### **Razlage konceptov iz predstavitve o MongoDB**

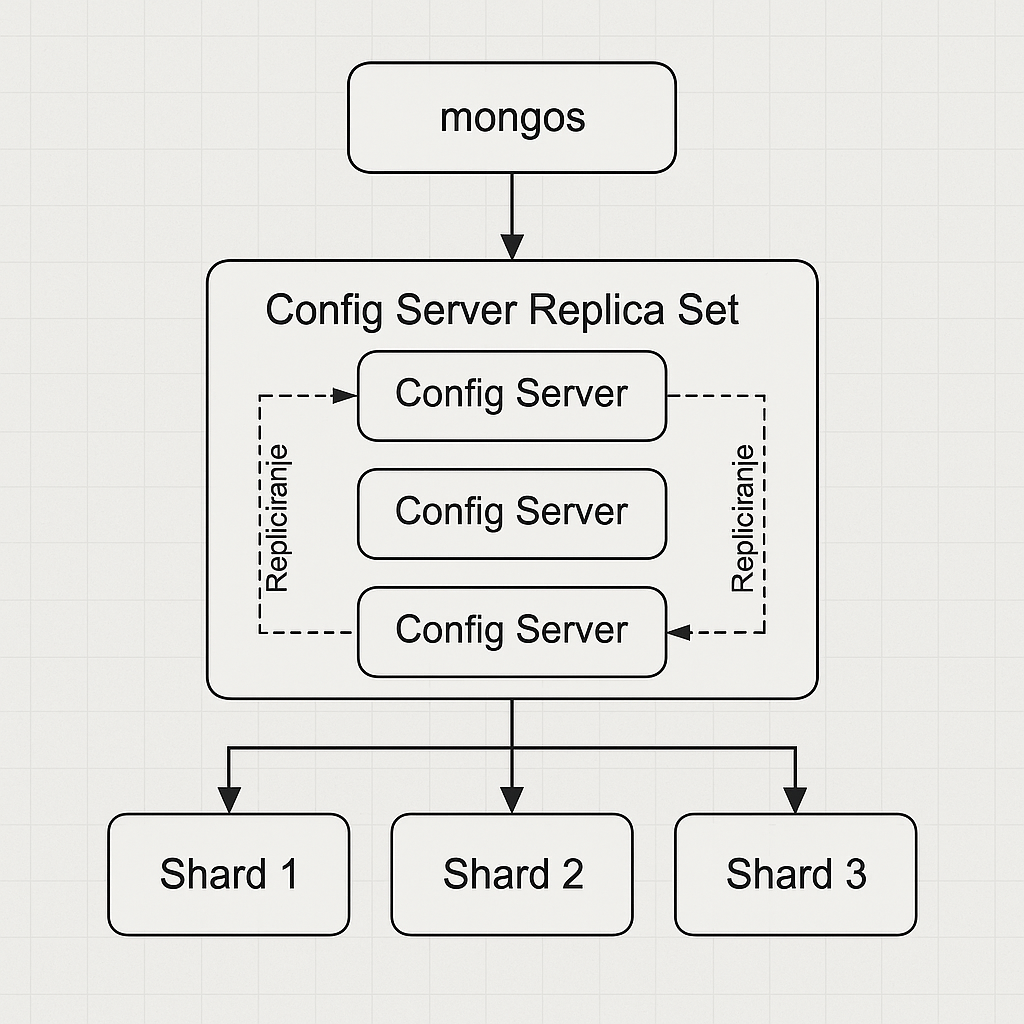
#### **1. MongoDB – osnovne značilnosti**

* MongoDB je dokumentno usmerjen sistem za upravljanje podatkovnih baz (SUPB). Namesto tabel in vrstic uporablja zbirke (collections) in dokumente (documents).
* Dokumenti so shranjeni v obliki **JSON** (berljiv za ljudi) ali **BSON** (binarna verzija, optimizirana za hitrost in dodatne tipe).
* Podpira **porazdelitev in replikacijo podatkov** – lahko se uporablja za večje sisteme z visoko razpoložljivostjo.
* Namesto SQL uporablja **JavaScript sintakso** v konzoli.

