

*Računarstvo i informatika*

*Katedra za računarstvo*

*Elektronski fakultet u Nišu*

Napredne baze podataka  
**Document-oriented baze  
podataka**

Zimski semestar 2020/2021

# Sadržaj

- Document store
- MongoDB

# Document store

- **Document-oriented baza podataka** predstavlja kompjuterski softver čija je osnovna namena **skladištenje, pribavljanje i upravljanje dokument-oriented ili polustrukturanim podacima.**
- Document-oriented baza podataka = **Document store**
- Document store predstavlja jednu od kategorije **NoSQL baza podataka.**
- Za razliku od relacionih baza podataka koje su organizovane oko koncepta relacije (tabele), document-oriented baze podataka su organizovane oko apstraktnog koncepta **DOKUMENT**.
- Svaka document-oriented baza podataka implementira koncept dokumenta sa **različitim nivoima detaljnosti.**

# Document store

- Document-oriented baza podataka predstavlja **kolekciju dokumenata.**
- Sve informacije iz dokumenta se nalaze u samom dokumentu, nema povezanih tabela.
- **Nema potrebe za striktnim definisanje šeme baze podataka.**
- Za razliku od relacione baze podataka gde sve vrste imaju iste kolone, kod document store baza podataka **ne moraju svi dokumenti da imaju istu strukturu.**
- Ukoliko postoji potreba da se u dokument doda novo polje, novo polje se dodaje bez uticaja na prethodno uskladištene dokumente.
- Dokumenti ne čuvaju praznu vrednost za polje čija je vrednost nepoznata ili nedefinisana.

# Document store

- **Objekti se čuvaju kao dokumenti** – ne postoji problem object-relational impedance mismatch. Objektni model se snima u dokument i smešta u bazu podataka.
- **Dokumenti mogu da budu kompleksni** – čitav objektni model može da se učita/snimi odjednom. Nema potrebe za velikim brojem atomičnih CRUD operacija.
- **Dokumenti su nezavisni** – Povećavaju se performanse i smanjuju problemi vezani za konkurenatan pristup podacima.
- **Otvoreni formati** – Podaci se najčešće čuvaju korišćenjem otvorenih formata: XML, JSON, BSON (Binary JSON), YAML. Postoje i baze podataka koje čuvaju i dokumenta u binarnim formatima (PDF ili Office documents).

# Document store

- **Ne postoji striktna šema baze podataka** – Povećava se fleksibilnost sistema koji se razvija. Novi podaci mogu da se dodaju bez potrebe da se menjaju postojeća dokumenta u bazi podataka.

{

**Ime:**“Petar”,**Prezime:**“Petar”,**Adresa:**“Voždova 23”

}

{

**Ime:**“Goran”,**Prezime:**“Stojanović”,**Adresa:**“Dušanova 23”**Deca:** [{**Ime:**“Vera”, **Pol:** “Ž”},{**Ime:**“Todor”, **Pol:** “M”}

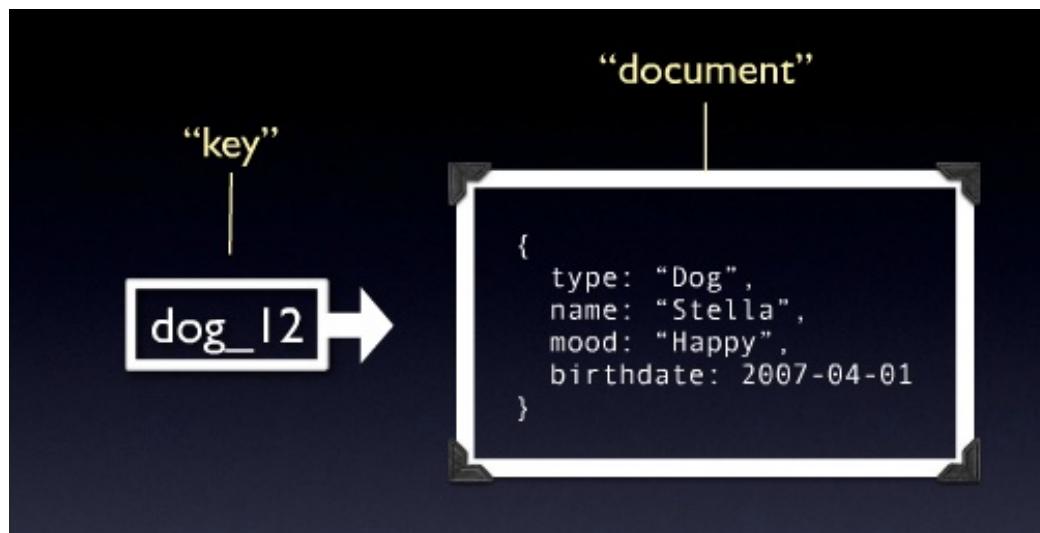
]

}

- **Čuvanje verzija dokumenata** – Većina rešenja podržava neki oblik čuvanja informacija o prethodnim verzijama dokumenata koja se nalaze u bazi podataka.

