Master
Slave/Master
Slave
Backups
Reports

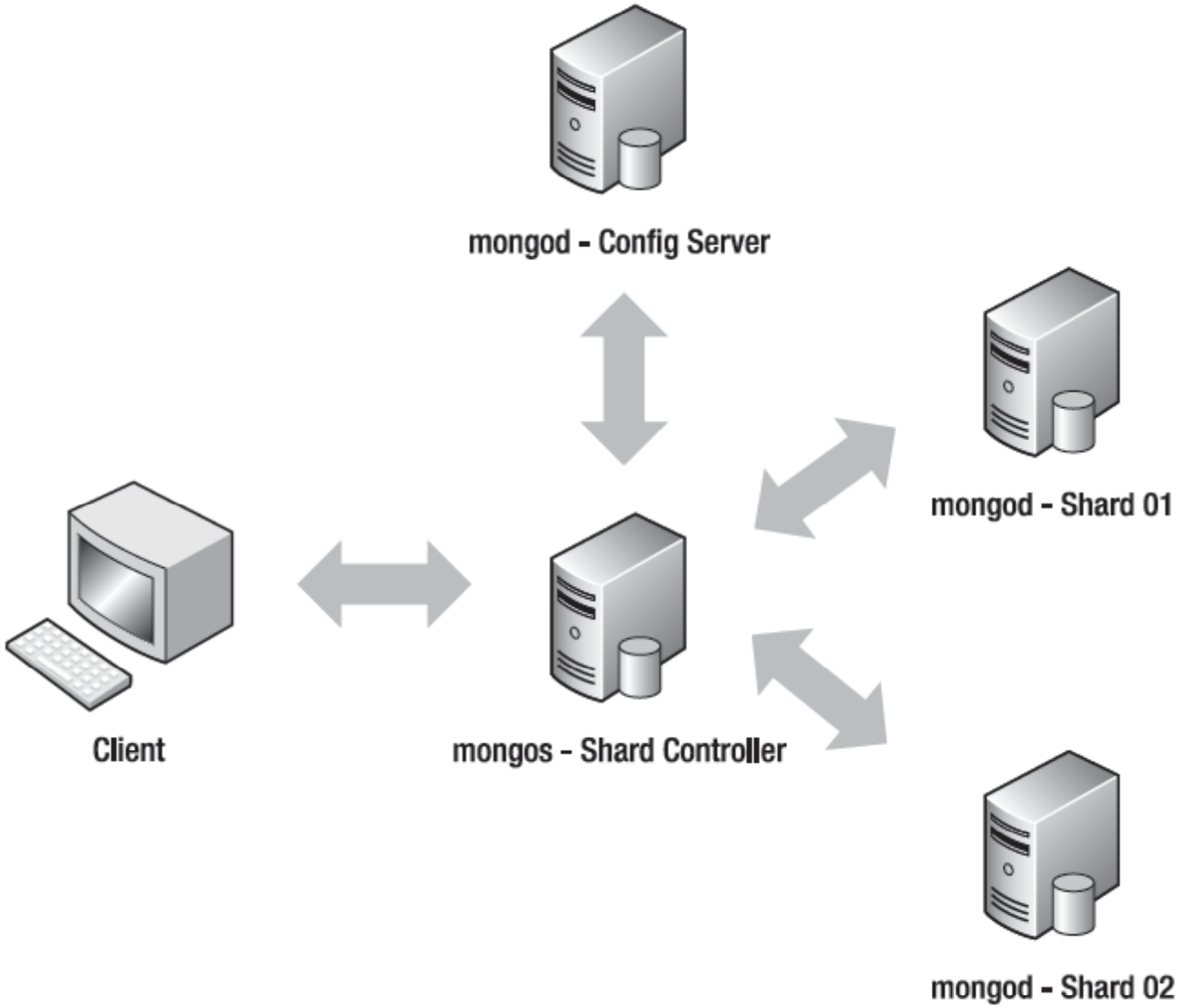

Client A
Master A
Client B
Master B

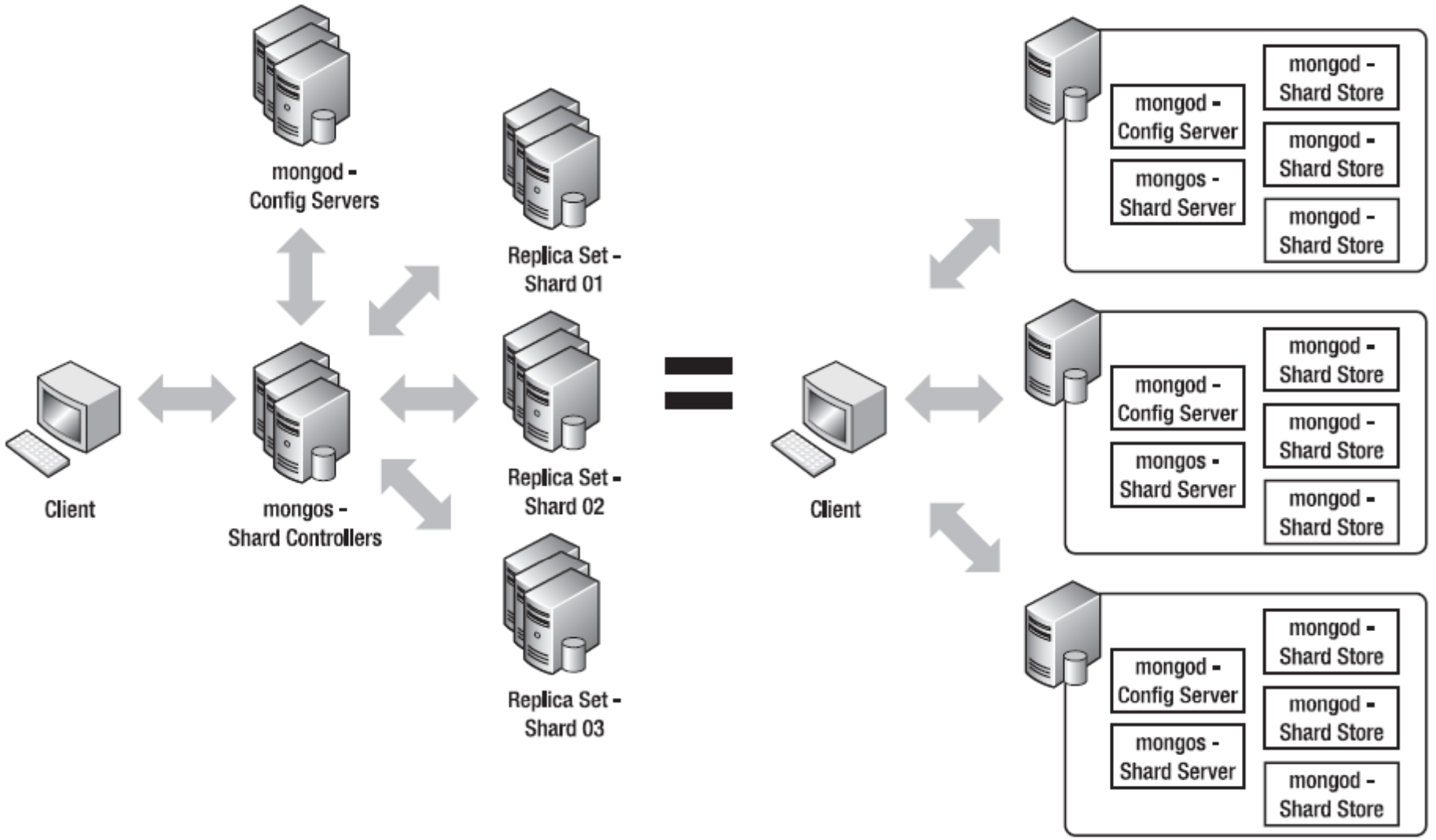

Client App A
Master App A/Slave App B
Master App B/Slave App A
Client App B


Client
Active
Primary
Secondary
Secondary
Secondary
Secondary
Passive
Secondary
Backups
Secondary
Reports


Clients
Arbiter
Network
Server1
Private Link
Server2


Users A-C
....
Sharding Function
Users M-O
....
Client
Users X-Z


mongod - Config Server
mongod - Shard 01
Client
mongos - Shard Controller
mongod - Shard 02


mongod -
Config Servers
Replica Set -
Shard 01
Client
mongos -
Shard Controllers
Replica Set -
Shard 02
Replica Set -
Shard 03
Client
mongod -
Config Server
mongos -
Shard Server
mongod -
Shard Store
mongod -
Shard Store
mongod -
Shard Store
mongod -
Config Server
mongos -
Shard Server
mongod -
Shard Store
mongod -
Shard Store
mongod -
Shard Store
mongod -
Config Server
mongos -
Shard Server
mongod -
Shard Store
mongod -
Shard Store
mongod -
Shard Store

Osnovne komande (raspakovati arhivu → kreirati C:\data\db → pokrenuti)

