|  |  |  |
| --- | --- | --- |
|  | Univerzitet u Nišu  Elektronski fakultet |  |

Seminarski rad

**Backup i restore MongoDB baze podataka**

**Mentor:** **Student:**

Doc. Dr Stanimirovic Aleksandar Kostić Jovana 1436

Sadržaj

[1. Uvod 3](#_Toc105011606)

[2. *Backup* i *Restore* 4](#_Toc105011607)

[2.1 Prednosti kreiranja rezervnih kopija 5](#_Toc105011608)

[2.2 Tipovi *Backup*-a 5](#_Toc105011609)

[3. Metode *MongoDB backup*-ovanja 7](#_Toc105011610)

[*3.1* *Backup-*ovanje korišćenjem *Atlas-*a 7](#_Toc105011611)

[3.2 *Backup*-ovanje *korišćenjem MongoDB Cloud Manager*-a i *Ops Manager–*a 7](#_Toc105011612)

[3.3 *Backup* kopiranjem osnovnih fajlova podataka 8](#_Toc105011613)

[3.3 *Backup* korišćenjem *mongodump*-a 8](#_Toc105011614)

[4. *Backup* i *restore* koriščenjem *Filesystem Snapshot*-ova 9](#_Toc105011615)

[4.1 Primer *backup-ovanja* i *restore-*ovanja korišćenjem *LVM*-a na *Linux*-u 9](#_Toc105011616)

[4.2 Primer *backup-ovanja* korišćenjem fajlova dnevnika na zasebnom logičkom disku ili bez vođenja dnevnika 12](#_Toc105011617)

[5. *Backup* i *restore* korišćenjem *MongoDB* alata 13](#_Toc105011618)

[5.1 *mongodump backup* 13](#_Toc105011619)

[5.2 *mongorestore restore* 14](#_Toc105011620)

[6. *Restore*-ovanje seta replika iz *MongoDB backup*-a 16](#_Toc105011621)

[6.1 *Restore*-ovanje baze podataka u jednom *node*-u seta replika 16](#_Toc105011622)

[6.2 Dodavanje članova skupu replika 17](#_Toc105011623)

[7. Proces *backup-ovanja* i *restore-*ovanja *sharded* klastera 18](#_Toc105011624)

[7.1 *Backup*-ovanje korišćenjem *snapshot*-ova fajl sistema 18](#_Toc105011625)

[7.2 Planiranje *backup*-ovanja *sharded* klastera 20](#_Toc105011626)

[7.3 Proces *restore*-ovanja *sharded* klastera 21](#_Toc105011627)

[8. Oporavak baze podataka nakon neočekivanog isključivanja 24](#_Toc105011628)

[Zaključak 25](#_Toc105011629)

[Literatura 26](#_Toc105011630)

# Uvod

Podaci su sredstvo koje organizacije koriste za različite procese obrade. Podaci prikupljeni od korisnika mogu se analizirati i koristiti za donošenje poslovnih odluka. Zbog toga, u slučaju da se ovi podaci izgube, to može rezultirati katastrofalnim posledicama, ne samo po organizaciju već i po korisnika. Organizacije koje dobijaju podatke od svojih kupaca, kao što su *online* prodavnice, osiguravajuće agencije, banke itd. Imaju visok rizik i ogromnu odgovornost da zaštite podatke svojih potrošača. Gubitak podataka može izazvati probleme u poslovnim procesima i zahtevati dodatan korak pri razvoju aplikacija koji će se baviti segmentom *backup*-a i *restore-*a.

Sa iznenadnim porastom propisa o privatnosti, organizacije imaju dodatnu odgovornost da obezbede podatke svojih potrošača i zaštite ih od gubitaka ili zloupotrebe. Zato je kreiranje *remote* rezervne kopije podataka industrijski standard u većini sektora.

Kako *MongoDB* pruža mogućnost skladištenja i obrade ogromne količine podataka, isto tako vodi računa i o njihovoj sigurnosti i konzistentnosti nudeći najrazličitije alate za kreiranje rezervnih kopija podataka lokalnih baza podataka, udaljenih baza podataka, skupova replika, klastera itd.

