NoSql, Document Store, MongoDB

Gombos Gergő

Tematika

- NoSql
  - Elosztott rendszerek konzisztenciája
  - CAP, ACID, BASE
- MongoDB
  - Koncepció
  - JSON, BSON
  - Adattárolás
  - Replica, sharding
  - Mongo CRUD
  - Aggregation framework

Elosztott rendszerek konzisztenciája

- Az adatokat sok gépen tároljuk
  - Scale-out (~2000-)
- A gépek meghibásodhatnak
- Az adatokhoz több belépési ponton keresztül is hozzáférhetünk
  - Ne legyen bottle-neck a hozzáférés
- Biztosítani kell:
  - Konzisztenciát
  - Rendelkezésre állást
  - Partíció tűrést

Konzisztencia (C)

- Biztonsági mentés helyett replikáció
  - Az adatok több példányban tárolódnak
- Mi biztosítja, hogy a replikák konzisztensek maradnak?
  - Kell valami replikaciós protokoll
  - Általában aszinkron
- Konzisztencia ablak
  - Mennyi idő után válik a rendszer konzisztenssé

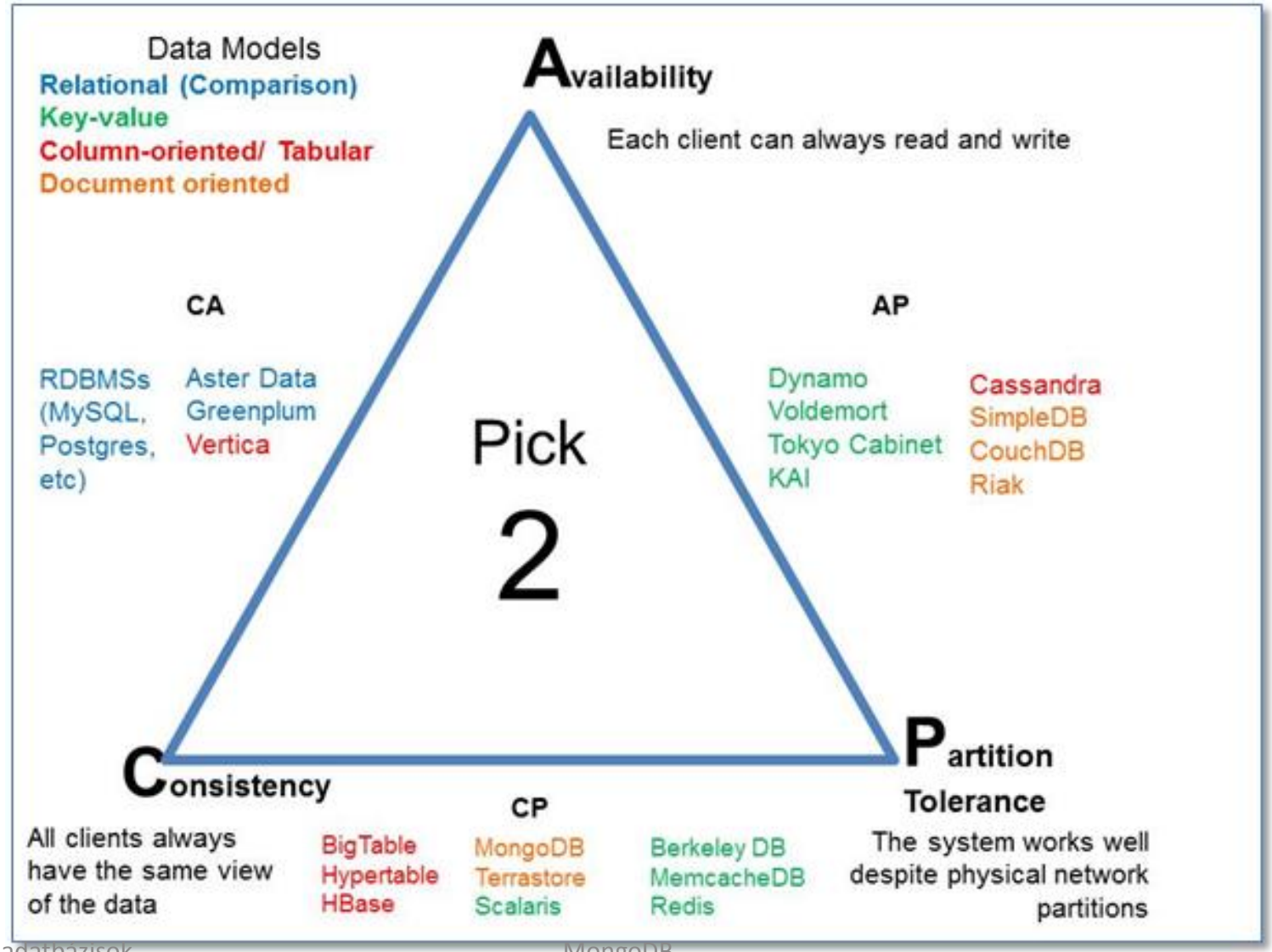
Rendelkezésre állás (A)

- Az adatok folyton elérhetőek
  - Több belépési pont
  - Nincsen egyetlen kritikus elem sem
  - Geo-redundancia
- A válasz legyen gyors
  - Minimális késleltetés
  - Meg akkor is, ha a visszaadott adat nem konzisztens (pl. Facebook idővonal)

Partíció tűrés (P)

- Elosztott rendszer
  - Hálózati kapcsolat (lassú, törékeny)
- Több belépési pont
  - A rendszer akkor is működőképes marad, ha egyes részei nem látják egymást
- Elosztott funkcionalitás
  - Pl. Facebook felhasználó bejelentkezése, fénykép megosztása, levelesláda

CAP tétel


Data Models
Relational (Comparison)
Key-value
Column-oriented/ Tabular
Document oriented
Availability
Each client can always read and write
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
Pick 2
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Consistency
All clients always have the same view of the data
CP
BigTable
Hypertable
HBase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance
The system works well despite physical network partitions

ACID tranzakciós modell

- Atomicity
  - Egy tranzakció vagy teljesen lefut, vagy nem fut le
- Consistency
  - Egy tranzakció hatására az adatbázis konzisztens állapotból konzisztens állapotba jut
- Isolation
  - A tranzakciók konkurens végrehajtásának meg kell egyezni valamilyen szekvenciális végrehajtás eredményével
- Durability
  - Ha egy tranzakció sikeresre volt jelentve, akkor a változásnak tartósnak kell lennie

BASE tranzakciós modell

- BA: Basically available
  - Főleg CP rendszer esetében
  - Legalább a rendszer egy része maradjon elérhető
- Soft-state
  - A változások véges ideig tartó üzenetekkel történnek
  - A rendszer állapota akkor is változhat, ha épp nincsen input
  - Minden adatra lehet egy érvényességi idő
  - Ha ez lejárt, akkor meg kell vizsgálni, hogy konzisztens-e még
- Eventually consistent
  - Főleg AP rendszer esetében
  - A változások aszinkron propagálnak (gossip protokollok)
  - A változások „Egy idő után” konzisztenssé válik
  - Konzisztencia ablak

Inkonzisztencia feloldása

- A hibák miatt inkonzisztencia léphet fel
  - Fel kell oldani: olvasáskor, íráskor, aszinkron
- Többségi szavazáson múló algoritmusok
  - A replikákat tartalmazó node-ok szavaznak, quorumok
- Időbélyegzőn alapuló megoldások
  - Mindig a legfrissebb adatot tekintjük

MongoDB

MongoDB=„document oriented”

- A dokumentumok tárolása nyelv független BSON formátumban.

```
{
        x : 3,
        y : “abc”,
        z : [1,2],
        d : { }
}
```