# Document store

- **Key/Value struktura** kod koje **VALUE** predstavlja polu-strukturani dokument čija je struktura razumljiva bazi podataka.
- Za **pristup dokumentima** u bazi podataka najčešće se koristi **vrednost ključa**. Ključ je najčešće string ali može biti URI ili neki oblik putanje.
- Baza je **indeksirana po vrednosti ključa** kako bi pristup dokumentima bio optimalan.



# Document store

- Osim pristupa dokumentima korišćenjem mehanizama ključa većina document store baza podataka nudi **API ili upitni jezik za pretraživanje dokumenata.**
- Cilj: pronaći dokumente kod kojih određeno polje ima određenu vrednost.
- **Ne postoji standardni API ili upitni jezik.** Većina document store baza podataka obezbeđuje sopstvenu implementaciju.
- Većina document store baza podataka obezbeđuje **RESTful API.**
- Često document store baze podataka obezbeđuju http servere koji obrađuju **standardne HTTP zahteve za podacima (PUT/GET/POST/DELETE).**

# Document store

- Za **organizaciju dokumenata** u okviru baze podataka koriste se različite tehnike:
  - Kolekcije
  - Mehanizam tagova
  - Hiperarhije direktorijuma
  - Nevidljivi metapodaci

# Document store

- Jedna od osnovnih karakteristika – **eventual consistency**.
- Eventual consistency omogućava korisnicima da menjaju dokumenta u bazi podataka. Izmene na dokumentima nisu vidljive u istom trenutku svim klijentima koji pristupaju dokumentima. Izmene postaju vidljive svim klijentima u nekom trenutku.
- Kao posledica povremeno se javlja **nekonzistentnost u podacima**.
- Cilj je **povećanje skalabilnosti i dostupnosti podataka**.

# Document store

- Primena:
  - **Dinamički podaci** - CMS (Content Management Systems) i CRM (Customer Relationship Management) objekti koje korisnici mogu da menjaju i prilagođavaju sopstvenim potrebama.
  - **Polustruktuirani podaci**
  - **Web podaci** – sesije, logovi, shopping cart i drugi podaci koji se održavaju za potrebe Web aplikacija. Document stores omogućavaju da se podacima pristupa kao celini jednim zahtevom ka udaljenom serveru.
  - **Obrada velike količine podataka** – dobra skalabilnost document store rešenja i podrška za distribuiranu obradu
- Problemi:
  - Kada se zahteva transakciona obrada (nema podrške za ACID transakcije)
  - Slučajevi u kojima je neophodno korišćenje SQL-a (potreba za velikim brojem spojeva)

# Document store

- Postojeća rešenja:
  - MongoDB
  - CouchDB
  - OrientDB
  - CosmosDB
  - Elasticsearch
  - Solr

# MongoDB

- **Initial release:** 2009
- **Stable release:** 4.4
- **Programming language:** C++
- **Operating system:** Cross-platform
- **Type :** Document store
- **License:** Open Source (GNU AGPL)
- **Website:** <http://www.mongodb.com/>
- **Verzije:**
  - MongoDB Community server
  - MongoDB Enterprise server
  - MongoDB Atlas

# MongoDB

- Namenjena za skladištenje **JSON dokumenata**
- Dokumenti se internu skladište u BSON formatu
  - Interni format
  - Binary JSON – efikasnije skladištenje i pretraga dokumenata
- Nema potrebe za postojanjem striktne šeme
- Podrška za veliki broj različitih programskih platformi: Java, .NET, PHP, Python, Ruby, ...
- Puna podrška za **indeksiranje dokumenata**
- Visok nivo **skalabilnosti**
- Visok nivo **dostupnosti**
- Podrška za **replikacije**
- **Kompleksni upiti** (predstavljeni u obliku JSON dokumenata)
- **Map/Reduce** podrška

# MongoDB

- **JSON (Java Script Object Notation)**
- **Otvoreni standard za razmenu podataka.** Tekstualni format koji je razumljiv i ljudima i mašinama.
- Bazira se na osnovama JavaScript jezika ali je generalno nezavistan od konkretnog programskog jezika.
- Osnovni tipovi JSON:
  - **Number** (double precision u pokretnom zarezu)
  - **String** (dvostruki navodnici, Unicode stringe backslash escape)
  - **Boolean** (true ili false)
  - **Array** (uređena sekvenca vrednosti irazdvojenih zarezima koje se nalaze zmeđu uglastih zagrada; vrednosti ne moraju biti istog tipa)
  - **Object** (neuređena kolekcija key/value parova; karakter : se koristi za razdvajanje key i value dela; parovi su razdvojeni zarezima i nalaze se između vitičastih zagrada; key je predstavljen kao string i mora biti jedinstven u okviru objekta)
  - **null**