# *Backup* i *Restore*

*Backup* i *restore* predstavljaju upravo ono što njihovo ime kaže, rezervnu kopiju i oporavak podataka. To je proces kreiranja kopija podataka i zasebnog skladištenja, koji omogućava preuzimanje rezervnih podataka u slučaju gubitka istih. Nepostojanje načina za backup i restore može dovesti do trajnih gubitaka podataka, što osim samog gubitka podataka prouzrokuje i finansijske gubitke, nepoverenje korisnika i slične negativne posledice.



Slika 1 - *Backup*

Svrha *backup*-a je da se kreira kopija podataka koja se može oporaviti i slučaju gubitka primarnih podataka, koji mogu biti rezultat hardverskog ili softverskog kvara, oštećenja podataka uzrokovanih malicioznim napadima ili slučajnim brisanjem. Rezervne kopije omogućavaju vraćanje ranijih podataka kako bi se omogućio oporavak od neplaniranog gubitka. Čuvanje kopije na zasebnom uređaju je ključno za zaštitu od gubitka ili oštećenja podataka. Ovaj dodatni uređaj može biti jednostavan, kao što su eksterni hard disk, *USB* disk, kompaktni diska ili nešto kompleksnije, kao što je sistem za skladištenje na disku, *cluoud* kontejner za skladištenje itd. Alternativni uređaji mogu biti na istoj lokaciji gde i primarni podaci ili *remote.*

Za najbolje rezultate, rezervne kopije se prave redovno i konzistentno kako bi se smanjila količina podataka koja bi mogla da bude izgubljena između *backup*-ovanja. Što više vremena prođe između rezervnih kopija, to je veća mogućnost za gubitak prilikom oporavka. Čuvanje više kopija podataka pruža veću sigurnost i fleksibilnost za vraćanje na trenutak u kome podaci nisu oštećeni.

*Backup* podataka je u osnovi rešenje za čuvanje najrazličitijih vrsta resursa i ima sposobnost da se nosi sa bilo kojom vrstom situacije gubitka podataka. Najjednostavnije rečeno, *backup* servis se priprema za najgori scenario, dok *restore* pruža način oporavka od najgoreg scenarija.

# Prednosti kreiranja rezervnih kopija

Rezervna kopija podataka je u osnovi rešenje koje ima sposobnost da se nosi sa bilo kojom vrstom gubitaka podataka. Dodavši na to režim oporavka imamo veoma snažno oružje za manipulaciju podacima u globalu. U moru prednosti ovog pristupa izdvajaju se sledeće:

1. Sigurnost i zaštita podataka
2. Zaštita u slučaju pada bilo koje vrste sistema
3. Brz i jednostavan pristup podacima
4. Jednostavnost oporavka podataka
5. Pouzdane replikacije
6. Efikasno upravljanje podacima

# Tipovi *Backup*-a

Jedini način za rešavanje problema gubitaka podataka je postavljanje čvrste strategije rezervnih kopija. Optimalna *backup* strategija varira u zavisnosti od potreba svake organizacije. Tri najčešće korišćene strategije su :