JSON

- strukturált dokumentumok tárolására szolgál
- hasonló dokumentum formátum pl. XML
- Adattípusok:
  - string
  - number
  - boolean
  - NULL
  - array
  - object/document
- asszociatív tömb,
  de csak string lehet a kulcs

```
{
  “name” : “jill”,
  “age” : 2,
  “voted”: true,
  “school”: null,
  “likes”: [“tennis”, “math”],
  “addr”: {
     “city”: “New York”,
     “addr”: “3rd Street”
   }
}
```

XML vs JSON

```
<phones>
        <phone type =“home”>123</phone>
        <phone type =“mobile”>456</phone>
</phones>
```

```
{
    “phones”: [
        {“phone”: 123, “type”: “home”},
        {“phone”: 456, “type”: “mobil”}
    ]
}
```

BSON

- JSON bináris reprezentációja
- Könnyen skálázható
- Mongoimport BSON műveletekhez
- JSON:

```
{
   „a”: 3,
   „b”:”xyz”
}
```

- BSON:

Dok.hossz típus(int) kulcs érték típus(string) kulcs érték dok. vege

Schemaless

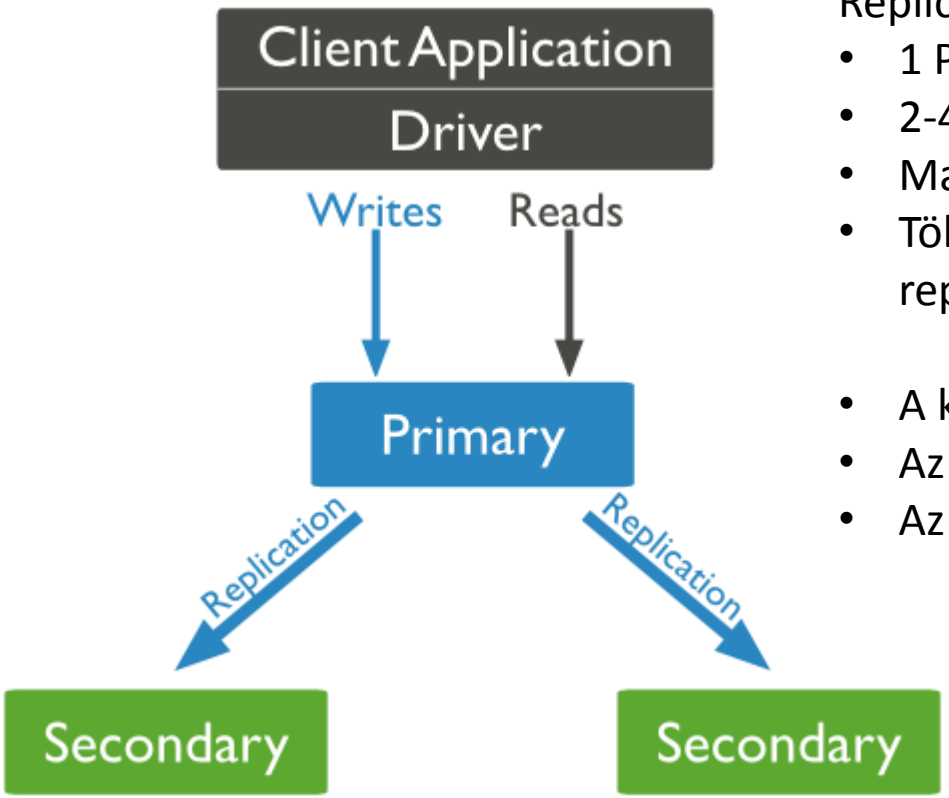
- Ez lehet egy collection:
  - {alakzat: “négyszög”, x: 3, y:4, “terület”: 12}
    {alakzat: “kör”, r: 1, “terület”: 3.14}
    {q: 456}

- Gyakorlatban konzisztens struktúra van!
  - { name : “Joe”, age : 30, interests : ‘football’ }
    { name : “Kate”, age : 25 }

MongoDB elemek

RDBMS	MongoDB
Database	Database
Table	Collection
Row	Document (JSON,BSON)
Column	Field
Index	Index
Join	Embedded Document
Foreign Key	Reference
Queries return records	Queries return a cursor

MongoDB high availability

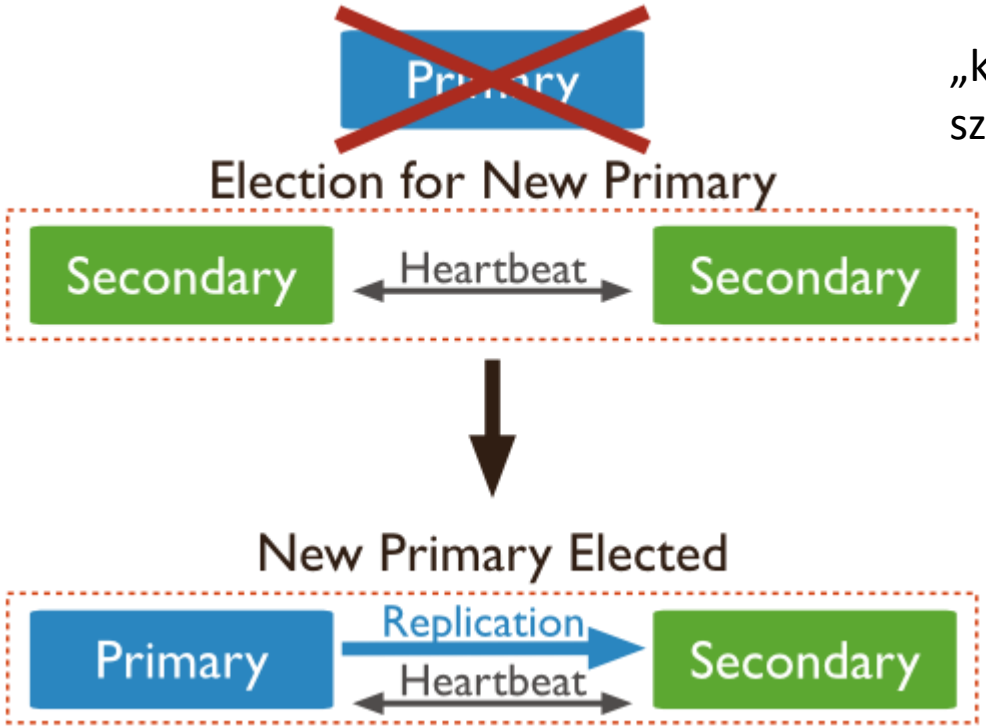


Replica Set

- 1 Primary
- 2-48 Secondaries
- Másolatok az eredeti adatról
- Több sort érintő műveletek soronként replikálódnak