#### **2. JSON in BSON**

* **JSON (JavaScript Object Notation):** tekstovni zapis, enostaven za izmenjavo podatkov med spletnimi aplikacijami. Podpira osnovne tipe: string, število, boolean, null, polje.
* **BSON (Binary JSON):** binarni zapis, ni berljiv za ljudi, podpira več tipov (npr. datum, binarne podatke) in je hitrejši za obdelavo. MongoDB interno uporablja BSON, vendar omogoča komunikacijo z JSON.

#### **3. Ključne komponente sistema**

* **mongod** – strežniški program, ki upravlja podatkovno bazo.
* **mongos** – usmerjevalnik za delo z deljenimi podatki (sharding).
  + 
* **mongosh** – ukazna vrstica za uporabnike.
* **compass** – grafični vmesnik za delo z bazo.

#### **4. Primerjava SQL in MongoDB pojmov**

* **Tabela (SQL)** = **Zbirka (collection)** v MongoDB.
* **Vrstica (row)** = **Dokument JSON**.
* **Stolpec (column)** = **Polje v JSON dokumentu**.
* **Primarni ključ** = **polje \_id** (samodejno ustvarjen ObjectId, če ga uporabnik ne poda).

#### **5. Delo z dokumenti**

* Ustvarjanje dokumentov: insertOne(), insertMany().
* Brisanje: deleteOne(), deleteMany(), drop().
* Branje: find(), findOne().
* Posodabljanje: updateOne(), updateMany() z operatorji $set, $inc, $push, $pull, …
* Dokumenti imajo omejitev velikosti **16 MB**.

#### **6. Poizvedbe**

* Uporablja operatorje, npr. $eq, $ne, $gt, $lt, $in, $or.
* Primer: db.x.find({visina: {$gt: 180}}) poišče vse zapise z višino večjo od 180.
* Podpira tudi iskanje v gnezdenih dokumentih.

#### **7. Indeksiranje**

* Podpira do 64 indeksov na zbirko.
* Indeksi pohitrijo iskanje (struktura je B-drevo).
* Možno je indeksiranje gnezdenih dokumentov.
* Obstajajo posebni indeksi za **geoprostorske podatke** (npr. lokacije GPS).

#### **8. Kurzorji**

* Rezultat find() je kurzor (iterator), po katerem se lahko sprehajamo.
* Možnosti: limit(), skip(), sort().
* Uporabno za straničenje (pagination).

#### **9. PyMongo (dostop iz Pythona)**

* Povezava z MongoClient.
* Funkcije so podobne kot v konzoli, a s podčrtaji: find\_one(), insert\_one().
* Podatki se v Pythonu hranijo kot slovarji (dict).

#### **10. MapReduce in agregacije**

* **MapReduce** omogoča paralelno obdelavo velikih količin podatkov. Sestavljen je iz funkcij **map** (preslika vhodne podatke) in **reduce** (združi rezultate).
* V novejših verzijah MongoDB je MapReduce označen kot **zastarel**, saj ga nadomešča **agregacijski cevovod**.
* Agregacije omogočajo funkcije kot v SQL: SUM, AVG, GROUP BY.
* Cevovodi omogočajo sestavljanje več faz obdelave (npr. $match, $group, $sort).

#### **11. Prednosti MongoDB**

* Enostaven model (bližje programerskemu razmišljanju kot relacijske baze).
* Fleksibilno shranjevanje podatkov (ni fiksne sheme).
* Horizontalno skaliranje in replikacija.
* Od verzije 4 naprej omogoča tudi **ACID transakcije** (omejeno: do 16 MB sprememb, časovna omejitev 60 sekund).