* *Full backups*
* *Differential backups*
* *Incremental backups*

***Full backups***

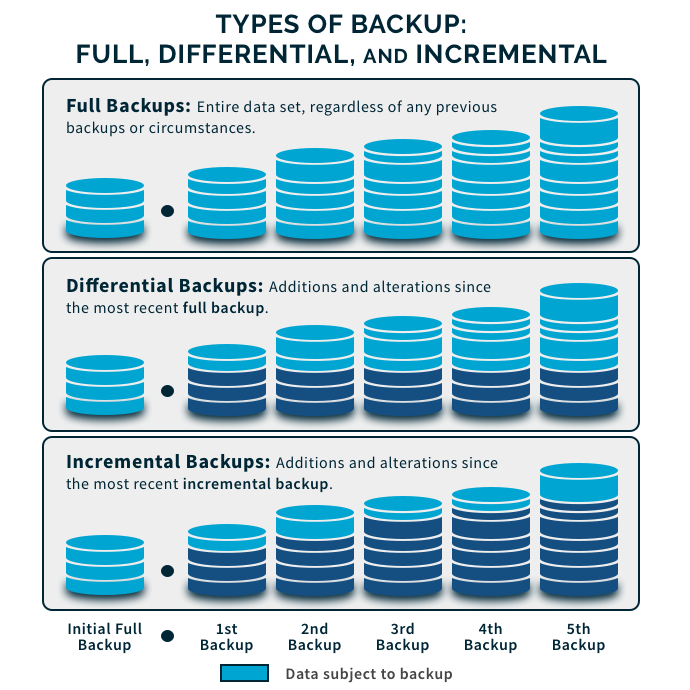
Potpuna rezervna kopija je najkompletnija vrsta *backup*-a gde se klonuju svi izabrani podaci, uključujući datoteke, direktorijume, *SaaS* aplikacije, hard diskove i još mnogo toga. Ključna karakteristika *full* *backup*-a je minimalno potrebno vreme za *restore–*ovanje podataka. Međutim, pošto se sve *backup*-uje u jednom potezu, potrebno je više vremena za samo kreiranje rezervne kopije u poređenju sa drugim tipovima *backup-*ovanja. Drugi uobičajeni problem je to što preopterećuje prostor za skladištenje. Zbog toga većina organizacija ima tendenciju da *full backup* prati i povremenom diferencijalnom ili inkrementalnom kopijom. Ovaj pristup smanjuje opterećenje prostora za skladištenje, povećavajući brzinu kreiranja rezervnih kopija.

***Differential backups***

Diferencijalna rezervna kopija predstavlja kombinaciju potpune i inkrementalne rezervne kopije. Ovaj tip *backup*-a uključuje kreiranje rezervne kopije podataka koji su kreirani ili promenjeni u odnosu na poslednju potpunu rezervnu kopiju. Drugim rečima, na samom početku se kreira *full backup,* a zatim se pokreću naknadne rezervne kopije kako bi se uključile sve u međuvremenu napravljene promene. Omogućava da se podaci *restore*-uju brže od *full backup*-ovanja jer zahteva samo dve komponente: početnu potpunu rezervnu kopiju i najnoviju diferencijalnu jer ona uključuje sve promene nakon prve potpune kopije.

***Incremental backup***

Prva rezervna kopija u inkrementalnom *backup*-u je potpuna rezervna kopija. Naredne rezervne kopije će čuvati samo promene koje su napravljene u prethodnom *backup*-ovanju. Inkrementalno *backup*-ovanje zahteva prostor samo za skladištenje promena, odnosno inkremenata, što omogućava munjevito kreiranje rezervnih kopija.



Slika 2 – Ilustracija strategija *backup*-ovanja

# Metode *MongoDB backup*-ovanja

U procesu razvoja kojeg god tipa aplikacije, gde imamo obradu i skladištanje podataka, bilo bi poželjno i sve češća je praksa posedovanje strategije za snimanje i vraćanje rezervnih kopija u slučaju gubitka podataka. *MongoDB* pruža četiri osnovne mogućnosti za kreiranje rezevnih kopija:

1. *Back Up with Atlas*
2. *Back Up with MongoDB Cloud Manager or Ops Manager*
3. *Back Up by Copying Underlying Data Files*
4. *Back Up with mongodump*

# *Backup-*ovanje korišćenjem *Atlas-*a

*MongoBD Atlas*, hostovani *MongoDB* servisni *cloud* pristup, nudi dve *fully-managed* metode *backup*-ovanja:

1. ***Cloud Backups*** koji u osnovi koriste *snapshot* funkcionalnost *cloud service provider deployment*-a nudeći robusne metode *backup*-ovanja. *Cloud Backup* nudi:

- *On-demand snapshots*  - pristup koji daje mogućnost kreiranja *snapshot*-a u bilo kom trenutku rada.