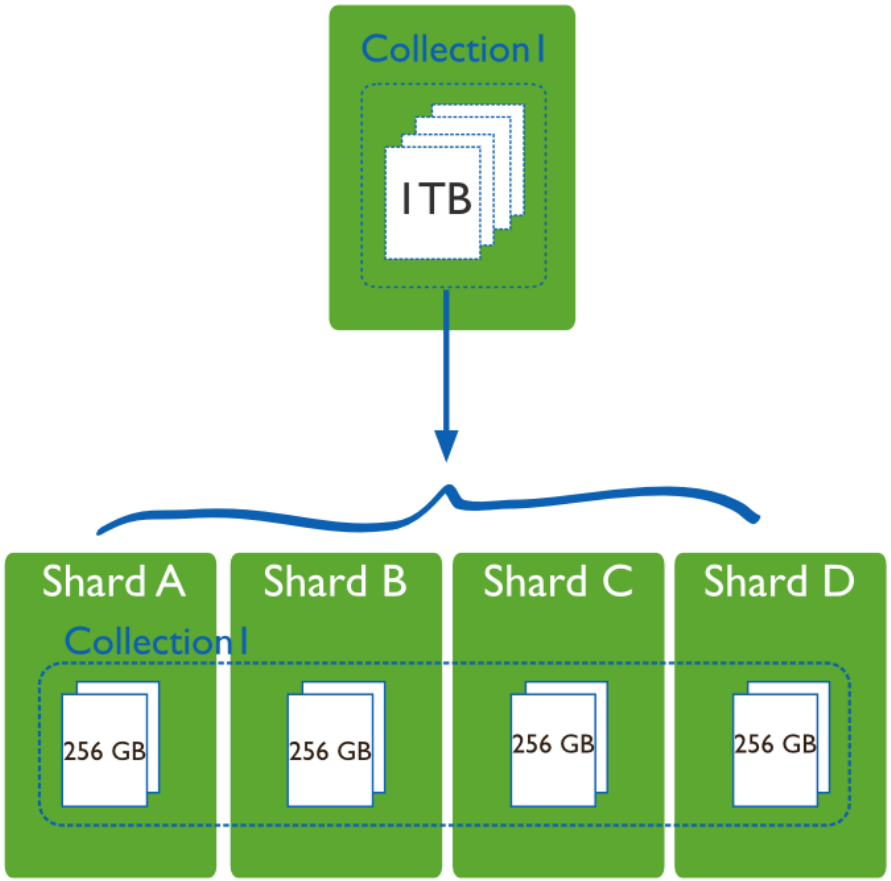
- A kliens mindig a primary-ra kapcsolódik
- Az írás mindig a primary-re megy
- Az olvasás mehet a secondary-ra

MongoDB high availability



„konszenzus”: primary hiba esetén szavazás az új primaryról

MongoDB high availability

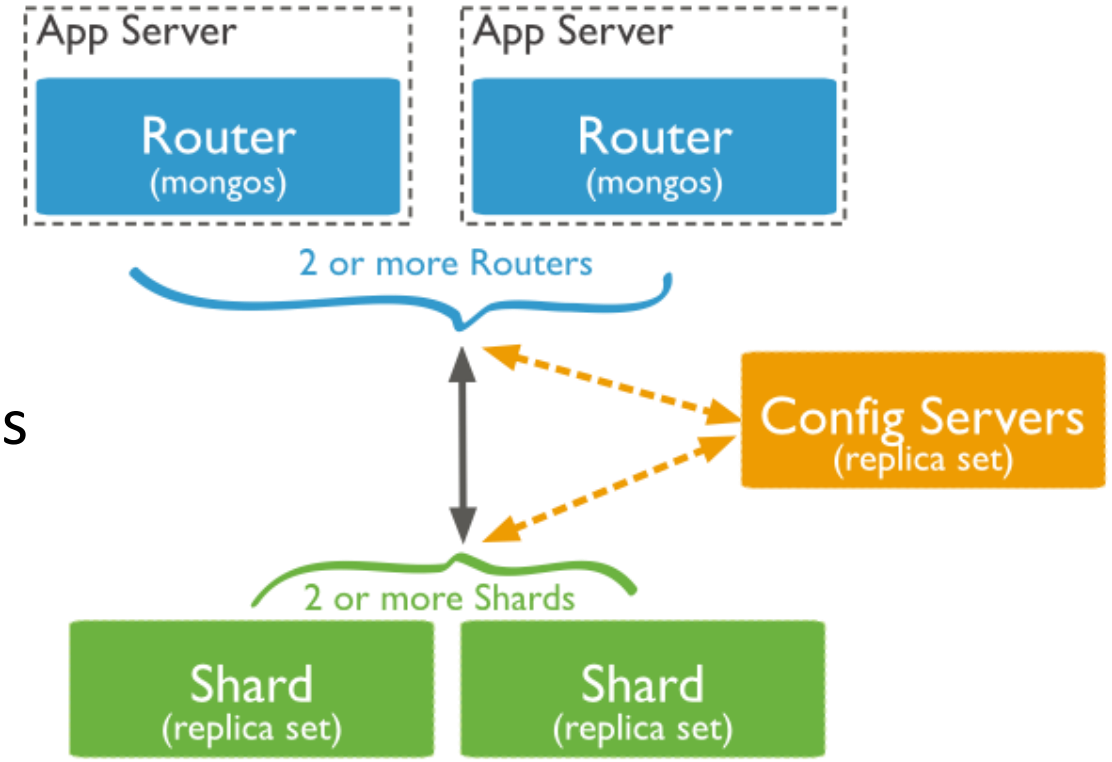


Sharding

- a collection-ök elosztva klaszterben shard key alapján
- shard key = indexelt adattag
- shardon belül lehetnek replikák
- csökkenti az egyes szerverek terhelést
  - lekérdezések több gépen
  - kevesebb adat az egyes gépeken

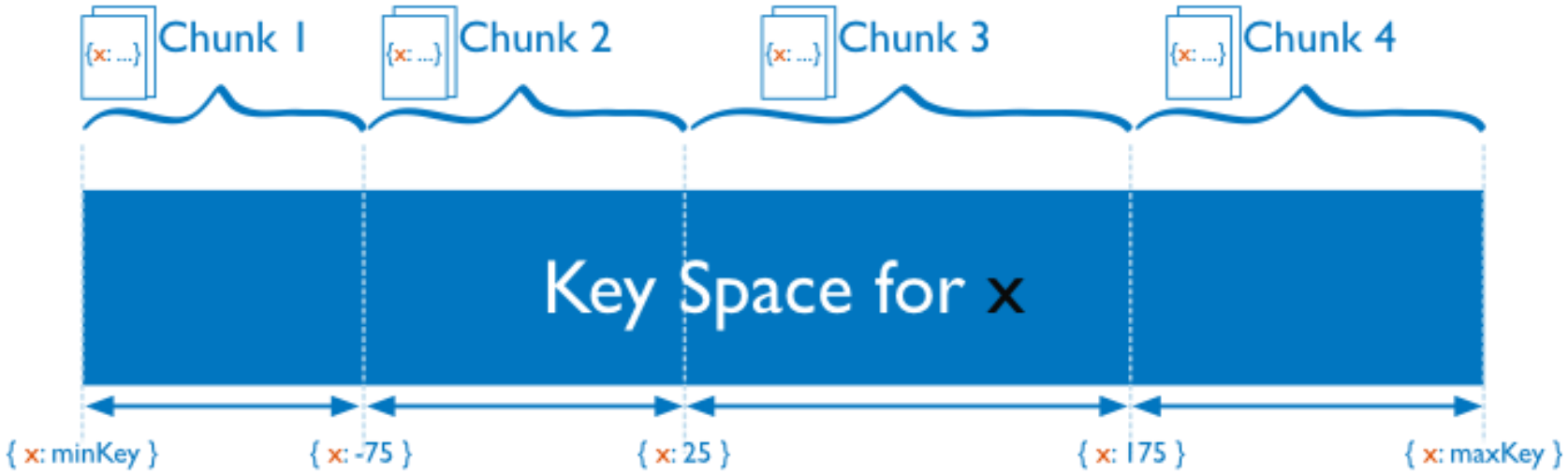
MongoDB sharding komponensek

- shards
  - tárolják az adatokat
- routers
  - interface kliens és shard között
- config server
  - metadata az adatokról