### **Razlage in praktični primeri za MongoDB**

#### **1. Osnovni ukazi**

// uporaba baze

use studenti

// dodajanje dokumentov

db.studenti.insertOne({ime: "Ana", priimek: "Kovač", starost: 21})

db.studenti.insertMany([

{ime: "Boris", priimek: "Novak", starost: 22},

{ime: "Cvetka", priimek: "Zupan", starost: 20}

])

// iskanje

db.studenti.find() // vrne vse

db.studenti.find({starost: {$gt: 21}}) // starost > 21

db.studenti.findOne({ime: "Ana"}) // prvi zadetek

// posodabljanje

db.studenti.updateOne({ime: "Ana"}, {$set: {starost: 22}})

db.studenti.updateMany({}, {$inc: {starost: 1}}) // vsem poveča starost za 1

// brisanje

db.studenti.deleteOne({ime: "Boris"})

db.studenti.drop() // izbriše celo zbirko

#### **2. Indeksi**

// ustvarjanje indeksa

db.studenti.createIndex({ime: 1})

// unikatni indeks

db.studenti.createIndex({priimek: 1}, {unique: true})

// odstranitev indeksa

db.studenti.dropIndex({ime: 1})

#### **3. Gnezdeni dokumenti**

db.knjige.insertOne({

naslov: "Programiranje v Pythonu",

avtor: {ime: "Janez", priimek: "Horvat"},

strani: 350

})

// iskanje po gnezdenem polju

db.knjige.find({"avtor.ime": "Janez"})

#### **4. Uporaba kurzorjev**

var cur = db.studenti.find()

cur.forEach(function(x) { printjson(x) })

// straničenje

db.studenti.find().skip(2).limit(2)

#### **5. Primer MapReduce**

Recimo, da imamo evidenco izpitov:

db.izpiti.insertMany([

{ime: "Luka", letnik: 1, izpitov: 4},

{ime: "Vili", letnik: 1, izpitov: 6},

{ime: "Jaka", letnik: 1, izpitov: 2},

{ime: "Taja", letnik: 2, izpitov: 2},

{ime: "Dino", letnik: 2, izpitov: 4},

{ime: "Igor", letnik: 3, izpitov: 2},

{ime: "Nina", letnik: 3, izpitov: 3},

{ime: "Dani", letnik: 3, izpitov: 6},

{ime: "Pika", letnik: 3, izpitov: 5},

{ime: "Dasa", letnik: 3, izpitov: 5}

])

**Skupno število izpitov:**

var mapper = function() {

emit(this.letnik, this.izpitov)

}

var reducer = function(key, values) {

return Array.sum(values)

}

db.izpiti.mapReduce(mapper, reducer, {out: "rezultati"})

// rezultat

db.rezultati.find()

**Povprečno število izpitov na letnik:**

var mapper = function() {

emit(this.letnik, {st: 1, vsota: this.izpitov})

}

var reducer = function(key, values) {

var rezultat = {st: 0, vsota: 0}

values.forEach(function(x) {

rezultat.st += x.st

rezultat.vsota += x.vsota

})

return rezultat

}

var finalizer = function(key, reducedValue) {

reducedValue.povprecje = reducedValue.vsota / reducedValue.st

return reducedValue

}

db.izpiti.mapReduce(mapper, reducer, {out: "povprecja", finalize: finalizer})

// rezultat

db.povprecja.find()

Rezultat bo nekaj takega:

{ "\_id": 1, "value": { "st": 3, "vsota": 12, "povprecje": 4 }}

{ "\_id": 2, "value": { "st": 2, "vsota": 6, "povprecje": 3 }}

{ "\_id": 3, "value": { "st": 5, "vsota": 21, "povprecje": 4.2 }}

#### **6. Agregacijski cevovodi (modernejši pristop)**

// skupno število izpitov po letnikih

db.izpiti.aggregate([

{ $group: { \_id: "$letnik", skupno: { $sum: "$izpitov" } }}

])

// povprečno število izpitov po letnikih

db.izpiti.aggregate([

{ $group: { \_id: "$letnik", povprecje: { $avg: "$izpitov" } }}

])

S tem imaš praktične primere za delo z dokumenti, iskanje, indekse in posebej za **MapReduce** ter sodobnejši **agregacijski cevovod**.