*- Continuous Cloud Backups* - pristup koji omogućava da isplaniramo *backup*-ovanje koje će se izržavati periodično.

1. ***Legacy Backups*** koji su već zastareli, ali kreiraju inkrementalne rezervne kopije podataka.

# *Backup*-ovanje *korišćenjem MongoDB Cloud Manager*-a i *Ops Manager–*a

***MongoDB Cloud Manager*** je hostovani *backup*, *monitoring* i servis automatizacije za *MondoDB.* Poržava *backup* i *restore* *MongoDB* setova replika i klastera koristeći grafički korisnički interfejs. Kontinualno kreira rezervne kopije čitajući *oplog* podatke iz *MongoDB deployment*-a. *MongoDB Cloud Manager* pravi *snapshot-*ove podataka u određenim intervalima vremena i samim tim može ponuditi mogućnost oporavka podataka, odnosno setova replika i klastera.

***Ops Manager*** je lokalno rešenje koje ima sličnu funkcionalnost kao *MongoDB Cloud Manager*. Sa *Ops Manager*-om korisnici mogu da instaliraju i pokreću u osnovi isti softver koji koristi *MongoDB Cloud Manager* na svojoj sopstvenoj infrastrukturi.

# 3.3 *Backup* kopiranjem osnovnih fajlova podataka

***Backup* korišćenjem *Filesystem Snapshot*-ova** omogućava kreiranje rezervne kopije *MongoDB deployment-*a tako što će se napraviti kopija osnovnih fajlova podataka *MongoDB*-a. Ako disk na kome *MongoDB* skladišti svoje podatke podržava *point-in-time snapshot*-ove, oni se mogu iskoristiti za *backup MongoDB* sistema u tačno određenom trenutku. *File system snapshot*-ovi su mogućnost koju pruža operativni sistem i nije karekteristična *MongoDB-*u. Mehanizam kreiranja *snapshot*-ova zavisi od osnovnog sistema skladištenja. Na primer na *Linux*-u *Logical Volume Manager* ima mogućnost kreiranja *snapshot*-ova, kao i Amazonov *EBS* sistem za skladištenje.

Da bi se dobio korektan *snapshot* pokrenutog *mongod* procesa, neophodno je imati omogućeno vođenje evidencije, odnosno dnevnika, koji mora biti na istom disku kao i ostali *MongoDB* fajlovi. Bez omogućenog evidentiranja, ne postoji garancija da će *snapshot*-ovi biti konzistentni i validni. Za konzistentan *snapshot* treba onemogućiti balanser i snimiti *snapshot* sa svakog *shard-*a (klastera), kao i sa servera za konfiguraciju u približno istom trenutku.

***Backup* uz pomoć *cp* ili *rsync*** pristup omogućava kreiranje rezervnih kopija ukoliko sistem ne podržava *snapshot*-ove, fajlovi se mogu kopirati direktno koristeći *cp, rsync* ili nekog sličnog alata. Pošto kopiranje više fajlova nije atomična operacija, svi upisi u bazu podataka moraju biti zaustavljeni pre samog kopiranja fajlova. U suprotnom, kopije neće biti validne.

Rezervne kopije kreirane kopiranjem osnovnih podataka ne podržavaju oporavak u trenutku za setove replika i teško je manipulisati njima u slučaju većih klastera. Pored toga, ove rezervne kopije su veće jer uključuju indekse i duplikate *padding*-a i fragmentacije osnovne memorije.

# *Backup* korišćenjem *mongodump*-a

***mongodump***čita podatke iz *MongoDB* baze podataka i kreira *BSON* datoteke visoke vernosti koje ***mongostore*** alat koristi za popunjavanje *MongoDB* baze podataka. mongodump i mongostore su jednostavni i efikasni alati za *backup* i *restore* malih *MongoDB* implementacija, ali nisu idealni za pravljenje rezervnih kopija većih sistema.