> use library
Switched to db library

> show dbs
admin
local

> show collections
system.indexes

```
> document = ( { "Type" : "Book", "Title" :"Definitive Guide to MongoDB, the",
"ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress",
"Author" : ["Membrey, Peter","Plugge, Eelco","Hawkins, Tim"] } )

> db.media.insert(document)

> db.media.insert( { "Type" : "CD“,  "Artist" : "Nirvana“, "Title" : "Nevermind",
"Tracklist" : [
{ "Track" : "1“, "Title" : "Smells like teen spirit“, "Length" : "5:02” },
{ "Track" : "2“, "Title" : "In Bloom“, "Length" : "4:15” }
]
} )
```

> db.media.find()

{ "_id" : "ObjectId("4c1a8a56c603000000007ecb"), "Type" : "Book", "Title" :
"Definitive Guide to MongoDB, the", "ISBN" : "987-4302-3051-9", "Publisher" :
"Apress", "Author" : ["Membrey, Peter", "Plugge, Eelco", "Hawkins, Tim"] }
{ "_id" : "ObjectId("4c1a86bb2955000000004076"), "Type" : "CD", "Artist" :
"Nirvana", "Title" : "Nevermind", "Tracklist" : [
{
"Track" : "1",
"Title" : "Smells like teen spirit",
"Length" : "5:02"
},
{
"Track" : "2",
"Title" : "In Bloom",
"Length" : "4:15"
}
] }

> db.media.find ({ Artist : "Nirvana" })

{ "_id" : "ObjectId("4c1a86bb29550000000004076"), "Type" : "CD", "Artist" :
"Nirvana", "Title" : "Nevermind", "Tracklist" : [
{
"Track" : "1",
"Title" : "Smells like teen spirit",
"Length" : "5:02"
},
{
"Track" : "2",
"Title" : "In Bloom",
"Length" : "4:15"
}
] }

> db.media.find({ "Tracklist.Title" : "In Bloom" })

{ "_id" : "ObjectId("4c1a86bb2955000000004076"), "Type" : "CD", "Artist" :
"Nirvana", "Title" : "Nevermind", "Tracklist" : [
{
"Track" : "1",
"Title" : "Smells like teen spirit",
"Length" : "5:02"
},
{
"Track" : "2",
"Title" : "In Bloom",
"Length" : "4:15"
}
] }

> db.media.find ({ "Tracklist" : {"Track" : "1" }})

→ ne vraća rezultat, jer je neophodno da se podobjekat u potpunosti poklapa

> db.media.find().sort({ Title: 1 })
→ rastuća sortiranost 1, opadajuća sortiranost -1

> db.media.find().limit(10)

> db.media.find().skip(20)

> db.media.find().sort ({ Title : -1 }).limit (10).skip (20)

> db.media.find().limit(1) → **> db.media.findOne()**

> db.media.find ({Artist : "Nirvana"}, {Title: 1})

{ "_id" : ObjectId("4c1a86bb2955000000004076"), "Title" : "Nevermind" }

Title: 0 → vraća sva polja izuzev Title

> db.createCollection("audit100", { capped:true, size:20480, max: 100})
{ "ok" : 1 }

> db.audit100.validate()
{ "ns" : "media.audit100",
"result" : "
validate
capped:1 max:100
firstExtent:0:54000 ns:media.audit100
lastExtent:0:54000 ns:media.audit100
extents:1
datasize?:0 nrecords?:0 lastExtentSize:20736
padding:1
first extent:
loc:0:54000 xnext:null xprev:null
nsdiag:media.audit100
size:20736 firstRecord:null lastRecord:null
capped outOfOrder:0 (OK)
0 objects found, nobj:0
0 bytes data w/headers
0 bytes data wout/headers
deletedList: 1100000000000000000
deleted: n: 2 size: 20560
nIndexes:0
",
"ok" : 1,
"valid" : true,
"lastExtentSize" : 20736 }

→ fiksna veličina kolekcije,
brišu se najstariji dokumenti automatski ,
dokument ne može da menja veličinu,
dokument ne može biti obrisan

> db.audit.find().sort({ $natural: -1 }).limit (10)
→ vraća poslednjih 10 dodatih zapisa

> db.media.count()
2

> db.media.find({ Publisher : "Apress", Type: "Book" }).count()
1

> db.media.find({ Publisher: "Apress", Type: "Book" }).skip (2) .count (true)
0
→ count (true), u suprotnom count ignoriše skip i limit

> db.media.distinct("Title")
["Definitive Guide to MongoDB, the", "Nevermind"]

> db.media.group (
{
key: {Title : true},
initial: {Total : 0},
reduce : function (items,prev) { prev.Total += 1}
}
)

→ key – funkcija koja obavlja grupisanje
cond – dodatni uslov za svaki dokument
finalize – funkcija pre prikaza rezltata

[{ "Title" : "Nevermind“, "Total" : 1},
{"Title" : "Definitive Guide to MongoDB, the", "Total" : 2 }]