# MongoDB

- Primer JSON dokumenta

```
{  
    "firstName": "John",  
    "lastName": "Smith",  
    "age": 25,  
    "address": {  
        "streetAddress": "21 2nd Street",  
        "city": "New York",  
        "state": "NY",  
        "postalCode": "10021"  
    },  
    "phoneNumber": [  
        {  
            "type": "home",  
            "number": "212 555-1234"  
        },  
        {  
            "type": "fax",  
            "number": "646 555-4567"  
        }  
    ]  
}
```



0

# MongoDB

MongoDB	SQL database
Document	Row / record
Collection	Table
_id	Primary key
DBRef	Foreign key
Aggregation	1:N relation
DBRefs array	M:N relation



# MongoDB



"In two months, engineers who never touched MongoDB were able to make a product that just scaled out of the box."

— Steven Bond, Forbes.com



Genentech



"We now have a modern, digitally-oriented application development environment which allows us to implement our innovative ideas as quickly as we create them."

— Peter Wolter, OTTO



"I want to be the city that's on the forefront, actually building a new path, building a new model for how cities operate."

— City of Chicago



SAILTHRU



"MongoDB gives you the ability to concentrate on your business and create the applications. Everything else is taken care of!"

— Steven Bond, Forbes.com



# MongoDB

- Startovanje servera

```
Administrator: C:\Windows\system32\cmd.exe - mongod --dbpath data
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\aca>d:

D:\>cd D:\mongodb-win32-x86_64-2.2.1\bin

D:\mongodb-win32-x86_64-2.2.1\bin>md data

D:\mongodb-win32-x86_64-2.2.1\bin>mongod --dbpath data
Tue Nov 20 23:40:29 [initandlisten] MongoDB starting : pid=2900 port=27017 dbpath=data 64-bit host=HAMURABI
Tue Nov 20 23:40:29 [initandlisten] db version v2.2.1, pdfile version 4.5
Tue Nov 20 23:40:29 [initandlisten] git version: d6764bf8dfe0685521b8bc7b98fd1fab8cfab5ae
Tue Nov 20 23:40:29 [initandlisten] build info: windows sys.getwindowsversion(major=6, minor=1, build=7601, platform=2, service_pack='Service Pack 1') BOOST_LIB_VERSION=1_49
Tue Nov 20 23:40:29 [initandlisten] options: { dbpath: "data" }
Tue Nov 20 23:40:29 [initandlisten] journal dir=data/journal
Tue Nov 20 23:40:29 [initandlisten] recover : no journal files present, no recovery needed
Tue Nov 20 23:40:29 [initandlisten] waiting for connections on port 27017
Tue Nov 20 23:40:29 [websvr] admin web console waiting for connections on port 28017
```



# MongoDB

- MongoDB CLI Shell

```
Administrator: C:\Windows\system32\cmd.exe - mongo
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\aca>d:

D:\>cd D:\mongodb-win32-x86_64-2.2.1\bin

D:\mongodb-win32-x86_64-2.2.1\bin>mongo
MongoDB shell version: 2.2.1
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
      http://docs.mongodb.org/
Questions? Try the support group
      http://groups.google.com/group/mongodb-user
>
```