*mongodump* i *mongostore* mogu direktno da manipulišu osnovnim fajlovima. Podrazumevano, *mongodump* ne snima sadržaj lokalne baze podataka. *mongodump* snima samo dokumente u bazi podataka. Rezultujući *backup* je prostorno efikasan, ali *mongostore* ili *mongod*  moraju ponovo kreirti indekse nakon *restore*-ovanja podataka. Kada je povezan sa *MongoDB* instancom, *mongodump* može negativno uticati na performanse *mongod*-a. Ako su podaci veći od sistemske memorije, upiti če izbaciti radni skup iz memorije, uzrokujuli greške.

Aplikacije mogu da nastave da menjaju podatke dok *mongodump* snima izlazne podatke. Za setove replika, *mongodump* obezbeđuje opciju –*oplog* da bi uključio *oplog* unose u svoje izlaze, koji se javljaju tokom *mongodump* operacije. Ovo omogućava odgovarajućoj *mongostore* operaciji da ponovo reprodukuje snimljeni *oplog.* Da bi se *restore*-ovao *backup* kreiran sa *–oplog*, koristi se *mongostore* sa opcijom *–oplogReplay*.

Svakako za skupove replika najbolje je koristiti *MongoDB Cloud Manager* ili *Ops Manager*.

# *Backup* i *restore* koriščenjem *Filesystem Snapshot*-ova

Ovaj pristup predstavlja način za kreiranje rezervnih kopija *MongoDB* pomoću alata na nivou sistema, kao što je *LVM* na primer. Ovi *filesystem snapshot*-ovi, odnosno takozvane ,,*block-level”* *backup* metode, koriste alate na nivou sistema za kreiranje kopija uređaja koji drži fajlove podataka *MongoDB*-a. Ove metode se izvršavaju brzo i rade pouzdano, ali zahtevaju dodatnu konfiguraciju sistema mimo *MongoDB*-a.

*Snapshot*-ovi funkcionišu tako što kreiraju pokazivače između trenutnih podataka I podataka *snapshot*-a uređaja. Kako se podaci sa kojima radimo razlikuju od *snapshot*-a do *snapshot*-a, process kreiranja *snapshot*-a koristi strategiju *copy-on-write*. Kao rezultat toga, *snapshot* čuva samo promenjene podatke. Nakon kreiranja *snapshot*-a, *snapshot image* se *mount*-uje na *file system* i kopiraju se podaci sa *snapshot*-a. Dobijena rezervna kopija sadrži potpunu kopiju svih podataka. Baza podataka mora da bude validna kada se pravi *snapshot*, to znači da svi upisi koje prihvata baza podataka moraju u potpunosti biti upisani na disk, ili u dnevnik evidencije ili u fajlove podataka. Ako postoje zapisi koji nisu na disku, kada se pokrene process *backup*-ovanja, te promene neće biti uključene u *backup*. U slučaju *WiredTiger* mehanizma za skladištenje, fajlovi održavaju konzistentno stanje od poslednje kontrolne tačke. Kontorlne tačke se javljaju na svakih 2GB podataka ili svakog minuta.

*MongoDB* 3.2 je dodao podršku za rezervne kopije *MongoDB* instanci na nivou uređaja korišćenjem *WiredTiger* mehanizma za skladištenje, kada se fajlovi *MongoDB* instance i fajlovi dnevnika evidencije nalaze na zasebnim uređajima. Međutim, da bi se napravila koherentna rezervna kopija, baza podataka mora biti zaklučana i svi upisi moraju biti suspendovani tokom procesa *backup*-ovanja. Pre ove verzije, kreiranje rezervne kopije na nivou uređaja pomoću *WiredTiger* mehanizma zahtevalo je da fajlovi podataka i dnevnika budu na istom uređaju.

Ukoliko *mongod* instanca ima omogućeno vođenje evidencije, odnosno dnevnika, tada se može koristiti bilo koja vrsta *file system*-a ili *snapshot* alat na nivou uređaja/bloka za kreiranje *backup*-a. Ako se *backup*-uje na *Linux* sistemu, neophodno je kofigurisati system uz pomoć *LVM*-a da se disku omoguće paketi za mogućnost kreiranja *snapshot*-ova.