> dvd = ({ "Type" : "DVD", "Title" : "Matrix, The", "Released" : 1999,
"Cast" : ["Keanu Reeves","Carry-Anne Moss","Laurence Fishburne","Hugo
Weaving","Gloria Foster","Joe Pantoliano"] })
> db.media.insert(dvd)
> dvd = ({ "Type" : "DVD", "Title" : "Blade Runner", "Released" : 1982 })
> db.media.insert(dvd)
> dvd = ({ "Type" : "DVD", "Title" : "Toy Story 3", "Released" : 2010 })
> db.media.insert(dvd)

> db.media.find ({ Released : {$gte : 1999 } }, { "Cast" : 0 })
{ "_id" : ObjectId("4c43694bc603000000007ed1"), "Type" : "DVD",
"Title" : "Matrix, The", "Released" : 1999 }
{ "_id" : ObjectId("4c4369a3c603000000007ed3"), "Type" : "DVD",
"Title" : "Toy Story 3", "Released" : 2010 }

> db.media.find({Released : {$gte: 1990, $lt : 2010}}, { "Cast" : 0 })
{ "_id" : ObjectId("4c43694bc603000000007ed1"), "Type" : "DVD",
"Title" : "Matrix, The", "Released" : 1999 }

> db.media.find({ Type : "Book", Author: {$ne : "Plugge, Eelco"})

> db.media.find({Released : {$in : ["1999","2008","2009"] } }, { "Cast" : 0 })
{ "_id" : ObjectId("4c43694bc603000000007ed1"), "Type" : "DVD",
"Title" : "Matrix, The", "Released" : 1999 }

> db.media.find({Released : {$nin : ["1999","2008","2009"] },Type : "DVD" }, { "Cast" : 0 })

{ "_id" : ObjectId("4c436969c603000000007ed2"), "Type" : "DVD", "Title" : "Blade Runner", "Released" : 1982 }
{ "_id" : ObjectId("4c4369a3c603000000007ed3"), "Type" : "DVD", "Title" : "Toy Story 3", "Released" : 2010 }

> db.media.find ({ Released : {$all : ["2010","2009"] } }, { "Cast" : 0 })

> db.media.find({ "Type" : "DVD", $or : [{ "Title" : "Toy Story 3" }, { "ISBN" : "987-1-4302-3051-9" }] })

{ "_id" : ObjectId("4c5fc943db290000000067ca"), "Type" : "DVD", "Title" : "Toy Story 3", "Released" : 2010 }

> db.media.find ({ Released : { $mod: [2,0] } }, {"Cast" : 0 })

> db.media.find({"Title" : "Matrix, The"}, {"Cast" : {$slice: 3}})

{ "_id" : ObjectId("4c5fcd3edb290000000067cb"), "Type" : "DVD", "Title" : "Matrix, The", "Released" : 1999, "Cast" : ["Keanu Reeves", "Carry-Anne Moss", "Laurence Fishburne"] }

→ skip i limit ne rade nad nizovima → npr. $slice: [-5,4] skoči na poslednjih 5, limitira na 4

> db.media.find ({ Tracklist : {$size : 2} })
→ samo oni dokumenti sa tačnom dužinom niza

> db.media.find ({ Author : {$exists : true } })
→ samo ukoliko polje Author postoji

> db.media.find ({ Tracklist: { $type : 3 } })
→ samo ukoliko je polje Tracklist odgovarajućeg tipa
-1 MiniKey, 1 Double, 2 Character string (UTF8), 3 Embedded object, 4 Embedded array, 5 Binary Data, 7 Object ID, 8 Boolean type,
9 Date type, 10 Null type, 11 Regular Expression, 13 JavaScript Code, 14 Symbol, 15 JavaScript Code with scope,
16 32-bit integer, 17 Timestamp, 18 64-bit integer, 127 MaxKey, 255 Min Key

> db.media.find ({ Tracklist: { "$elemMatch" : { Title:
"Smells like teen spirit", Track : "1" } } })
→ pronalazi samo ako upari čitav dokument unutar niza
→ bitno se razlikuje od
> db.media.find ({ "Tracklist.Title" : "Smells like teen spirit", "Tracklist.Track" : "1" })

> db.media.update({ "Title" : "Matrix, the"}, {"Type" : "DVD", "Title" : "Matrix, the", "Released" : "1999", "Genre" : "Action"}, true)
→ ukoliko je true, znači radi upsert (update, a ako ne postoji onda insert)

> db.media.save({ "Title" : "Matrix, the"}, {"Type" : "DVD", "Title" : "Matrix, the", "Released" : "1999", "Genre" : "Action"})