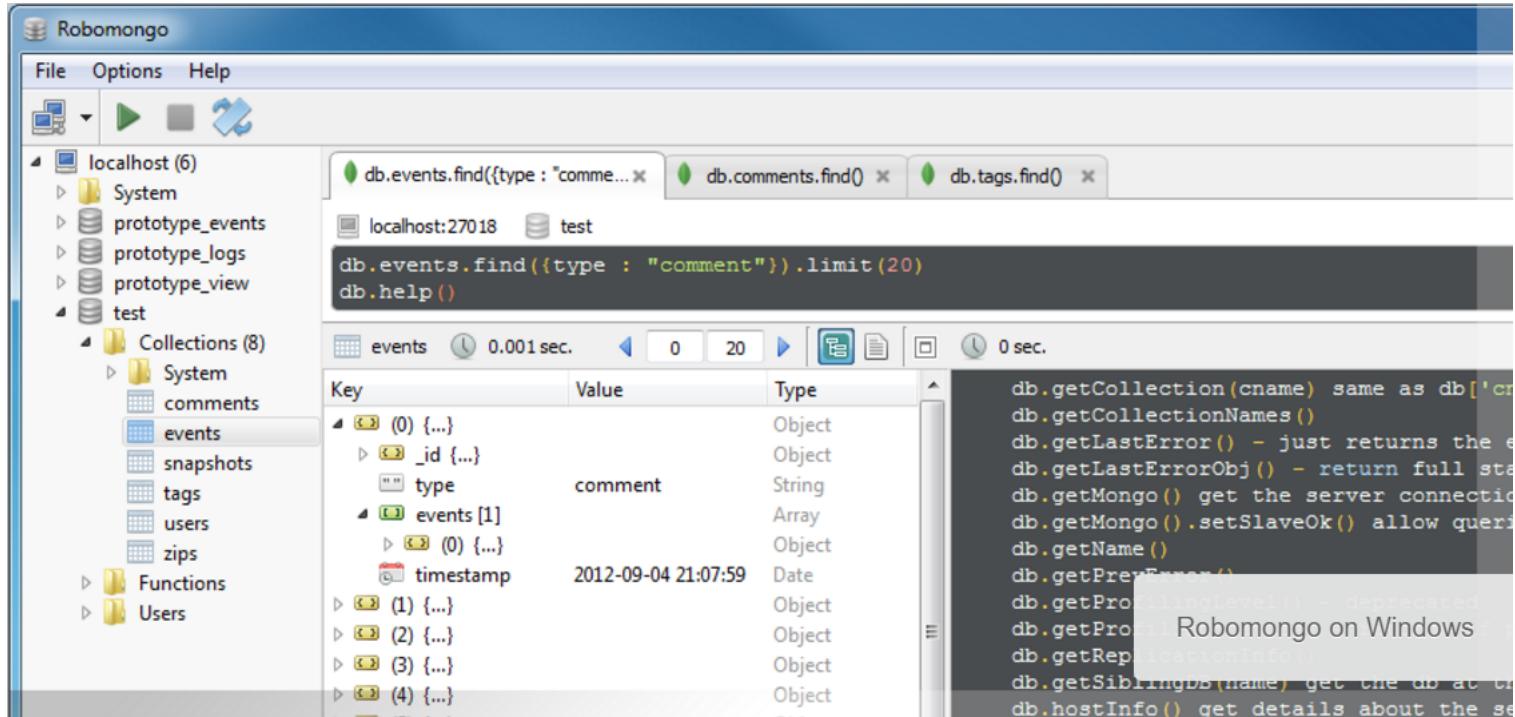
# MongoDB

- MongoDB server: <http://www.mongodb.org/downloads>

Name	Date modified	Type	Size
bsondump.exe	10/29/2012 3:31 PM	Application	8,881 KB
mongo.exe	10/29/2012 2:44 PM	Application	4,425 KB
<b>mongod.exe</b>	10/29/2012 3:11 PM	Application	8,943 KB
mongod.pdb	10/29/2012 3:12 PM	Program Debug D...	69,243 KB
mongodump.exe	10/29/2012 3:23 PM	Application	8,924 KB
mongoexport.exe	10/29/2012 3:29 PM	Application	8,887 KB
mongofiles.exe	10/29/2012 2:58 PM	Application	8,909 KB
mongoimport.exe	10/29/2012 3:25 PM	Application	8,910 KB
mongooplog.exe	10/29/2012 3:18 PM	Application	8,882 KB
mongoperf.exe	10/29/2012 3:05 PM	Application	8,882 KB
mongorestore.exe	10/29/2012 3:11 PM	Application	8,914 KB
mongos.exe	10/29/2012 3:16 PM	Application	6,360 KB
mongos.pdb	10/29/2012 3:16 PM	Program Debug D...	53,099 KB
mongostat.exe	10/29/2012 3:05 PM	Application	8,929 KB
mongotop.exe	10/29/2012 2:58 PM	Application	8,887 KB

