NoSQL

FERIT

MongoDB

Arhitektura i modeliranje podataka

Predavanje 6

Sadržaj

- Što je MongoDB? Kratka povijest i filozofija.
- Ključni arhitekturalni koncepti
- Upoznavanje s MongoDB alatima
- Osnove rada u mongosh-u (interaktivni shell)
- Modeliranje podataka u MongoDB-u (ponavljanje i primjena)
- Obrasci modeliranja (*Design Patterns*)
- Kada normalizirati vs. kada denormalizirati?

Zastupljenost sustava koji koriste MongoDB

Izvor: db-engines.com

Rank Jul 2025	Rank Jun 2025	Rank Jul 2024	DBMS	Database Model	Score Jul 2025	Score Jun 2025	Score Jul 2024
1.	1.	1.	Oracle	Relational, Multi-model	1217.05	-13.33	-23.31
2.	2.	2.	MySQL	Relational, Multi-model	940.73	-12.85	-98.73
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	771.14	-5.61	-36.51
4.	4.	4.	PostgreSQL	Relational, Multi-model	680.89	+0.23	+41.98
5.	5.	5.	MongoDB	Document, Multi-model	403.83	+0.99	-25.99
6.	6.	↑ 7.	Snowflake	Relational	176.17	+1.68	+39.64
7.	7.	↓ 6.	Redis	Key-value, Multi-model	149.72	-2.01	-7.05
8.	8.	↑ 9.	IBM Db2	Relational, Multi-model	127.51	+2.38	+3.11
9.	9.	↓ 8.	Elasticsearch	Multi-model	118.83	-2.45	-12.00
10.	10.	10.	SQLite	Relational	115.44	-1.60	+5.49

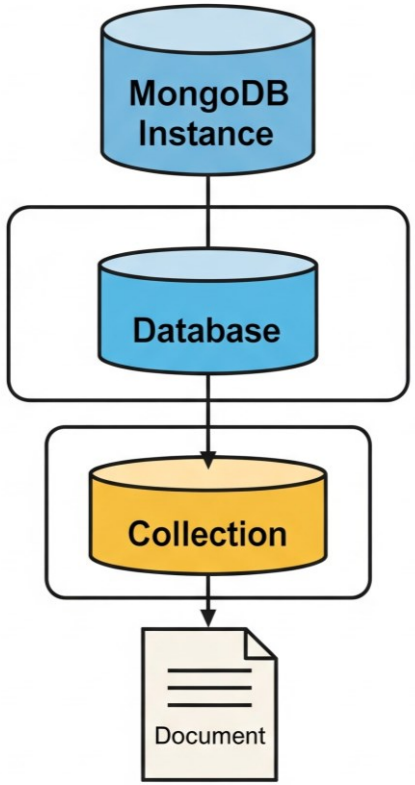
Rank Jul 2025	Rank Jun 2025	Rank Jul 2024	DBMS	Database Model	Score Jul 2025	Score Jun 2025	Score Jul 2024
1.	1.	1.	MongoDB	Document, Multi-model	403.83	+0.99	-25.99
2.	2.	2.	Databricks	Multi-model	108.04	+3.36	+24.74
3.	3.	3.	Amazon DynamoDB	Multi-model	84.15	+0.81	+13.20
4.	4.	4.	Microsoft Azure Cosmos DB	Multi-model	21.68	-0.75	-5.44
5.	5.	↑ 6.	Firebase Realtime Database	Document	14.35	+0.12	+0.47
6.	6.	↓ 5.	Couchbase	Multi-model	12.96	-0.83	-3.45
7.	7.	↑ 9.	Google Cloud Firestore	Document	8.11	+0.32	+1.36
8.	8.	↓ 7.	CouchDB	Document, Multi-model	7.12	+0.35	-0.70
9.	9.	↓ 8.	Realm	Document	6.45	-0.14	-0.88
10.	10.	10.	Aerospike	Multi-model	5.00	-0.26	-0.16

Što je MongoDB?

- Open-source, cross-platform, dokumentno orijentirana NoSQL baza podataka.
- Razvijen od strane tvrtke MongoDB Inc. (ranije 10gen).
- "Mongo" dolazi od riječi "humongous" (ogroman), ciljajući na rad s velikim količinama podataka.
- Pohranjuje podatke u fleksibilnim, JSON-*like* dokumentima (interno BSON).
- Filozofija:
  - Brzina i agilnost u razvoju.
  - Skalabilnost i visoka dostupnost kroz replikaciju i sharding.
  - Bogat i moćan jezik za upite.