# 4.1 Primer *backup-ovanja* i *restore-*ovanja korišćenjem *LVM*-a na *Linux*-u

Ovaj primer ilustruje korake pri kreiranju rezervne kopije i oporavka podataka korišćenjem *Logical volume manager*-a na *Linux* sistemu. *LVM*  je softver koji apstrahuje *disk image* na osnovu fizičkih uređaja i pruža brojne manipulacije diskom kao i mogućnost kreiranja *snapshot*-ova korisnih za upravljanje sistemom.

1. **Kreiranje *snapshot*-a**

Počevši od *MongoDB* 3.2, u svrhu kreiranja rezervnih kopija *MongoDB* instanci na nivou uređaja pomoću *WiredTiger-*a, fajlovi sa podacima i sam dnevnik evidencije ne moraju se nalaziti na istom uređaju. Prvi korak u kreiranju ovakvih kopija je kreiranje *snapshot*-ova.

Za kreiranje *snapshot*-a korišćenjem *LVM*-a koristi se sledeća komanda:

  
Slika 3 – Naredba za kreiranje *snapshot*-a

Ovom naredbom se kreira *LVM snapshot* *MondoDB* uređaja, koristeći opciju *–snapshot*, pod nazivom *mdb-snap01*. Ovaj primer *snapshot*-a se nalazi na lokaciji */dev/vg0/mongodb*, čime se i naglašava da pripada *vg0* grupi uređaja. Lokacija i putanja do grupa uređaja mogu se neznatno razlikovati u zavisnosti od kofiguracije *LVM*-a operatvinog sistema. Veličina *snapshot*-a je ograničena na 100MB zbog parametra veličine *–size 100M*. Ova veličina se ne odnosi na ukupnu količinu podataka na disku, već predstavlja veličinu razlike između trenutnog stanja */dev/vg0/mongodb* i kreiranja *snapshot*-a */dev/vg0/mongodb/mdb-snap01.*

Podaci se mogu *restore*-ovati direktno iz *snapshot*-a u bilo kom trenutku ili kreiranjem novog logičkog diska i *restore*-ovanjem iz kreiranog *snapshot*-a na *image* alternativu. Medjutim, iako su *snapshot*-ovi odlično rešenje za brzo kreiranje rezervnih kopija visokog kvaliteta, nisu idealni kao format za čuvanje rezervnih kopija podataka. Obično zavise i nalaze se na istoj infrastrukturi za skladištenje kao i originalni disk *image*. Zbog toga je ključno da se *snapshot*-ovi arhiviraju i skladište na drugom mestu.

1. **Arhiviranje *snapshot*-a**

Nakon što je kreiran *snapshot*, neophodno je *mount*-ovati ga i kopirati podatke na zasebnom skladištu. Sistem može pokušati da kompresuje *backup images* ako se prebacuju *offline*. Alternativno, može se napraviti kopija *snapshot image* na nivou bloka, na sledeći način:

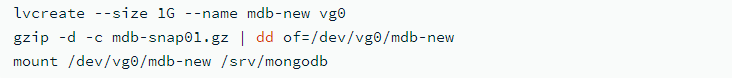
  
Slika 4 – Naredbe za unmountovanje uređaja i arhiviranje *snapshot*-a

Prva naredba garantuje da */dev/vg0/mdb-snap01* uređaj nije *mount*-ovan. Nikada ne treba kreirati kopiju na nivou bloka *filesystem*-a ili *filesystem snapshot* dok je *mount*-ovan.

Druga naredba vrši kopiranje na nivou bloka celog *snapshot image*-a koristeći *dd* komandu i kompresuje rezultat u zipovani fajl u trenutnom radnom direktorijumu.