# MongoDB

- Robo 3T: <https://robomongo.org/>

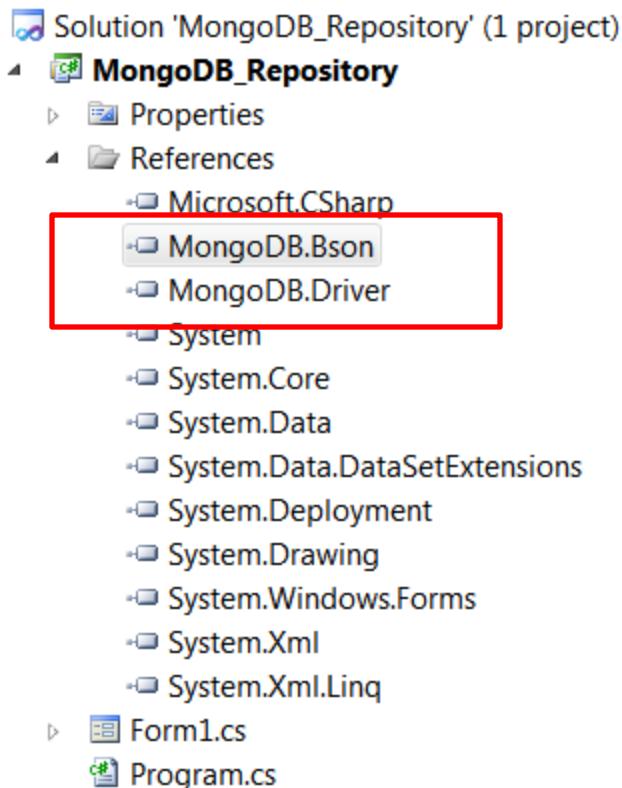


The screenshot shows the Robo 3T application interface. On the left, a sidebar displays the database structure under 'localhost (6)'. The 'test' database is expanded, showing collections: System, prototype\_events, prototype\_logs, prototype\_view, events, comments, snapshots, tags, users, and zips. The 'events' collection is selected and highlighted in blue. In the center, there are three tabs: 'db.events.find({type : "comment"})' (selected), 'db.comments.find()', and 'db.tags.find()'. Below the tabs, the connection details are shown: 'localhost:27018' and 'test'. The main query editor contains the command: `db.events.find({type : "comment"}).limit(20)`. To the right of the query editor is a results table titled 'events' with a timestamp of '0.001 sec.' and a row count of '0 sec.'. The table has columns: Key, Value, and Type. One row is visible: '\_id' (Object), 'type' (String, value: 'comment'), and 'events [1]' (Array). The 'events [1]' row contains a timestamp ('2012-09-04 21:07:59'). On the far right, a large text area displays the MongoDB documentation for the `getCollection` method.

```
db.getCollection(cname) same as db['cname']
db.getCollectionNames()
db.getLastErr() - just returns the error message
db.getLastErrObj() - return full status object
db.getMongo() get the server connection
db.getMongo().setSlaveOk() allow querying slave
db.getName()
db.getPrevError()
db.getProfilingLevel() - deprecated
db.getProfilingLevel() Robomongo on Windows
db.getReplicaSetNames()
db.getSiblingsDbNames() get the db names at the same host
db.hostInfo() get details about the server
```

# MongoDB

- mongo-csharp-driver: <https://github.com/mongodb/mongo-csharp-driver/downloads>



```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Windows.Forms;

using MongoDB.Driver;
using MongoDB.Bson;
```



# MongoDB

```
using MongoDB.Bson;

namespace MongoDB_Repository
{
    public class Radnik
    {
        public ObjectId Id { get; set; }
        public string Ime { get; set; }
        public string Prezime { get; set; }
        public string Adresa { get; set; }
        public float Plata { get; set; }
        public List<string> Oznake { get; set; }

    }
}

var connectionString = "mongodb://localhost/?safe=true";
var server = MongoServer.Create(connectionString);
var database = server.GetDatabase("test");
var collection = database.GetCollection<Radnik>("preduzece");

MongoCursor<Radnik> radnici = collection.FindAll();

foreach (Radnik r in radnici.ToArray<Radnik>())
{
    MessageBox.Show(r.Ime);
}
```

# MongoDB

- MongoDB indeks predstavlja **strukturu podataka** koja omogućava **brzo lociranje** dokumenata na osnovu **vrednosti određenih polja u dokumentima**.
- MongoDB indeksi se kreiraju na nivou kolekcije dokumenata.
- Indeksi mogu da se kreiraju nad **poljem dokumenta, nad delom polja ili nad više polja (kompozitni indeksi)**.
- MongoDB indeks je implementiran kao struktura B stabla.
- Indeksi ubrzavaju operacije pretrage ali usporavaju operacije ažuriranja podataka.
- Svaka MongoDB operacija (pretraga ili ažuriranje) koristi **samo jedan indeks**.
- Query optimizer bira indeks korišćenjem empirijskih podataka. Povremeno se izvršavaju **alternativni planovi** i za svaki tip upita se bira **plan sa minimalnim vremenom izvršenja**.

# MongoDB

- Za svaku kolekciju MongoDB kreira **podrazumevani \_id indeks**. Indeks \_id je unique indeks i ne može se obrisati.
- U polje \_id se može upisati bilo koja jedinstvena vrednost, ali se najčešće koristi **ObjectId vrednost**.
- Indeksi koje kreira korisnik su **secondary indeksi**.
- Za kreiranje indeksa se koristi metoda **ensureIndex()**.
- Za polja koja čine indeks može da se specificira **smer sortiranja**: I za rastući redosled i -I za opadajuće redosled.
- Kreira se indeks koji za ključ ima polje plata u kolekciji radnici. Indeks je sortiran u rastućem redosledu po vrednosti ključa.

```
db.radnic.ensureIndex({plata:I});
```

```
db.radnici.find({plata:{$gt: 35000}});
```

```
db.radnici.find({plata:{$gt: 35000}}).explain();
```