> manga = ({ "Type" : "Manga", "Title" : "One Piece", "Volumes" : 612, "Read" : 520 })
> db.media.insert(manga)
> db.media.update ({ "Title" : "One Piece"}, {$inc: {"Read" : 4} })

> db.media.update ({ "Title" : "Matrix, the" }, {$set : { Genre : "Sci-Fi" } }) → postavlja
> db.media.update ({"Title": "Matrix, the"}, {$unset : { "Genre" : 1 } }) → briše
→ za niz:
> db.media.update ({"ISBN" : "1-4302-3051-7"}, {$push: { Author : "Griffin, Stewie"} })
> db.media.update ({"ISBN" : "1-4302-3051-7"}, {$pull : { Author : "Griffin, Stewie" } })

> db.media.update({ "ISBN" : "1-4302-3051-7" }, {$pop : {Author : 1 } })
→ vrednost 1 znači poslednjeg dodatog, vrednost -1 znači prvog dodatog

> db.media.update({"ISBN" : "1-4302-3051-7"},{$pushAll: {Author : ["Griffin, Louis","Griffin, Peter"] } })
> db.media.update({ "ISBN" : "1-4302-3051-7" }, {$addToSet : { Author :"Griffin, Brian" } })

$set, $unset, $inc, $push, $pushAll, $pull, $pullAll → atomične operacije

→ update if current:

> db.media.update({ "Tracklist.Title" : "Been a son"},
{$inc:{"Tracklist.$.Track" : 1} })
> db.$cmd.findOne({getlasterror:1})
{ "err" : null, "updatedExisting" : true, "n" : 1, "ok" : 1 }

→ modify and return:

> db.media.findAndModify({ query: { "ISBN" : "987-1-4302-3051-9" }, sort: {"Title":-1}, update: {$set: {"Title" : " Different Title"} } })

> db.media.findAndModify({ query: { "ISBN" : "987-1-4302-3051-9" }, sort: {"Title":-1}, update: {$set: {"Title" : " Different Title"} }, new:true })
→ vraća novu, promenjenu, vrednost

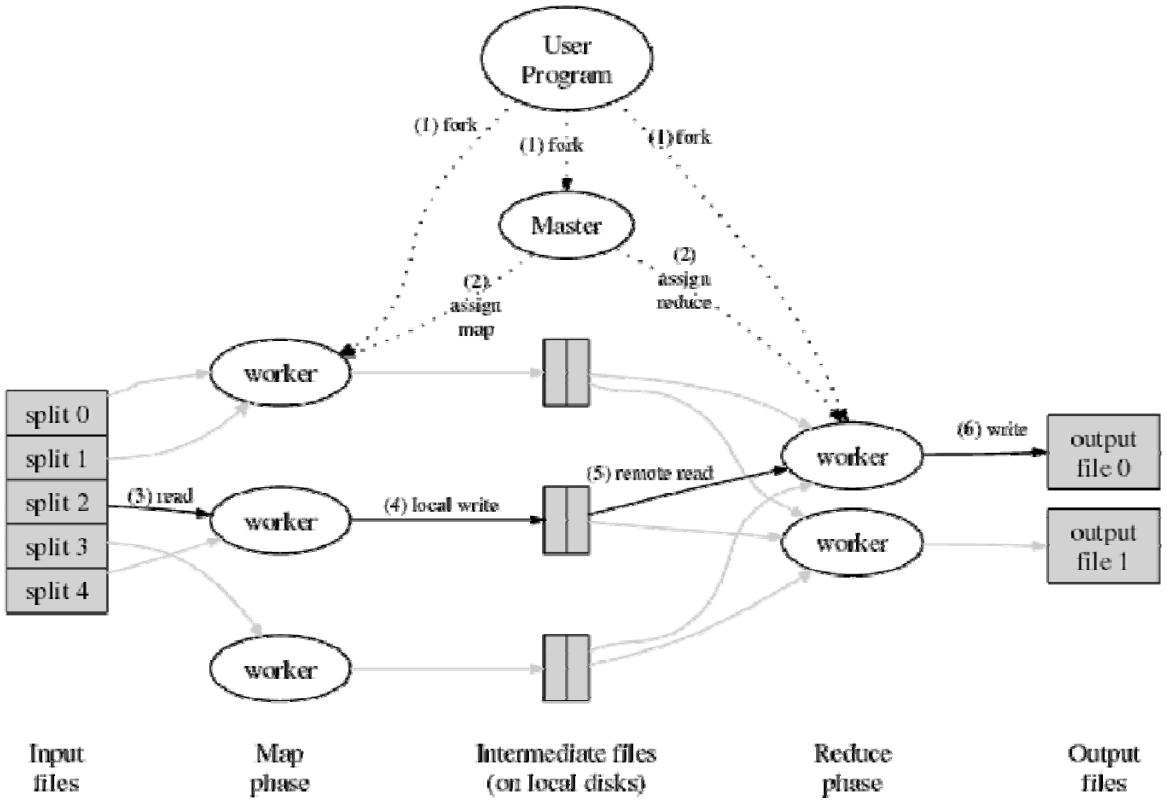
→ manuelno referisanje:

```
> apress = ( { "_id" : "Apress", "Type" : "Technical Publisher", "Category" :
["IT", "Software","Programming"] } )
> db.publisherscollection.insert(apress)

> book = ( { "Type" : "Book", "Title" : "Definitive Guide to MongoDB, the",
"ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress","Author" : ["Membrey,
Peter","Plugge, Eelco","Hawkins, Tim"] } )
> db.media.insert(book)

> book = db.media.findOne();
db.publisherscollection.findOne( { _id : book.Publisher } )
```