#### **7. Vizualna primerjava: MapReduce vs. Agregacijski cevovod**

**Kdaj uporabiti katerega?**

| **Potreba / Lastnost** | **MapReduce** | **Agregacijski cevovod** |
| --- | --- | --- |
| Preprosti group-by, sum, avg, count | ✅ možno, a nepotrebno komplicirano | ✅✅ priporočeno (hitro, jedrnato) |
| Kompleksne transformacije v JS | ✅ (map/reduce v JavaScript) | ⚠️ delno z $function (počasneje), sicer raje predelaj faze |
| Hitrost pri velikih količinah podatkov | ⚠️ počasneje (JS, brez indeksne optimizacije v fazah) | ✅✅ hitreje (C++ implementacija, faze izrabijo indekse) |
| Skaliranje v grozdu | ✅ (paralelno), a dražje | ✅✅ naravno horizontalno skalira |
| Uporabnost danes | ❌ **zastarelo (deprecated)** | ✅ **priporočeno** |
| Enostavnost pisanja | ⚠️ treba pisati JS funkcije | ✅ deklarativno s stopnjami ($match, $group, …) |
| Vmesni/končni izhod v zbirko | ✅ z out: | ✅ z $out / $merge |
| Poraba virov | ⚠️ večja (JS engine, serializacije) | ✅ manjša (native) |

**Tipične faze agregacijskega cevovoda (mentalni zemljevid):**

1. $match → filtriraj čim prej (izkoristi indekse).
2. $project → izberi/pretvori polja (tudi izračuni).
3. $unwind → razvij polja tipa array.
4. $group → seštej/povpreči/štetje po ključu.
5. $sort → uredi rezultate (poskusi po $group, če je mogoče).
6. $limit / $skip → straničenje.
7. $lookup → *join* (po potrebi, ni brezplačen).
8. $out / $merge → zapiši rezultat v zbirko.

**Prevodi iz MapReduce v agregacije**

1. *Skupno število izpitov po letnikih*

// MapReduce (iz dokumenta): emit(letnik, izpitov) → sum

// Agregacije:

db.izpiti.aggregate([

{ $group: { \_id: "$letnik", skupno: { $sum: "$izpitov" } } }

])

1. *Povprečno število izpitov po letnikih*

// MapReduce: emit(letnik, {st:1, vsota:izpitov}) + finalize

// Agregacije:

db.izpiti.aggregate([

{ $group: { \_id: "$letnik", povprecje: { $avg: "$izpitov" } } }

])

1. *Filtriranje + skupine + sortiranje (best practice)*

db.izpiti.aggregate([

{ $match: { letnik: { $gte: 2 } } },

{ $group: { \_id: "$letnik", stStudentov: { $sum: 1 }, vsotaIzpitov: { $sum: "$izpitov" }, povprecje: { $avg: "$izpitov" } } },

{ $sort: { povprecje: -1 } },

{ $project: { letnik: "$\_id", \_id: 0, stStudentov: 1, vsotaIzpitov: 1, povprecje: 1 } }

])

**Pogoste pasti (in kako se jim izogniti):**

* *MapReduce brez potrebe:* če lahko, uporabi agregacije – bodo hitrejše in krajše.
* *Prepozen $match*: filtriraj na začetku, da zmanjšaš obdelavo.
* *Nepotrebni $project/$unwind*: vsaka faza nekaj stane – pusti le nujne.
* *Sortiranje pred grupiranjem:* pogosto je bolje sortirati po $group.
* *Veliki dokumenti v transakcijah:* ne pozabi na 16 MB omejitve dokumenta in omejitve trajanja transakcij.