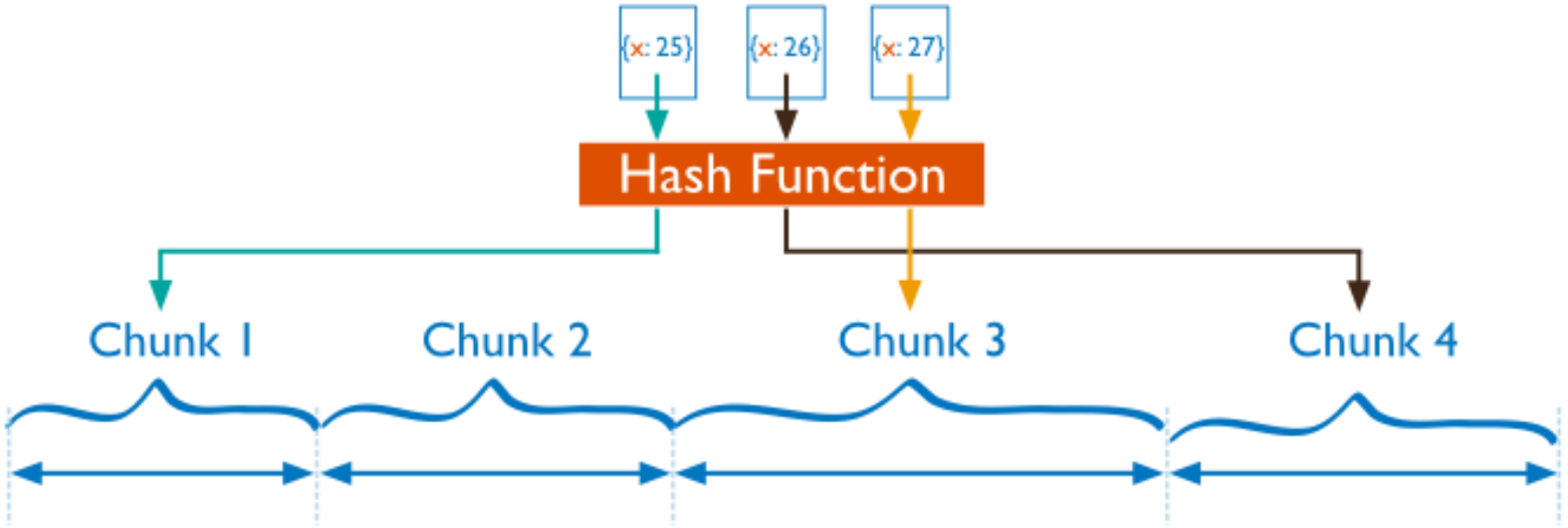
Sharding megoldások

- Range based sharding
  - shard key tartományok szerinti tárolás
  - hatékony tartományi lekérdezésekhez



Sharding megoldások

- Hash based sharding
  - adott hash függvény alapján osztjuk az adatot



Adatok beszúrása

- „write concern”

- Insert, update, delete utasításnál:
  - Erős ellenőrzés = lassú válasz
  - Gyenge ellenőrzés = gyors válasz

- MongoDB paraméterezhető

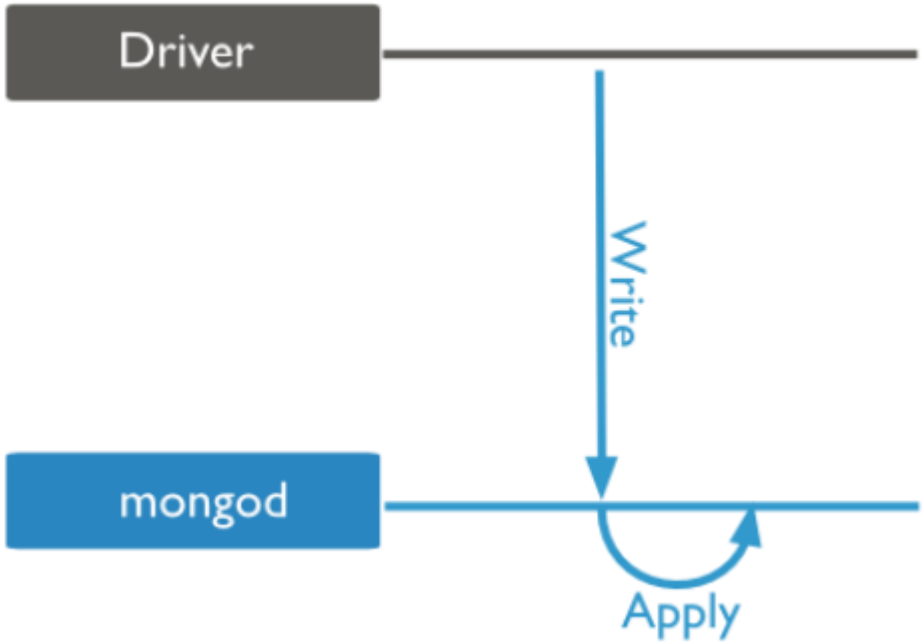
„write concern” szintjei

- w és j paraméterekkel állíthatóak be!

- Unacknowledged
- Acknowledged (w:1)
- Journaled (j:true)
- Replica Set Acknowledged (w:2)

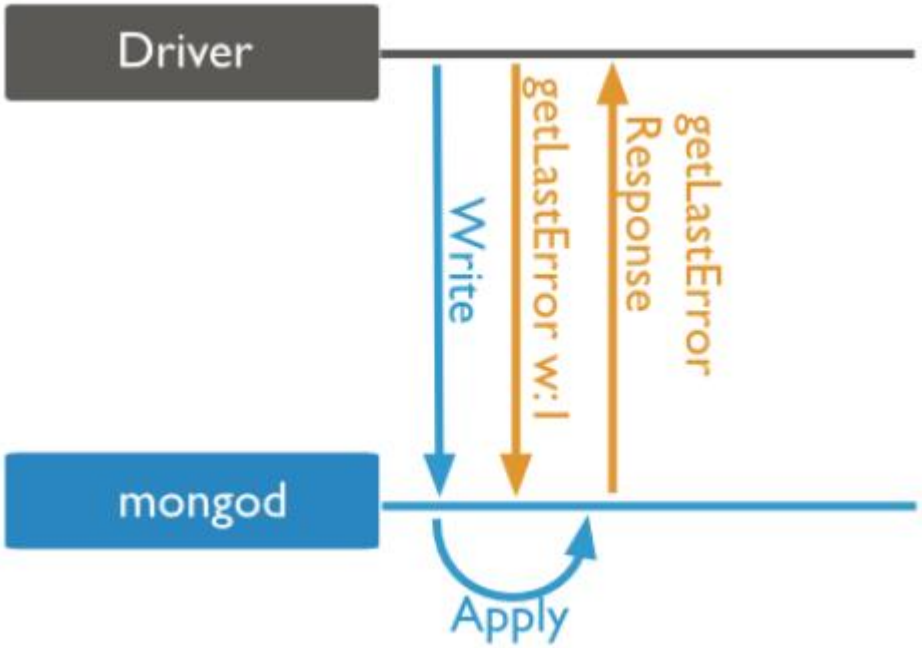
Unanknowledged

- Kliens nem kap visszajelzést az írásról
- Nagyon gyors írás
- Adatok elveszhetnek!
  - hálózati hiba
  - kulcs ütközés