→ upotreba DBRef (otporno na promenu naziva):

{ $ref : <collectionname>, $id : <id value>[, $db : <database name>] }

```
> apress = ({"Type" :"Technical Publisher", "Category":["IT","Software","Programming"] } )
> db.publisherscollection.save(apress)
> book = { "Type" : "Book", "Title" : "Definitive Guide to MongoDB, the",
"ISBN":"987-1-4302-3051-9","Author": ["Membrey, Peter","Plugge, Eelco","Hawkins, Tim"],
Publisher : [ new DBRef ('publisherscollection‘, apress._id) ] }
> db.media.save(book)
```



Invertovano indeksiranje
- map: <word, docID>
- reduce: <word, list(docID)>

Reverzni graf Web linkova
- map: <dstID, srcID>
- reduce: <dstID, list(srcID)>

Prebrojavanje pristupa određenom URL
- map: <URL, 1>
- reduce: <URL, totalCount>

```
db.runCommand ({
    mapReduce: <collection>,
    map: <function>,
    reduce: <function>,
    out: <output>,
    query: <document>,
    sort: <document>,
    limit: <number>,
    finalize: <function>,
    scope: <document>,
    jsMode: <boolean>,
    verbose: <boolean> });
```

Osobine Map funkcije:

```
function() {
   ...
   emit(key, value);
}
```

- Referencira trenutni dokument koristeći **this**
- Ne sme **pristupati** bazi podataka
- Ne sme imati **bočne efekte**
- Mora da pozove **emit(key,value)** funkciju
- Jedan **emit** ima ograničenje veličine (npr. 50% max document ~ 8MB)
- Nije ograničen **broj poziva** emit po dokumentu
- Može da pristupa promenljivama definisanim unutar **scope** parametra

Osobine Reduce funkcije:

```
function(key, values) {
   ...
   return result;
}
```

- Ne sme **pristupati** bazi podataka
- Ne sme imati **bočne efekte**
- **Ne poziva se** za one ključeve koji imaju samo jednu vrednost
- Može da pristupa promenljivama definisanim unutar **scope** parametra
- Tip povratne vrednosti mora biti **identičan** tipu koji ima **value** u **map**
- Funkcija mora biti **idempotentna**
- Rezultat funkcije ne sme zavisiti od redosleda unutar **values** niza

reduce(key, [C, reduce(key, [A, B])]) == reduce(key, [C, A, B])

reduce(key, [reduce(key, valuesArray)]) == reduce(key, valuesArray)

→ map/reduce (M/R) posao koji vraća ukupnu cenu svih narudžbina svakog od kupaca:

```
{
_id: ObjectId("50a8240b927d5d8b5891743c"),
cust_id: "abc123",
ord_date: new Date("Oct 04, 2012"),
status: 'A',
price: 250,
items: [ { sku: "mmm", qty: 5, price: 2.5 }, { sku: "nnn", qty: 5, price: 2.5 } ]
}
```

```
var mapFunction1 = function() {
    emit(this.cust_id, this.price);
};
var reduceFunction1 = function(keyCustId, valuesPrices) {
    return Array.sum(valuesPrices);
};

db.orders.mapReduce( mapFunction1, reduceFunction1, { out: "map_reduce_example" });
```

{ out: replace "result_collection"} – menja postojeću kolekciju ukoliko postoji
{ out: merge "result_collection"} – dopunjuje postojeću, a istovetne ključeve zamenjuje
{ out: reduce "result_collection"} – dopunjuje postojeću, a istovetne ključeve redukuje