# MongoDB

- Indeksi nad ugnježdenim dokumentima:

```
db.radnici.insert({ime:"Jovana", prezime:"Petrović", adresa: {Ulica:"Kozaračka", Broj:12,  
Ulez:"II", Sprat: 3, Stan: 10}, Plata: 41000, oznake:["žena"]});
```

```
db.radnici.insert({ime:"Petar", prezime:"Vasić", adresa: {Ulica:"Nemanjina", Broj:19}, Plata:  
41000, oznake:["žena"]});
```

```
db.radnici.ensureIndex({adresa:1});
```

```
db.radnici.find({adresa: {$gte: {broj: 12}}});
```

```
db.radnici.find({adresa: {$gte: {ulica: "Kozaračka"}}});
```

```
db.radnici.find({adresa: {$gte: {ulica: "Kozaračka"}}, {adresa:1}});
```

# MongoDB

- Indeksi nad ugnježdenim poljima

```
db.radnici.ensureIndex({"adresa.ulica":1});
```

- Multikey indeksi (indeksi nad kolonama koje sadrže kolekciju)

```
db.radnici.ensureIndex({oznake:1});
```

```
db.radnici.find({oznake:"žena"});
```

- Kompozitni indeksi:

```
db.radnici.ensureIndex({plata:1, adresa:1});
```

```
db.radnici.find({plata:{$gte: 35000}, "adresa.ulica":"Nemanjina"});
```

```
db.radnici.find({plata:{$gte: 35000}});
```

```
db.radnici.find({"adresa.ulica":"Nemanjina"});
```

# MongoDB

- Ugnježdeni indeksi podržavaju upite nad bilo kojom **prefiksnom kombinacijom** ključeva koji čine indeks.
- Za indeks {plata:l, ime:l, adresa:l}
  - Pokrivene su kombinacije: plata-ime-adresa plata-ime, adresa
  - Nisu pokrivene kombinacije: adresa, ime, ime-adresa, plata-adresa
- **Smer sortiranja** ima **ključnu ulogu** kod kompozitnih indeksa ukoliko je potrebno sortirati dokumente nakon pribavljanja.
- **Unique indeksi** sprečavaju dupliranje vrednosti u indeksiranim poljima. Podrazumevano indeksi nisu unique.

```
db.radnici.ensureIndex({ime:l}, {unique:true});
```

- MongoDB dozvoljava postojanje samo jednog dokumenta koji nema vrednost za kolone nad kojima je kreiran unique indeks.

# MongoDb

- **Sparse indeksi** dozvoljavaju indeksiranje samo dokumenata u kojima postoji vrednost kolona nad kojima je indeks definisan. Podrazumevano MongoDB indeksi nisu sparse indeksi.

# MongoDB

- MongoDB obezbeđuje mehanizme za **agregaciju podataka**.
- Mehanizmi za agregaciju prihvataju kolekciju dokumenata, koja se prosleđuje kroz pipeline tj. niz međusobno povezanih operatora.
- Podržani su operatori:
  - \$project – određuje polja iz dokumenta koja će biti uključena u rezultat
  - \$match – filtriranje dokumenata na osnovu vrednosti polja
  - \$limit – definiše broj dokumenata koji prolazi kroz pipeline
  - \$skip – ignoriše određeni broj dokumenata na početku kolekcije
  - \$unwind – transformiše niz
  - \$group – grupisanje dokumenata
  - \$sort – sortiranje dokumenata



# MongoDB

```
db.radnici.aggregate({ $project : {ime : 1 ,prezime : 1}});  
db.radnici.aggregate({ $project : {_id:0, ime : 1 ,prezime : 1}});  
db.radnici.aggregate({ $project : {_id:0, ime : 1 ,prezime : 1, uvecanaplata: {$add:["$plata", 1000]}}});  
db.radnici.aggregate({ $project : {_id:0, ime : 1 ,prezime : 1, ulica: "$adresa.ulica"}});  
db.radnici.aggregate({ $match : {ime : "Jovana"} });  
db.radnici.aggregate({ $match : {plata : {$gt: 2000, $lt:4000 } }});  
db.radnici.aggregate({ $project : {ime : 1 ,prezime : 1, oznake: 1}}, {$unwind : "$oznake"});  
db.radnici.aggregate({$group : {_id : "$ime"} });  
db.radnici.aggregate({$group : {_id : "$ime", ukplata : {$sum : "$Plata"} }});  
db.radnici.aggregate({$group : {_id : "$ime", ukradnika:{$sum: 1}, ukplata : {$sum : "$plata"} }});  
db.radnici.aggregate({$group : {_id : "$ime", ukradnika:{$sum: 1}, ukplata : {$sum : "$plata"} }}, { $match : {ukplata : { $gte : 400000 } } });  
db.radnici.aggregate({$group : {_id : "$ime", ukradnika:{$sum: 1}, ukplata : {$sum : "$plata"} }}, { $match : {ukplata : { $gte : 400000 } } });  
db.radnici.aggregate({$group : {_id : "$ime", ukradnika:{$sum: 1}, ukplata : {$sum : "$plata"} }}, { $match : {ukplata : { $gte : 400000 } } }, {$sort : {ukplata: 1}});  
db.radnici.aggregate({$sort : {plata: 1}}, {$group : {_id : "$plata", imep: {$first : "$ime"} }});
```