Osnovna Struktura: Baze, Kolekcije, Dokumenti

- **MongoDB Server (Instanca):** Može sadržavati više neovisnih baza podataka.
- **Baza podataka (Database):**
  - Kontejner za kolekcije. Svaka baza ima svoj set datoteka na disku.
  - Primjeri: admin, local, config (interne), shop, blog (korisničke).
- **Kolekcija (Collection):**
  - Grupa srodnih dokumenata. Analogno tablici u RDBMS-u.
  - Ne nameće shemu.
  - Primjeri: users, products, orders.
- **Dokument (Document):**
  - Osnovna jedinica podataka. Jedan BSON objekt. Analogno retku u RDBMS-u.
  - Maksimalna veličina: 16MB.



BSON i ObjectId

- **Podsjetnik na BSON:**
  - Binarna reprezentacija JSON-a. Efikasan za parsiranje i pohranu. Podržava dodatne tipove podataka.
- **Ključni tip: ObjectId**
  - Jedinstveni 12-bajtni identifikator koji MongoDB automatski generira za _id polje ako ga korisnik ne specificira.
  - **Struktura ObjectId-a (12 bajtova):**
    - **4 bajta:** Vremenska oznaka (timestamp) u sekundama
    - **5 bajtova:** Nasumična vrijednost
    - **3 bajta:** Inkrementirajući brojač
  - **Karakteristike:**
    - Globalno jedinstven (u praksi).
    - Može se generirati na strani klijenta (drivera).
    - Sadrži vremensku informaciju, pa se može sortirati po vremenu kreiranja.

MongoDB Alati: mongod i mongosh

- **mongod (Mongo Daemon):**
  - **Glavni proces baze podataka.**
  - Pokreće se na serveru, sluša na mrežnom portu (default 27017).
  - Upravlja podacima, prihvaća konekcije, izvršava upite.
- **mongosh (MongoDB Shell):**
  - **Moderna, interaktivna JavaScript ljuska** za administraciju i rad s bazom.
  - Omogućava izvršavanje CRUD operacija, agregacija, administrativnih naredbi.
  - Nudi autocomplete, sintaksno bojanje, povijest naredbi.
- **MongoDB Compass:**
  - Službeni GUI (grafičko sučelje) za MongoDB.
  - Omogućava vizualizaciju podataka, kreiranje upita, analizu performansi, upravljanje shemom.
  - Izvrstan alat za vizualno istraživanje baze.

Osnove rada u mongosh-u

- **Primjeri osnovnih naredbi:**
  - `show dbs`: Prikazuje sve baze podataka.
  - `use <ime_baze>:` Prebacuje na odabranu bazu (kreira je ako ne postoji i ako se unese podatak).
  - `show collections:` Prikazuje sve kolekcije u trenutnoj bazi.
  - `db:` Varijabla koja pokazuje na trenutnu bazu.
  - `db.ime_kolekcije:` Referenca na kolekciju.

Prvi koraci: Umetanje dokumenata

- **Naredbe za umetanje:**
  - db.<kolekcija>.insertOne(dokument): Umeće jedan dokument.
  - db.<kolekcija>.insertMany([dokument1, dokument2, ...]): Umeće niz dokumenata.
- **Primjer u mongosh:**

```
// Prebacujemo se na bazu 'shop'
use shop

// Umećemo jedan proizvod u kolekciju 'products'
db.products.insertOne({
    _id: 1,
    name: "Laptop Pro X1",
    brand: "TechCorp",
    price: 999.99,
    tags: ["electronics", "laptops", "powerful"]
})
```

```
// Umećemo više proizvoda odjednom

db.products.insertMany([
    { name: "Bežični miš",
      brand: "Clicker",
      price: 25.50 },
    { name: "Mehanička tipkovnica",
      brand: "Typist",
      price: 75.00 }
])
```

Modeliranje Podataka: Ugnijezditi ili referencirati?

Ugniježđivanje (Embedding):

- Podaci se nalaze unutar roditeljskog dokumenta.
- **Prednost:** Brzo čitanje (jedan upit), atomičnost na razini dokumenta.
- **Mana:** Redundancija, problemi s ažuriranjem, ograničenje veličine dokumenta (16MB).

Referenciranje (Linking):

- Pohranjuje se samo ID povezanog dokumenta.
- **Prednost:** Nema redundancije (normalizacija), lakše ažuriranje, nema problema s veličinom.
- **Mana:** Potrebno više upita (ili $lookup) za dohvaćanje povezanih podataka ("join" na strani aplikacije).