→ M/R posao koji za svaku naručenu stavku vraća prosečnu količinu po narudžbini:

```
var mapFunc2 = function() {
    for (var idx = 0; idx < this.items.length; idx++) {
        var key = this.items[idx].sku;
        var value = { count: 1,
                      qty: this.items[idx].qty
                    };
        emit(key, value);
    }
};
```

```
var reduceFunc2 = function(keySKU, countObjVals) {
    reducedVal = { count: 0, qty: 0 };
    for (var idx = 0; idx < countObjVals.length; idx++) {
        reducedVal.count += countObjVals[idx].count;
        reducedVal.qty += countObjVals[idx].qty;
    }
    return reducedVal;
};
```

```
var finalizeFunc2 = function (key, reducedVal) {
    reducedVal.avg = reducedVal.qty/reducedVal.count;
    return reducedVal;
};
```

```
db.orders.mapReduce( mapFunc2, reduceFunc2,
{ out: merge: "map_reduce_example", finalize: finalizeFunc2});
```

→ inkrementalni M/R posao koji na osnovu kolekcije sa sesijama vraća za svakog korisnika, ukupan broj sesija tog korisnika, kao i ukupno trajanje i prosečno trajanje sesija tog korisnika:

```
{ _id: ObjectId("50a8240b927d5d8b5891743c"),
  userid: "a",
  ts: ISODate('2011-11-03 14:17:00'),
  length: 95
}
```

```
var mapFunction = function() {
    var key = this.userid;
    var value = { userid: this.userid,
                  total_time: this.length,
                  count: 1,
                  avg_time: 0
                 };
     emit( key, value );
};
```

```
var reduceFunction = function(key, values) {
    var reducedObject = { userid: key,
                          total_time: 0,
                          count:0,
                          avg_time:0
                         };
    values.forEach( function(value) {
        reducedObject.total_time += value.total_time;
        reducedObject.count += value.count;
     });
     return reducedObject;
};
```

```
var finalizeFunction = function (key, reducedValue) {
    if (reducedValue.count > 0)
        reducedValue.avg_time = reducedValue.total_time / reducedValue.count;
    return reducedValue;
};
```

Početno stanje:

```
db.sessions.save( { userid: "a", ts: ISODate('2011-11-03 14:17:00'), length: 95 } );
db.sessions.save( { userid: "b", ts: ISODate('2011-11-03 14:23:00'), length: 110 } );
db.sessions.save( { userid: "c", ts: ISODate('2011-11-03 15:02:00'), length: 120 } );
db.sessions.save( { userid: "d", ts: ISODate('2011-11-03 16:45:00'), length: 45 } );
db.sessions.save( { userid: "a", ts: ISODate('2011-11-04 11:05:00'), length: 105 } );
db.sessions.save( { userid: "b", ts: ISODate('2011-11-04 13:14:00'), length: 120 } );
db.sessions.save( { userid: "c", ts: ISODate('2011-11-04 17:00:00'), length: 130 } );
db.sessions.save( { userid: "d", ts: ISODate('2011-11-04 15:37:00'), length: 65 } );
```

```
db.sessions.mapReduce( mapFunction, reduceFunction,
{ out: { reduce: "session_stat" }, finalize: finalizeFunction }
);
```

Dopunjeno stanje:

```
db.sessions.save( { userid: "a", ts: ISODate('2011-11-05 14:17:00'), length: 100 } );
db.sessions.save( { userid: "b", ts: ISODate('2011-11-05 14:23:00'), length: 115 } );
db.sessions.save( { userid: "c", ts: ISODate('2011-11-05 15:02:00'), length: 125 } );
db.sessions.save( { userid: "d", ts: ISODate('2011-11-05 16:45:00'), length: 55 } );
```

```
db.sessions.mapReduce( mapFunction, reduceFunction,
{ query: { ts: { $gt: ISODate('2011-11-05 00:00:00') } },
out: { reduce: "session_stat" }, finalize: finalizeFunction }
);
```

Osobine Agregate funkcije:

- Zamenjuje Map/Reduce u slučaju jednostavnijih poslova
- Deklarativnog tipa → JavaScript nije potreban
- Definiše lanac operacija koje je potrebno izvršiti (**pipeline**)
- Izračunavanje iskaza → vraća izračunate vrednosti
- Brža implementacija (C++ umesto JavaScript)
- Planirano dodavanje novih operacija

$project [oblikuje rezultat]
- uključuje i isključuje određena polja iz rezultata
- izračunava iskaze (poziva ugrađene funkcije, čita/upisuje ugnježđeni dokument)

$unwind [razmotava nizove]
- jedan po jedan član niza uparuje sa ostatkom okolnog dokumenta