# MongoDB

- MongoDB dokumenti su **ograničeni na veličinu od 16MB**.
- Za pamćenje većih dokumenata korist se **GridFS mehanizam**.
- GridFS mehanizam omogućava pamćenje velikih dokumenata koji su podeljeni na niz delova (chunk-ova).
- Za čuvanje velikih dokumenata koriste se dve kolekcije:
  - kolekcija **files** koja sadrži metapodatke o dokumentu
  - kolekcija **chunks** koja sadrži delov dokumenta
- Veličina delova na koje se dokument deli je obično 256KB.
- Prilikom rada sa kolekcijama files i chunks koristi se prefiks. Time je omogućeno da postoji veći broj GridFS skladišta u MongoDB bazi.
- Podrazumevan je prefiks fs, pa postoje fs.files i fs.chunks.
- Moguće je kreirati i druge prefikse. Npr: prefiks radnici podrazumeva kolekcije radnic.files i radnici.chunks.



# MongoDB

- Kolekcija files mora minimalno da poseduje sledeća polja:

```
{  
    "_id" : <unspecified>,                      // unique ID for this file  
    "length" : data_number,                        // size of the file in bytes  
    "chunkSize" : data_number,  
    256k                                            // size of each of the chunks. Default is  
    "uploadDate" : data_date,                      // date when object first stored  
    "md5" : data_string                            // result of running the "filemd5" command on  
    this file's chunks  
}
```

- Pored obaveznih polja moguće je dodati i druga proizvoljna polja:

```
{  
    "filename" : data_string,                     // human name for the file  
    "contentType" : data_string,                  // valid mime type for the object  
    "aliases" : data_array of data_string,        // optional array of alias strings  
    "metadata" : data_object,                    // anything the user wants to store  
}
```

# MongoDB

- Kolekcija chunks mora da poseduje sledeća polja:

```
{  
    "_id" : <unspecified>,           // object id of the chunk in the _chunks collection  
    "files_id" : <unspecified>,       // _id of the corresponding files collection entry  
    "n" : chunk_number,              // chunks are numbered in order, starting with 0  
    "data" : data_binary,            // the chunk's payload as a BSON binary type  
}
```



# MongoDB

```
var connectionString = "mongodb://localhost/?safe=true";
var server = MongoServer.Create(connectionString);
var db = server.GetDatabase("preduzece");

var fileName = "D:\\Untitled.png";
var newFileName = "D:\\new_Untitled.png";
using (var fs = new FileStream(fileName, FileMode.Open))
{
    var gridFsInfo = db.GridFS.Upload(fs, fileName);
    var fileId = gridFsInfo.Id;

    ObjectId oid = new ObjectId(fileId.ToString());
    var file = db.GridFS.FindOne(Query.EQ("_id", oid));

    using (var stream = file.OpenRead())
    {
        var bytes = new byte[stream.Length];
        stream.Read(bytes, 0, (int)stream.Length);
        using (var newFs = new FileStream(newFileName, FileMode.Create))
        {
            newFs.Write(bytes, 0, bytes.Length);
        }
    }
}
```

# MongoDB

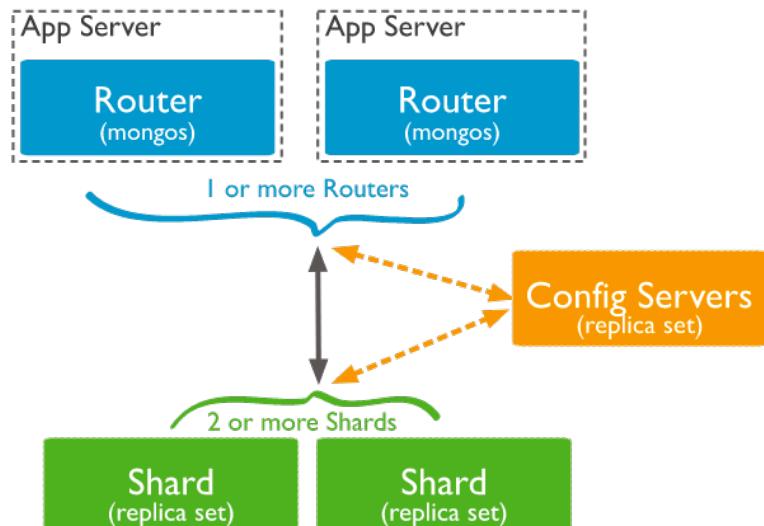
- **MongoDB mehanizam replikacije** omogućava sinhronizaciju podataka između većeg broja MongoDB instanci.
- Skup instanci (najmanje dve) između kojih je uspostavljen mehanizam replikacije predstavlja **MongoDB replication set**.
- Jedna instanca je **primarna** a sve ostale su **sekundarne**.
- Svi klijenti upisuju podatke u primarnu instancu,. Podaci se nakon toga asinhrono sinhronizuju sa sekundarniminstancama.
- Replikacija ima zadatak da obezbedi redundantnost, poveća dostupnost i olakša neke administrativne zadatke poput pravljenja rezervnih kopija.
- Većina produkcionih okruženja implementira mehanizam replikacije.
- MongoDB obezbeđuje podršku za failover mehanizam. U slučaju optkaza primarne instance, ostale instance mehanizmom glasanja (voting) proglašavaju jednu od sekundarnih instanci za novu primarnu.