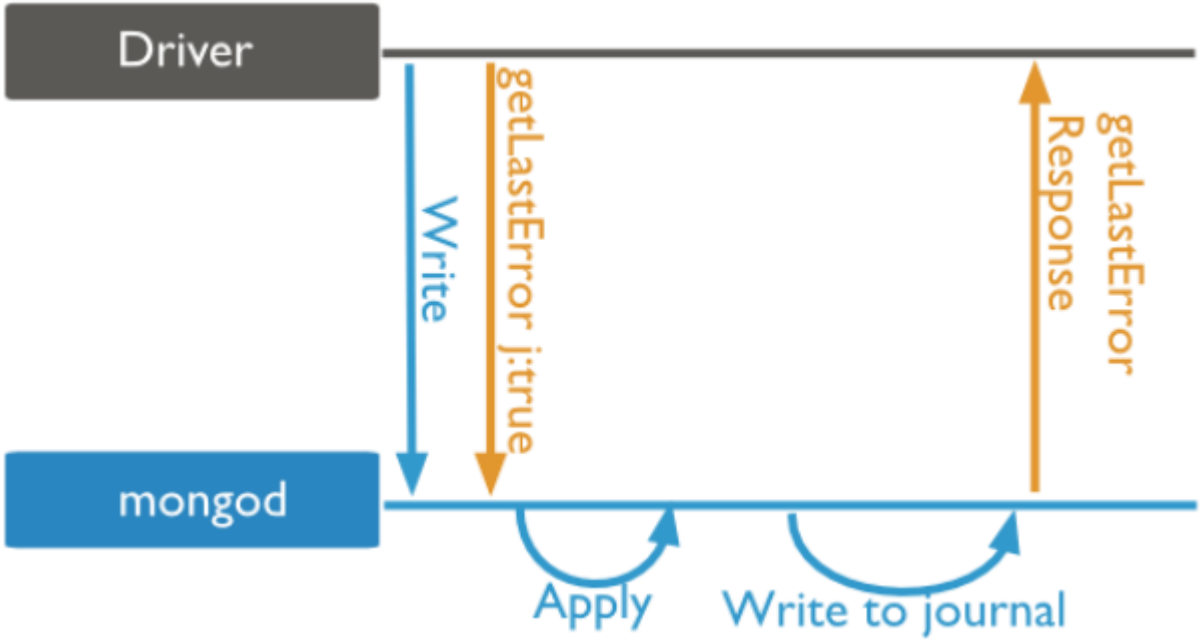
Acknowledged

- A mongod nyugtát küld
- Kliens érzékeli a hibákat (hálózati, kulcsütközés)
- Ez a default érték
- Nem garantálja hogy lemezre is kiírtuk az adatot!



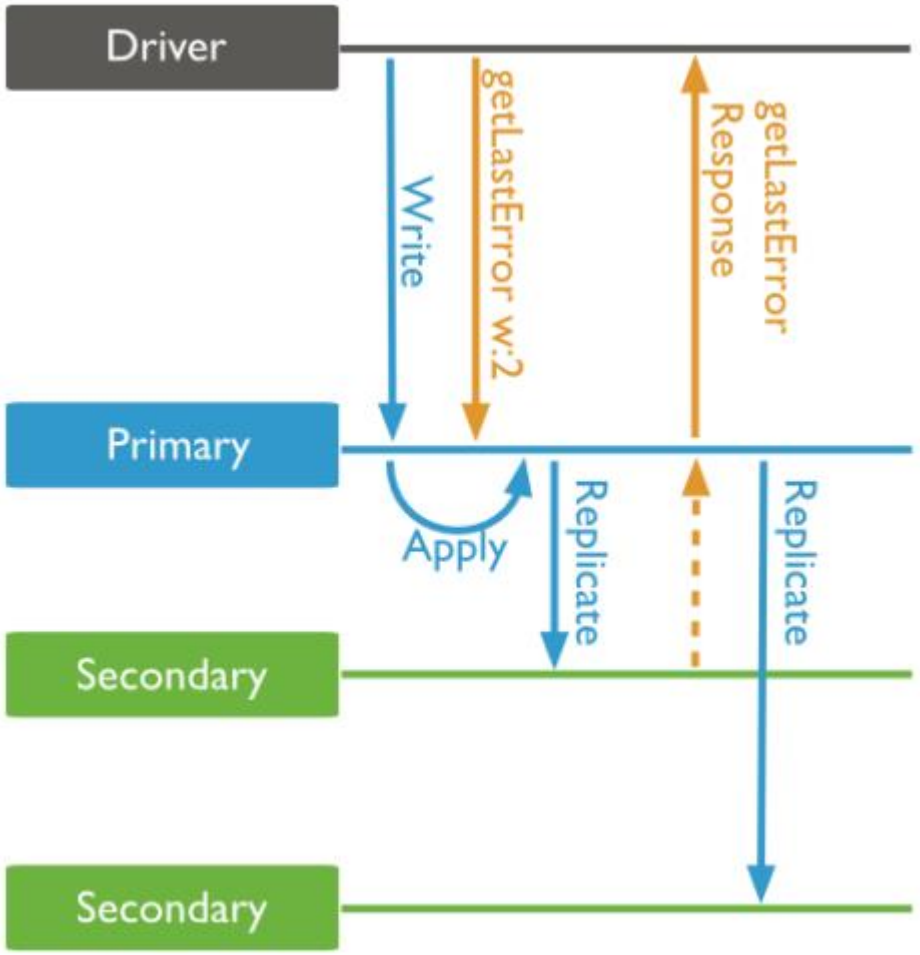
Journaled

- A mongod addig nem nyugtáz, amíg nem írja be a Journal-be. (logolás)
- Journal-t engedélyezni kell



Replica Set Acknowledged

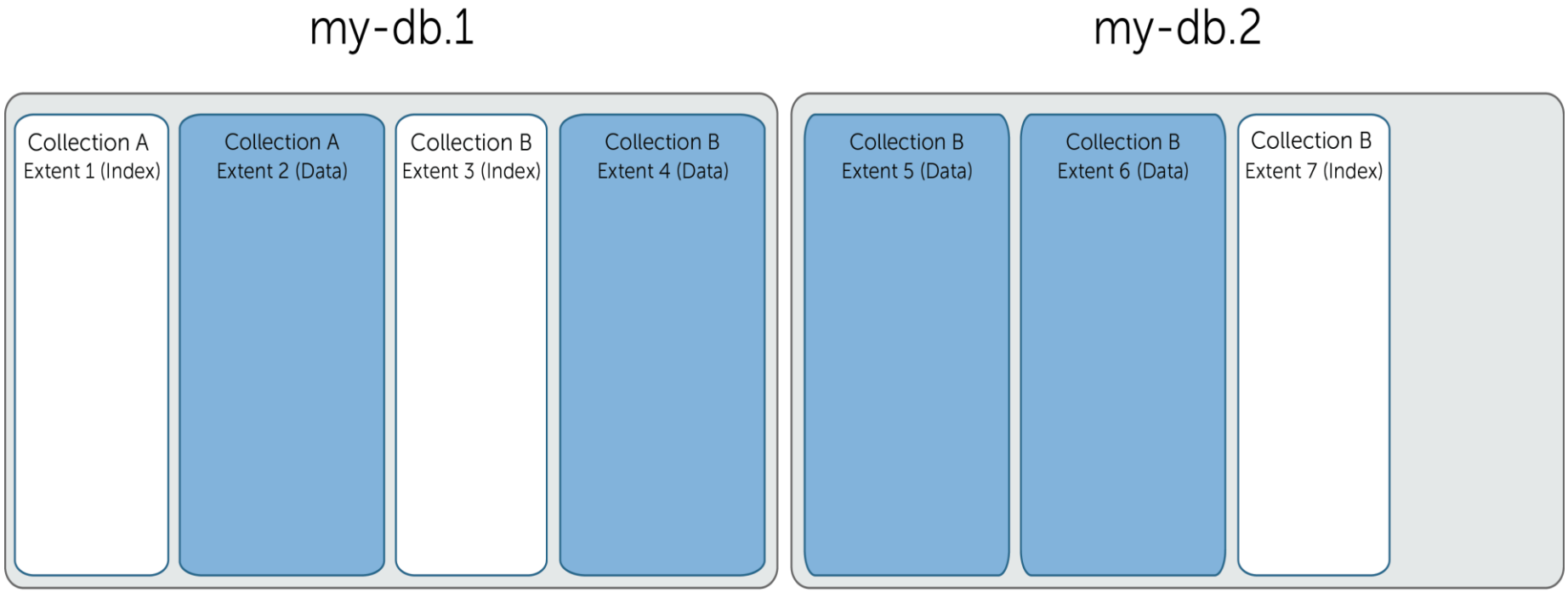
- Beállítható hány replikának kell visszaigazolni az írást, mielőtt a kliensnek jelzi a mongod