$sort [sortira dokumente] – npr. $sort: {key1: 1, key2: -1, ...}

$match [filtrira na osnovu predikata] – isti format kao kod **find()** metode

$group [grupiše]
- definiše ključ **_id** po kome se radi grupisanje
- agregira grupisane vrednosti **$sum**, **$avg**, **$min**, **$max**
- formira niz ili skup na osnovu grupisanih vrednosti **$push**, **$addToSet**
- dodatne funkcije **$first**, **$last** – imaju smisla samo nakon sortiranja

$limit [ograničava broj dokumenata]

$skip [preskače određen broj dokumenata]

$geoNear [sortira dokumenta na osnovu geografske udaljenosti] – mora biti prvi u lancu

kolekcija **zipcodes** sa informacijama o gradovima:

```
{
    "_id": "10280",
    "city": "NEW YORK",
    "state": "NY",
    "pop": 5574,
    "loc": [ -74.016323, 40.710537 ]
}
```

→ skript koji vraća sve države sa više od 10 miliona stanovnika

```
db.zipcodes.aggregate( { $group :
                                { _id : "$state",
                                  totalPop : { $sum : "$pop" }
                                 }
                        },
                        { $match : {totalPop : { $gte : 10*1000*1000 } } } )
```

npr. izgleda rezultata:

```
{
"_id" : "AK",
"totalPop" : 550043
}
```

```
SELECT state, SUM(pop) AS totalpop
FROM zipstate
GROUP BY state
HAVING SUM(pop) > (10*1000*1000)
```

→ skript koji vraća prosečan broj stanovnika u gradovima na nivou svake države

```
db.zipcodes.aggregate( { $group :
                            { _id : { state : "$state", city : "$city" },
                              pop : { $sum : "$pop" }
                             }
                       },
                       { $group :
                            { _id : "$_id.state",
                              avgCityPop : { $avg : "$pop" }
                             }
                        } )
```

npr. izgleda rezultata nakon prvog grupisanja:

```
{
  "_id" : {
            "state" : "CO",
            "city" : "EDGEWATER"
           },
  "pop" : 13154
}
```

npr. izgleda rezultata nakon drugog grupisanja:

```
{
  "_id" : "MN",
  "avgCityPop" : 5335
}
```

→ skript koji na nivou svake od država vraća gradove koji imaju najveći i najmanji broj stanovnika

```
db.zipcodes.aggregate( { $group:
                                { _id: { state: "$state", city: "$city" },
                                  pop: { $sum: "$pop" } } },
                       { $sort: { pop: 1 } },
                       { $group:
                                { _id : "$_id.state",
                                  biggestCity: { $last: "$_id.city" },
                                  biggestPop: { $last: "$pop" },
                                  smallestCity: { $first: "$_id.city" },
                                  smallestPop: { $first: "$pop" } } },
// $project je opcioni
// modifikuje izlazni format
                        { $project:
                              { _id: 0,
                                state: "$_id",
                                biggestCity: { name: "$biggestCity", pop: "$biggestPop" },
                                smallestCity: { name: "$smallestCity", pop: "$smallestPop" }
                               }
                          }
                       )
```

kolekcija **users** sa informacijama o članovima sportskog udruženja:

{ _id : "jane", joined : ISODate("2011-03-02"), likes : ["golf", "racquetball"] }
{ _id : "joe", joined : ISODate("2012-07-02"), likes : ["tennis", "golf", "swimming"] }

→ skript koji vraća imena članova napisana velikim slovima, sortiranim po alfabetu

```
db.users.aggregate( [
                      { $project : { name:{$toUpper:"$_id"} , _id:0 } },
                      { $sort : { name : 1 } }
                    ]
)
```

→ skript koji za svakog člana vraća mesec učlanjenja i ime, sortiranim po mesecu

```
db.users.aggregate([
                    { $project : { month_joined : { $month : "$joined"},
                                   name : "$_id",
                                   _id : 0
                     },
                    { $sort : { month_joined : 1 } }
                   ]
)
```

→ skript koji vraća informaciju o tome koliko je novih članova bilo po mesecima

```
db.users.aggregate([
                    { $project : { month_joined : { $month : "$joined" } } } ,
                    { $group :
                                { _id : {month_joined:"$month_joined"} ,
                                  number : { $sum : 1 }
                                 }
                     },
                     { $sort : { "_id.month_joined" : 1 } }
                    ]
                   )
```

→ skript koji vraća pet najčešće omiljenih sportova

```
db.users.aggregate([
                    { $unwind : "$likes" },
                    { $group :
                                { _id : "$likes" ,
                                  number : { $sum : 1 }
                                 }
                     },
                     { $sort : { number : -1 } },
                     { $limit : 5 }
                    ]
                   )
```