**Mini “cheat sheet” za agregacije**

// count

[{ $count: "stevilo" }]

// distinct-like (seznam unikatnih vrednosti)

[{ $group: { \_id: "$polje" } }]

// top N po metrikah

[{ $sort: { metrika: -1 } }, { $limit: 10 }]

// group več metrik hkrati

[{ $group: { \_id: "$kljuc", sumX: { $sum: "$x" }, avgY: { $avg: "$y" }, minZ: { $min: "$z" } } }]

// write-back rezultata

[{ $merge: { into: "rezultati", on: "\_id", whenMatched: "replace", whenNotMatched: "insert" } }]

**Hitri kontrolni seznam za pripravo predavanja**

* Razloži *zakaj* je MapReduce zgodovinsko pomemben, a danes večinoma zamenjan.
* Pokaži isti problem na obeh pristopih → študentje vidijo razliko v kompleksnosti.
* Poudari vlogo indeksov in zgodnjega $match.
* Daj nalogo: “Pretvori ta MapReduce v agregacijski cevovod.”
* Zaključi s praktičnim demo na db.izpiti podatkih.

#### **8. Demo scenarij za predavanje**

**Cilj:** v 10–15 minutah pokazati razliko med osnovnimi operacijami, MapReduce in agregacijami.

**Priprava podatkov (db.izpiti):**

db.izpiti.drop()

db.izpiti.insertMany([

{ime: "Luka", letnik: 1, izpitov: 4},

{ime: "Vili", letnik: 1, izpitov: 6},

{ime: "Jaka", letnik: 1, izpitov: 2},

{ime: "Taja", letnik: 2, izpitov: 2},

{ime: "Dino", letnik: 2, izpitov: 4},

{ime: "Igor", letnik: 3, izpitov: 2},

{ime: "Nina", letnik: 3, izpitov: 3},

{ime: "Dani", letnik: 3, izpitov: 6},

{ime: "Pika", letnik: 3, izpitov: 5},

{ime: "Dasa", letnik: 3, izpitov: 5}

])

**Zaporedje demonstracije:**

1. **Pokaži osnovno poizvedbo:**

db.izpiti.find()

Vprašaš publiko: *Kaj vsebina pove? Kaj bi želeli izračunati?*

1. **Skupno število izpitov (MapReduce):**

var mapper = function() { emit(this.letnik, this.izpitov) }

var reducer = function(key, values) { return Array.sum(values) }

db.izpiti.mapReduce(mapper, reducer, {out: "rezultati"})

db.rezultati.find()

Vprašanje: *Ali je koda intuitivna ali nekoliko zapletena?*

1. **Skupno število izpitov (agregacija):**

db.izpiti.aggregate([

{ $group: { \_id: "$letnik", skupno: { $sum: "$izpitov" } }}

])

Vprašanje: *Kaj je bolj pregledno?*

1. **Povprečje po letnikih (MapReduce s finalizerjem):**

var mapper = function() { emit(this.letnik, {st: 1, vsota: this.izpitov}) }

var reducer = function(key, values) {

var r = {st: 0, vsota: 0}

values.forEach(function(x) { r.st += x.st; r.vsota += x.vsota })

return r

}

var finalizer = function(key, reduced) { reduced.povprecje = reduced.vsota / reduced.st; return reduced }

db.izpiti.mapReduce(mapper, reducer, {out: "povprecja", finalize: finalizer})

db.povprecja.find()

1. **Povprečje po letnikih (agregacija):**

db.izpiti.aggregate([

{ $group: { \_id: "$letnik", povprecje: { $avg: "$izpitov" } }}

])

1. **Zaključek:**

* Pokaži, da agregacije nadomestijo večino MapReduce potreb.
* Opozori, da so agregacije hitrejše, enostavnejše in priporočene.

**Dodatno vprašanje publiki:**

* *Kako bi izračunali največje/minimalno število izpitov na letnik?* (namig: $max, $min).

S tem demo scenarijem lahko v 10–15 minutah praktično prikažeš osnovne ukaze, MapReduce ter moderno alternativo z agregacijami.