Primjer 1: Ugniježđivanje (Embed) - *one-to-few*

Scenarij: Korisnik i njegove adrese. Korisnik obično ima mali, ograničen broj adresa.

Zašto ugnijezditi?

Odnosi "jedan-prema-malo".	Podaci o adresama se gotovo uvijek trebaju zajedno s podacima o korisniku.	Dohvaćanje svih podataka u jednom upitu je velika prednost.	Adrese nemaju smisla bez korisnika.

```
{
  "_id": "user123",
  "username": "pperic",
  "email": "pero@example.com",
  "adrese": [
    {
      "tip": "privatna",
      "ulica": "Glavna 5",
      "grad": "Zagreb"
    },
    {
      "tip": "poslovna",
      "ulica": "Ilica 1",
      "grad": "Zagreb"
    }
  ]
}
```

Primjer 2: Referenciranje (Link) - *one-to-many*

Scenarij: Proizvod i njegovi dijelovi (parts). Jedan proizvod se može sastojati od stotina dijelova, a jedan te isti dio (npr. vijak) može se koristiti u više različitih proizvoda.

Zašto referencirati?

Izbjegavanje masovne redundancije (podaci o vijku se ne ponavljaju).	Lakše ažuriranje podataka o dijelu (samo na jednom mjestu).	Podaci o dijelu imaju smisla i sami za sebe (npr. za praćenje zaliha).

Kolekcija products:

```
{
  "_id": "product_xyz",
  "naziv": "Super Mašina",
  "cijena": 15000,
  "dijelovi": [
    { "part_id": "vijak_m5", "kolicina": 50 },
    { "part_id": "motor_v2", "kolicina": 1 },
    { "part_id": "kuciste_a", "kolicina": 1 }
  ]
}
```

Kolekcija parts:

```
{ "_id": "vijak_m5", "naziv": "Vijak M5",
"proizvodjac": "Šaraf d.o.o." },
{ "_id": "motor_v2", "naziv": "Motor V2",
"proizvodjac": "Motori d.d." }
```

Modeliranje *many-to-many* odnosa

- **Primjer:** Artikli i tagovi. Jedan artikl može imati više tagova, a jedan tag može biti na više artikala.
- **Opcija 1: Niz referenci na obje strane**
  - **Kada:** Broj veza je relativno malen.
  - **Problem:** Ažuriranje zahtijeva promjene na dva mjesta.

Kolekcija articles:

```
{ "_id": "art1", "tags": ["nosql", "database"] }
{ "_id": "art2", "tags": ["nosql", "performance"] }
```

Kolekcija tags:

```
{ "_id": "nosql", "articles": ["art1", "art2"] }
{ "_id": "database", "articles": ["art1"] }
```

Modeliranje *many-to-many* odnosa

- **Primjer:** Artikli i tagovi. Jedan artikl može imati više tagova, a jedan tag može biti na više artikala.
- **Opcija 2: Niz referenci na jednoj strani (češći pristup)**
  - **Kada:** Upiti najčešće idu u jednom smjeru (npr. "pronađi sve tagove za artikl").
  - Puno jednostavnije za održavanje. Za pronalazak svih artikala s tagom "nosql", koristimo upit s indeksom na polju tags.

Kolekcija articles:

```
{
  "_id": "art1",
  "title": "...",
  "tags": ["nosql", "database", "mongodb"]
}
```

Arhitektura

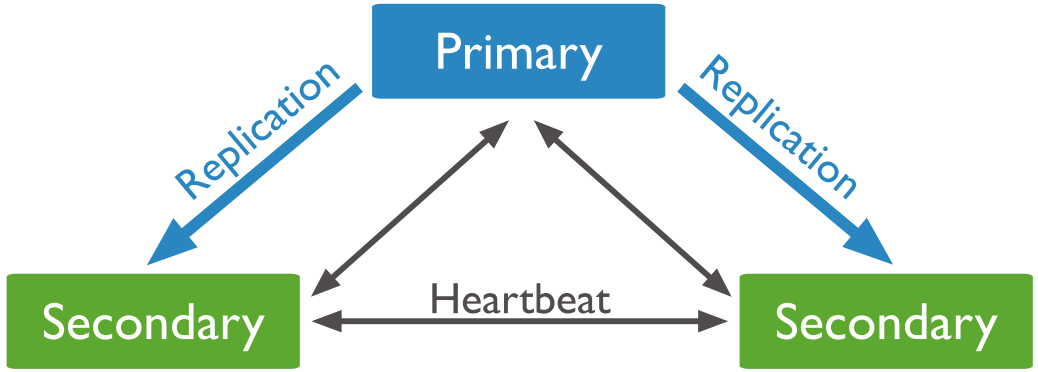
- **Tri glavna načina postavljanja MongoDB-a:**