MongoDB adattárolás

- Minden MongoDB instance tartalmaza:
  - Namespace file
  - Journal file
  - Data file
- Data file tárolja extentekben:
  - BSON dokumentumok
  - Indexek
  - MongoDB metadata adat
- Extent: logikai konténer

Extents

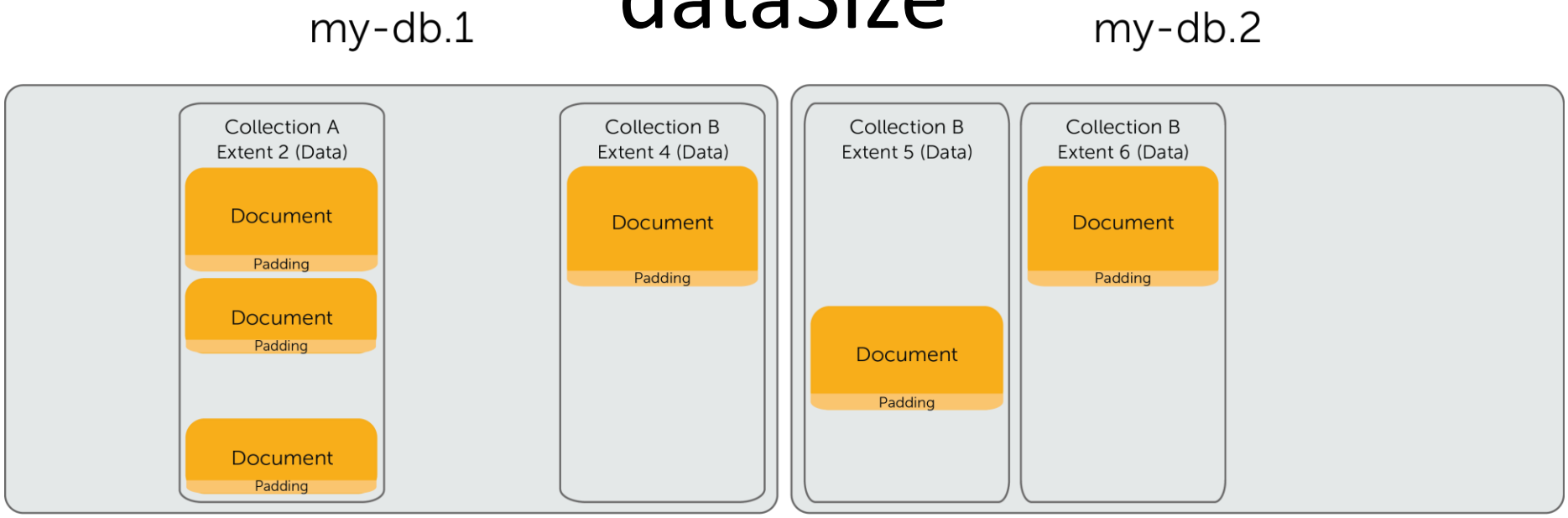


- Adat és index külön extent
- Egy extent egy collection-höz tartozik
- Egy collection lehet több extentben

Metrikák

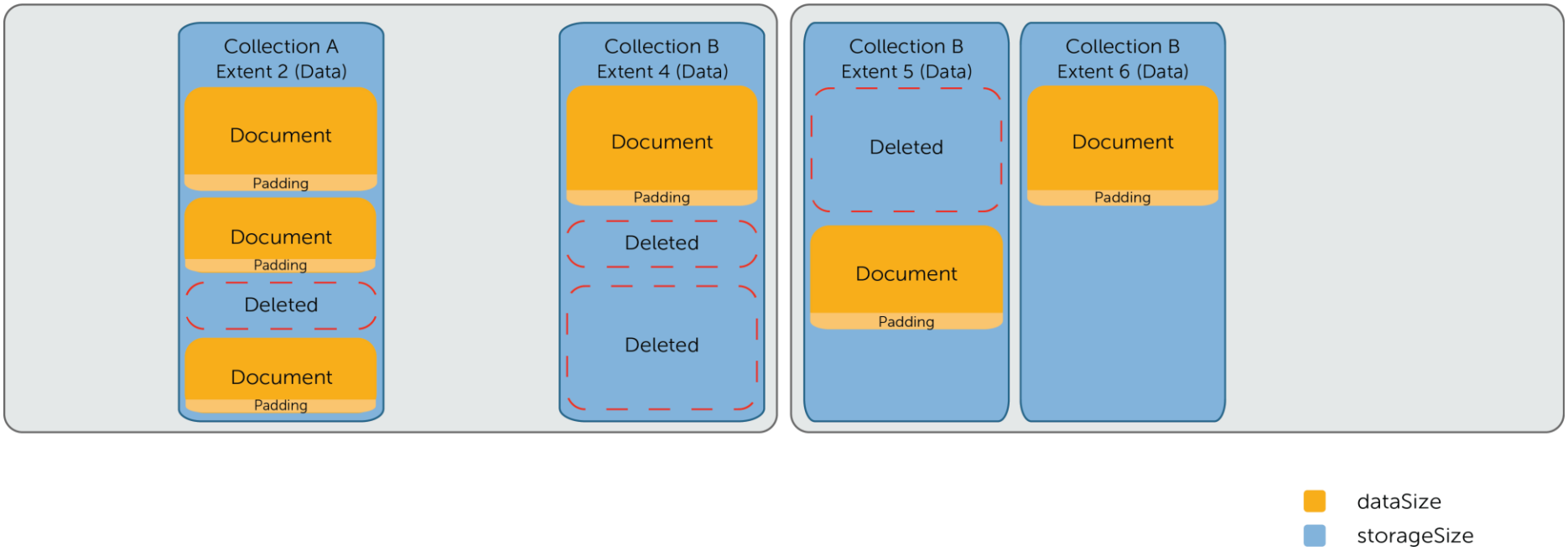
- db.stats()
  - statisztikák az adatbázisunkról
  - dataSize
  - storageSize
  - fileSize