1. ***Restore*-ovanje *snapshot*-a**

*Restore*-ovanje *snapshot*-a kreiranog koristeći *LVM* može se postići sekvencom sledećih naredbi:

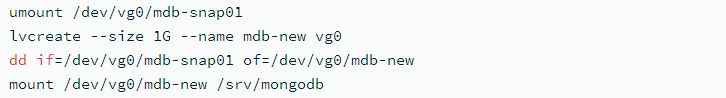
  
Slika 5 – Sekvenca naredbi za proces *restore*-ovanja *snapshot*-a

Prvom naredbom se kreira novi logički disk pod nazivom *mdb-new*, u */dev/vg0* grupi logičkih diskova. Putanja do ovako kreiranog diska biće */dev/vg0/mdb-new*. Druga naredba predstavlja proces dekompresovanja i smeštanja na *mdb-new* disk *image*, dok poslednja naredba *mount*-uje disk *image* na /srv/mongodb direktorijum.

*Restore*-ovani *snapshot* će imati zastareli *mongo.lock* fajl. Ako se ovaj fajl ne ukloni iz *snapshot*-a, *MongoDB* na osnovu ovog zastarelog *lock* fajla može zaključiti da je došlo do neočekivanog gašenja. Ako se radi sa omogućenim *storage.journal.enabled* i ne korisiti se *db.fsyncLock()* ne moramo ukljanjati ovaj fajl, u suprotnom moramo.

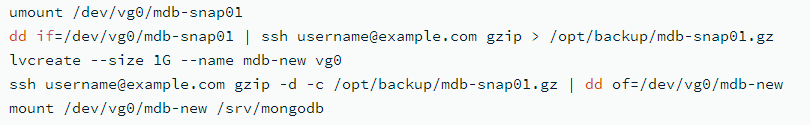
1. ***Restore*-ovanje direktno iz *snapshot*-a**

Na način sličan prethodnom, može se izvršiti *restore*-ovanje *backup*-a bez upisa u kompresovani *gz* fajl, korišćenjem sledeće sekvence naredbi:

  
Slika 6 - Sekvenca naredbi za proces *restore*-ovanja *snapshot*-a bez upisa u zip fajl

1. ***Remote backup storage***

Postoji i način implementacije rezervne kopije van sistema, odnosno kreiranje udaljene kopije koristeći kombinovane procese i *SSH*. Ova sekvenca naredbi identična je prethodno objašnjenim, osim što arhivira i kompresuje rezervnu kopiju na udaljenom sistemu koristeći *SSH*.

  
Slika 7 – Sekvenca naredbi za *remote backup storage*

# Primer *backup-ovanja* korišćenjem fajlova dnevnika na zasebnom logičkom disku ili bez vođenja dnevnika

Počevši od *MongoDB* verzije 3.2, u svrhu pravljenja rezervnih kopija *MongoDB* instanci, na nivou logičkog diska koriščenjem *WiredTiger*-a, fajlovi podataka i fajlovi dnevnika ne moraju nužno biti na istom logičkom disku. Međutim, baza podataka mora biti zaključana i svi upisi moraju biti suspendovani tokom procesa kreiranja rezervne kopije, kako bi se osigurala konzistentnost rezervne kopije.

U slučaju da *mongod* instanca funkcioniše bez vođenja dnevnika ili su fajlovi dnevnika na zasebnom logičkom disku, svi upisi na disk moraju biti *flush*-ovani, a sama baza podataka zaključana da bi se sprečio upis tokom procesa kreiranja rezervne kopije. Ako je u pitanju konfiguracija seta replika, onda za *backup* treba koristiti sekundar koji ne prima čitanja.

Postupak kreiranje rezervne kopije na ovakav način ukratko se može ilustrovati ovako:

1. ***Flush*-ovanje upisa na disk i zaključavanje baze podataka**

*Flush*-ovanje upisa na disk kao i zaključavanje same baze podataka možemo postići sledećom metodom koju treba izvršiti u *mongosh:*

Slika 8 – Metoda *flush*-ovanja upisa i *lock*-ovanja baze podataka

1. **Proces *backup*-ovanja**

Sam proces kreiranja rezervne kopije obavlja se na isti način kao što je opisano u prethodnom primeru za kreiranje *snapshot*-ova.