1. Standalone

- Jedan mongod proces.
- Jednostavno za razvoj i testiranje.
- **Nije za produkciju!** Nema redundancije, nema visoke dostupnosti. Ako server padne, baza je nedostupna.

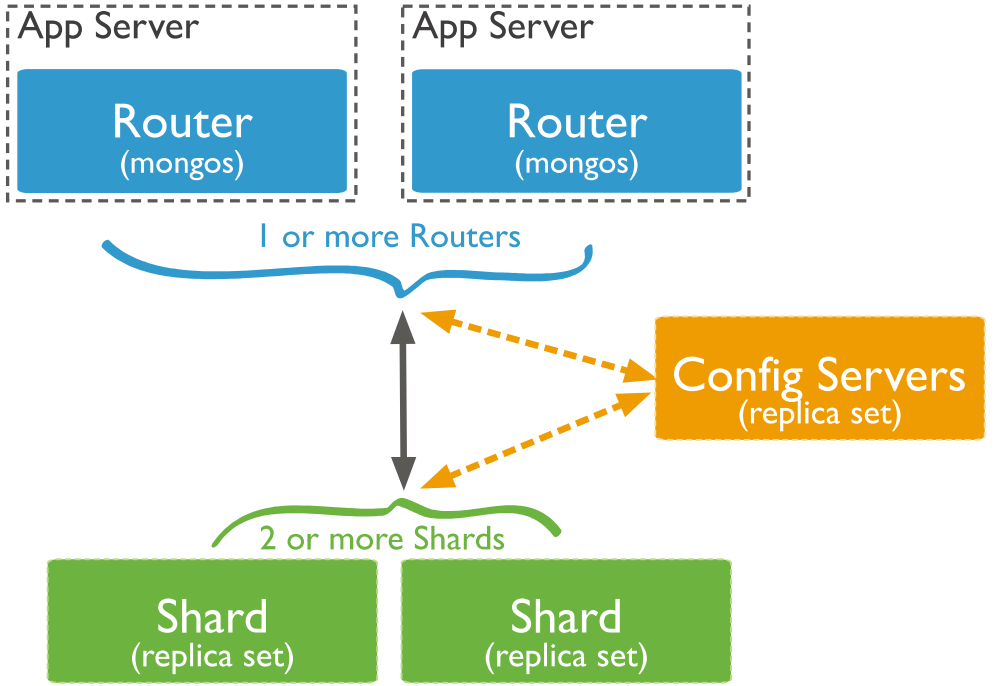
2. Replica Set:
(Skup replika)

- Grupa od (obično) 3 ili više mongod servera koji drže **istu kopiju podataka**.
- Jedan server je **Primary** (prima sve operacije pisanja).
- Ostali su **Secondaries** (repliciraju podatke s Primary-a, mogu služiti za čitanje).
- **Osigurava visoku dostupnost:** Ako Primary padne, jedan od Secondary-a se automatski promovira u novog Primary-a. **Ovo je preporučeni minimum za produkciju.**



3. Sharded Cluster:
(Klaster s particioniranjem)

- Za **horizontalno skaliranje** ogromnih skupova podataka.
- Podaci se dijele (particioniraju) preko više Replica Set-ova (koji se zovu **Shards**).
- Dodatne komponente: **mongos** (router za upite) i **Config Servers** (čuvaju metapodatke o distribuciji podataka).
- Kompleksno za postavljanje i održavanje.



Sažetak današnjeg predavanja

- MongoDB je vodeća dokumentna baza koja koristi BSON format.
- Osnovni gradivni blokovi su dokumenti i kolekcije. ObjectId je zadani primarni ključ.
- mongosh je moćan interaktivni shell za rad s bazom.
- Modeliranje podataka zahtijeva pažljivu odluku između ugniježđivanja i referenciranja, ovisno o odnosima i obrascima pristupa.
- Za produkciju, MongoDB se postavlja kao **Replica Set** (za visoku dostupnost) ili **Sharded Cluster** (za horizontalno skaliranje).