dataSize



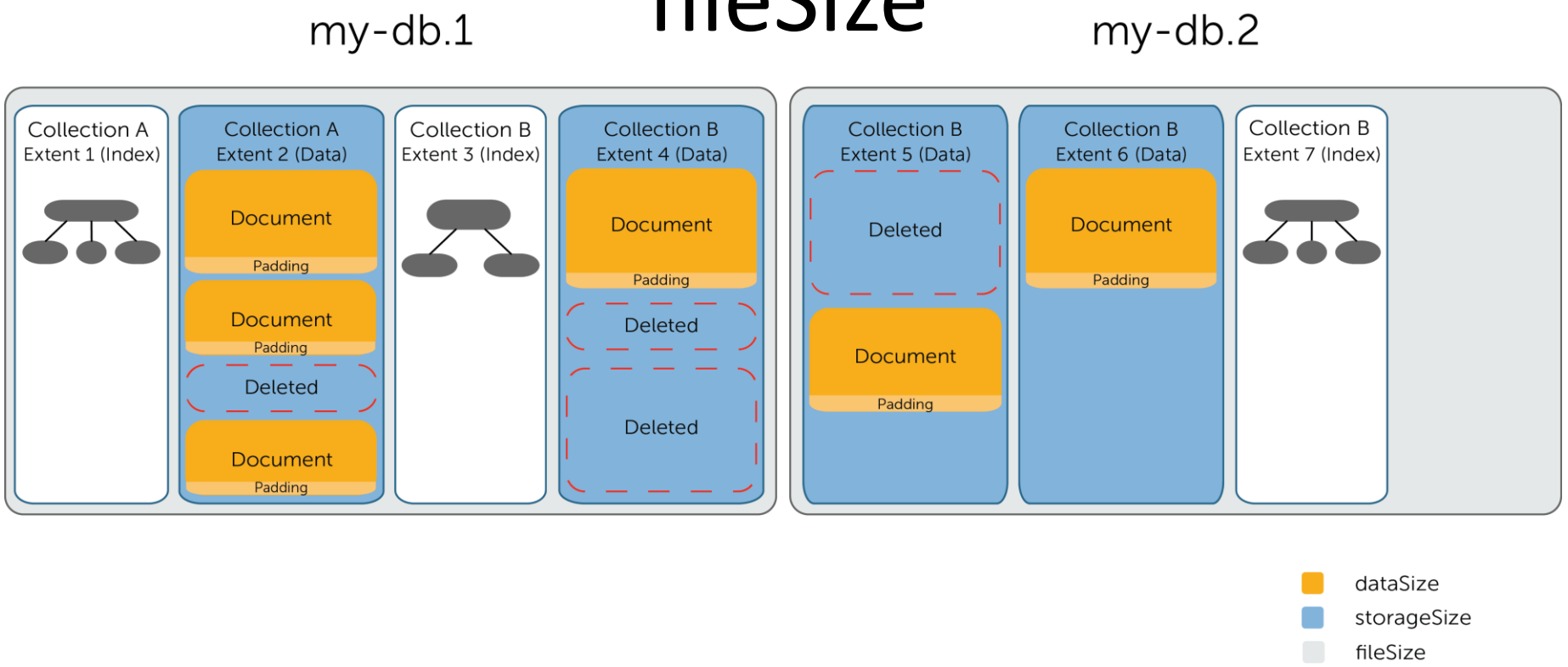
- A dataSize a dokumentumok és a lefoglalt mezők összege byte-ban
- Ha az adat változik,de elfér a lefoglalt területen akkor a dataSize nem változik

storageSize



- Az adat extent-ek mérete
- Ez tartalmazza a nem használt területeket is

fileSize



- adat extent-ek, index extent-ek és a nem használt területek a file-ban
- A teljes lefoglalt terület a lemezen

Mongo shell

- Elérhető: Windows, Linux, MacOS

- adatok importálása
  - mongoimport.exe --db test --collection restaurants --drop --file mongo_restaurant_dataset.json

- mongo shell elindítása
  - mongo.exe

Mongo shell

- adatbázisok listázása
  - show dbs
- collectionok listázása
  - show collections
- adatbázis kiválasztása
  - use test

Mongo CRUD

- Create
  - db.collection.insert(<document>)
  - db.collection.update(<query>, <update>, { upsert: true })
- Read
  - db.collection.find(<query>, <projection>)
  - db.collection.findOne(<query>, <projection>)
- Update
  - db.collection.update(<query>, <update>, <options>)
- Delete
  - db.collection.remove(<query>, <justOne>)

Mongo CRUD

- adatok listázása
  - db.restaurants.find()
- hány elem van
  - db.restaurants.count()
- mely éttermek vannak Manhattan kerületben?
  - db.restaurants.find({borough:"Manhattan"})

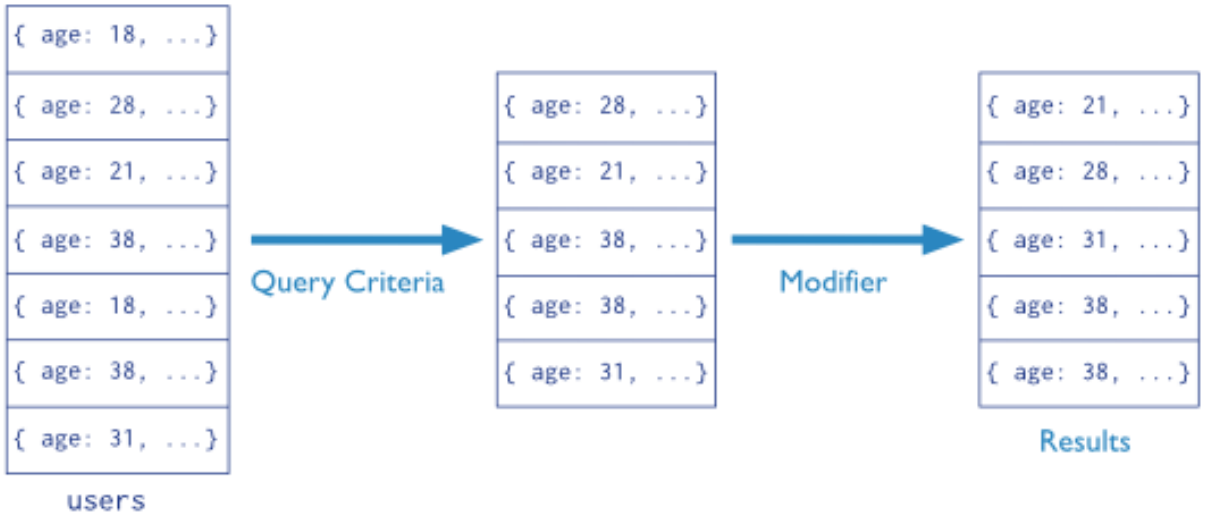
Mongo CRUD

```
db.users.find(
    { age: { $gt: 18 } },
    { name: 1, address: 1 }
).limit(5)
```

collection
query criteria
projection
cursor modifier

Collection Query Criteria Modifier

```
db.users.find( { age: { $gt: 18 } } ).sort( {age: 1 } )
```



restaurant példa


```
{
        "_id" : ObjectId("56ed68032ea3567be5c25d32"),
        "address" : {
                "building" : "284",
                "coord" : [ -73.9829239, 40.6580753 ],
                "street" : "Prospect Park West",
                "zipcode" : "11215"
        },
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "grades" : [
                { "date" : ISODate("2014-11-19T00:00:00Z"), "grade" : "A", "score" : 11 },
                { "date" : ISODate("2013-11-14T00:00:00Z"), "grade" : "A", "score" : 2 },
                { "date" : ISODate("2012-12-05T00:00:00Z"), "grade" : "A", "score" : 13 },
                { "date" : ISODate("2012-05-17T00:00:00Z"), "grade" : "A", "score" : 11 }
        ],
        "name" : "The Movable Feast",
        "restaurant_id" : "40361606"
}
```

Mongo CRUD

- Beágyazott objektumra szűrés
  - db.restaurants.find({ "address.zipcode": "10075" })
- keresés tömbben
  - db.restaurants.find({ "grades.grade": "B" })
- operátorok használata
  - db.restaurants.find({ "grades.score": { $gt: 30 } })
  - db.restaurants.find({ "grades.score": { $lt: 10 } })