# MongoDB

- MongoDB obezbeđuje **sharding mehanizam**.
- **Sharding mehanizam** omogućava **particionisanje** kolekcije dokumenata i njihovu **distribuciju** na veći broj instanci (shards).
- Sharding mehanizam ima zadatak da poveća kapacitet sistema (scale out).
- Osobine:
  - Particionisanje podataka na osnovu vrednosti zadatog ključa. Particija obuhvata dokumente koji se nalaze u određenom opsegu vrednosti ključa. Kada particija preraste fizičke kapacitete instance, ona se deli na više manjih particija.
  - Autmatski se balansira količina dokumenata na različitiminstancama. Bitan je pravilan izbor vrednosti ključa.
  - Distribuirane instance su popotpuno transparentne za korisnike
  - Opterećenje se raspoređuje na veći broj instanci čime se povećava ukupni kapacitet sistema.

# MongoDB

- Shard konfiguracija
  - 1. Tri konfiguraciona MongoDB servera – sadrže metapodatke i mapiraju particije i MongoDB instance
  - 2. Više MongoDB replica setova koji sadrže podatke
  - 3. Veći broj mongos instanci koji se zadužene za rutiranje. **Mongos** instance rutiraju klinentske zahteve ka odgovarajućiminstancama na osnovu metapodataka.



# MongoDB

- **Map/Reduce** - Programski model za obradu velike količine podataka.
- Tipična namena je **distribucija obrade na veći broj čvorova**.
- Omogućava paralelnu obradu velike količine podataka na distribuiranim čvorovima. Obrađuju se podaci koji se nalaze u file sistemu ili u bazi podataka.
- Odvija se u dva koraka:
  - **Map** – master čvor prihvata ulaz, deli ga na manje potprobleme i distribuira ih worker čvorovima. Worker čvor može ponoviti isiti korak kreirajući stablo čvorova koji obavljaju procesiranje podataka. Worker čvor obrađuje manji problem, i rezultate obrade prosleđuje master čvoru.
  - **Reduce** – master čvor prikuplja od worker čvorova rezultate obrade potproblema i kombinuje ih u rezultat originalnog problema
- Postoje implementacije u različitim programskim jezicima.
- Popularna implementacija: **Apache Hadoop**

# MongoDB

```
var mapFunction1 = function() {emit(this.ime, this.plata);};  
var reduceFunction1 = function(key, values) {return Array.sum(values);};  
db.radnici.mapReduce(mapFunction1, reduceFunction1, { out:  
    "map_reduce_example" });  
var c = db.radnici.mapReduce(mapFunction1, reduceFunction1, { out:  
    "map_reduce_example" });  
db[c.result].find();
```

# MongoDB

- Content Management System (CMS)
- Katalozi proizvoda
- Operational Intelligence
- Real-time analytics
- IoT
- Data integration (Single-view)
- Mobilne aplikacije

# MongoDB

- Ugnježdeni dokumenti:
  - Svi podaci su unutar jednog dokumenta (ograničenje od 16MB)
  - Denormalizovana šema, podaci mogu da budu redundantni
  - Pogodno za relacije tipa *contains* i *one-to-many*
  - Bolje performanse read operacija
  - Operacije ažuriranja nad dokumentom su atomične

```
{  
  _id: <ObjectId1>,  
  username: "123xyz",  
  contact: {  
    phone: "123-456-7890",  
    email: "xyz@example.com"  
  },  
  access: {  
    level: 5,  
    group: "dev"  
  }  
}
```

The diagram illustrates an example of an embedded sub-document in a MongoDB document. The document structure is shown in a blue-bordered box. Two green arrows point from the text 'Embedded sub-document' to the 'contact' field and the 'access' field, indicating that these fields contain their own separate documents.

# MongoDB

- Reference između dokumenata:
  - Normalizovana šema, nema redundanse u podacima
  - Mogu se predstavljati kompleksne relacije (*many-to-many* relacije i za modeliranje kompleksnih hijerarhija u skupu dokumenata)
  - Koriste se kada koristi od efikasnih *read* operacija nije toliko velika da bi opravdala veliku redundansu u podacima.