Mongo CRUD

- feltételek összekapcsolása
- logikai és
  - db.restaurants.find({ "cuisine": "Italian", "address.zipcode": "10075" })
- logikai vagy
  - db.restaurants.find({ $or: [{ "cuisine": "Italian" }, { "address.zipcode": "10075" }] })
- rendezés
  - db.restaurants.find().sort({ "borough": 1, "address.zipcode": 1 })

Mongo CRUD

- update - frissíti az első „Juni” nevű objektumot
  - db.restaurants.update(
    { "name" : "Juni" },
    {
    $set: { "cuisine": "American (New)" },
    $currentDate: { "lastModified": true }
    }
)

- update – objektumon belül
  - db.restaurants.update(
    { "restaurant_id" : "41156888" },
    { $set: { "address.street": "East 31st Street" } }
)

Mongo CRUD

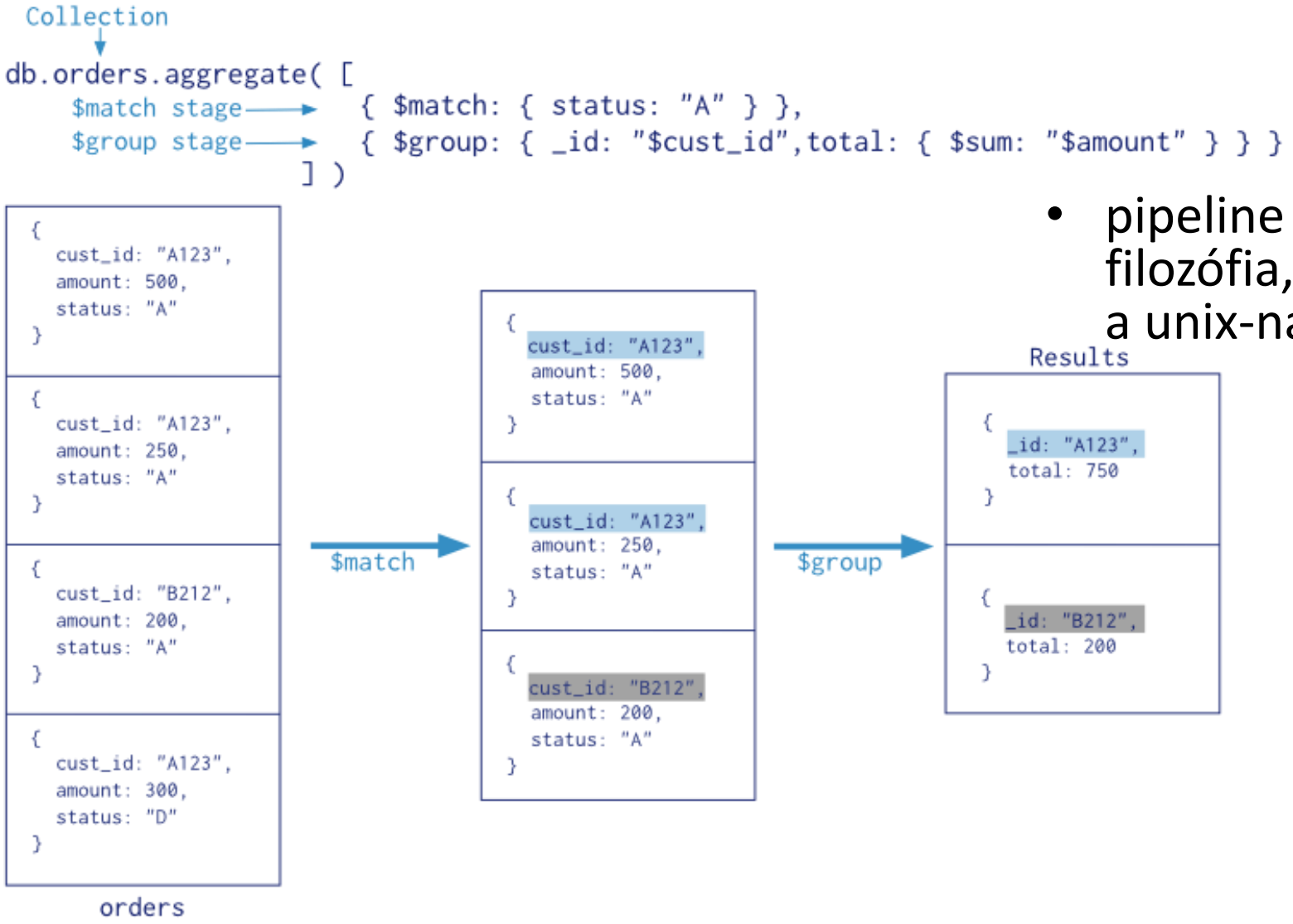
- update – több elem frissítése
  - db.restaurants.update(
    { "address.zipcode": "10016", cuisine: "Other" },
    {
      $set: { cuisine: "Category To Be Determined" },
      $currentDate: { "lastModified": true }
    },
    { multi: true}
)

Mongo CRUD

- törlés – összes egyező dokumentumot törli
  - db.restaurants.remove({ "borough": "Manhattan" })

- törlés – csak az elsőt
  - db.restaurants.remove({ "borough": "Queens" }, { justOne: true })

Aggregation framework



- pipeline filozófia, mint a unix-nál

Aggregation framework

- operátorok:
  - $project - vetítés
  - $match - szűrés
  - $limit - első n sor
  - $skip - hagyjon ki n sort
  - $group - csoportosítás
  - $sort - rendezés
  - $exists - létezik

Aggregation framework

- csoportosítás
  - db.restaurants.aggregate(
    [
    { $group: { "_id": "$borough",
    "count": { $sum: 1 } } }
    ]
    );

Példa lekérdezés

- Hotel adatbázis

- Melyik az a három állam, ahol a legtöbb olyan szálloda van, amelyiknek nincsen parkolója, és mégis 50% fölötti a kihasználtsága?

Példa lekérdezés

- Azon szállodák érdekelnek, amiknek nincs parkolója
- {$match: {parking: false}},

- Megtartjuk az _id-t, a state-et, illetve pct néven behozunk egy új (számított) attribútumot, ami a free/rooms hányadosként áll elő
- {$project: {_id:1, state:1, pct: {$divide: ["$free", "$rooms"]}}},

Példa lekérdezés

- Ezekből továbbengedjük azokat, ahol a pct 0.5 fölött van
- {$match: {pct: {$gt: 0.5}}},

- Csoportosítunk, a csoportosítás kulcsa a state attribútum, és az egyes eredmény-rekordoknak lesz még egy n nevű attribútuma, ami úgy áll elő, hogy az adott state kódú bemenő rekordokra szummázzuk az 1-et, azaz megszámoljuk az ilyen szállodákat
- {$group: {_id: "$state", n: {$sum: 1}}},

Példa lekérdezés

- A szállodaszám szerinti csökkenő sorrendből érdekel minket....
- {$sort: {n: -1}},

- Az első három.
- {$limit: 3},

Példa lekérdezés

- db.hotels.aggregate([
  {$match: {parking: false}},
  {$project: {_id:1, state:1, pct: {$divide:
  ["$free", "$rooms"]}}},
  {$match: {pct: {$gt: 0.5}}},
  {$group: {_id: "$state", n: {$sum: 1}}},
  {$sort: {n: -1}},
  {$limit: 3},])

Köszönöm a figyelmet!

Forrás

- http://blog.flux7.com/blogs/nosql/cap-theorem-why-does-it-matter
- http://crocodillon.com/images/blog/2013/mongodb-for-dbas__bson.png
- http://blog.mlab.com/2014/01/how-big-is-your-mongodb/
- https://docs.mongodb.org/manual/core/sharding-introduction/
- https://docs.mongodb.org/manual/core/replica-set-primary/