1. **Otključavanje baze podataka**

Nakon što je *snapshot* kreiran, treba otključati bazu podataka sledećom komandom koju ćemo takođe izvršiti u *mongosh*:

  
Slika 9 – Metoda za otključavanje baze podataka

# 5. *Backup* i *restore* korišćenjem *MongoDB* alata

*mongodump* i *mongorestore* alati rade koristeći *BSON data dumps* i korisni su u slučaju kreiranja rezervnih kopija aplikacija manjeg obima. Ove alate za proces *backup*-ovanja možemo koristiti u slučaju da druge metode, poput *MongoDB Cloud Manager*-a ili *snapshot-*ova *file system*-a, nisu dostupne.

# 5.1 *mongodump backup*

Za pokretanja *mongodump*-a nad *MongoDB* sistemom koji ima omogućenu kontrolu prostupa, moramo posedovati privilegije koje omogućavaju *find action* za svaku bazu podataka za *backup*-ovanje. Ove privilegije neophodne su za proces kreiranja rezervnih kopija bilo koje baze podataka.

*mongodump* kreira rezervnu kopiju podataka tako što se povezuje sa aktivnim *mongod*-om. Ovaj alat pruža mogućnost kreiranja rezervne kopije celog server, baze podataka ili kolekcije, a isto tako se može koristiti i upit za pravljenje rezervne kopije samo dela kolekcije.

**Osnovne *mongodump* operacije**

Kada se *mongodump* pokrene bez ikakvih argumenata, komanda se povezuje sa *MongoDB* instancom na lokalnom sistemu, *localhost*, na portu 27017 i kreira rezervnu kopiju baze podataka pod nazivom *dump/* u trenutnom direktorijumu. Takođe, koristeći ovu naredbu ali sa argumentima, može se specificirati *–host* i *–port* *MongoDB* instance na koju bi *mongodump* treba da se poveže.

  
Slika 10 – Komanda za kreiranje rezervne kopije uz pomoć *mongodump-*a

Ovom naredbom *mongodump* će kreirati *BSON* fajl koji će sadržati kopiju podataka kojima se može pristupiti uz pomoć *mongod*-a koji sluša na portu 27017 *mogodb.example.net* hostu.

Za specificiranje drugog *output* direktorijuma, postoji opcija *–out* ili *–o*:

  
Slika 11 - Komanda za kreiranje rezervne kopije uz pomoć *mongodump-*a sa specificiranim *output* direktorijumom

Da bi se ograničila količina podataka uključenih u *dump* baze podataka, postoje opcije *–db* i *–collection* kao opcije *mongodump*-a.

  
Slika 12 – za kreiranje rezervne kopije uz pomoć *mongodump-*a sa ograničenom količinom podaataka   
Ovom operacijom kreira se *dump* kolekcije pod nazivom *myCollection* iz *test* baze podataka u poddirektorijumu *dump/* trenutnog radnog direktorijuma.

*mongodump* prepisuje *output* fajlove ako oni postoje u folderu sa rezervnim kopijama, zato pre nego što se ova komanda pokrene ponovo, treba proveriti da li su i dalje potrebni postojeći fajlovi ili rešiti jednostavnim preimenovanjem fajlova ili foldera.

Nad svakom *mongodump* naredbom mogu se specificirati *username* i *password* za autentifikaciju baze podataka

**Kreiranje rezervnih kopija korišćenjem *Oplog*-ova**

*mongodump* sa opcijom *–oplog* prikuplja *oplog* ulaze i omogućava pravljenje rezervne kopije podataka nad trenutnim takozvanim *live* podacima. Kada se kasnije *restore-*uje baza podataka iz *backup*-a, baza podataka biće ista kao što je bila kada je proces pravljenja rezervne kopije završen.

Sa *–oplog* opcijom, *mongodump* kopira sve podatke iz izvorne baze podataka, kao i sve *oplog* unose od početka do kraja procesa kreiranja rezevne kopije. Ova operacija, u kombinaciji sa *mongorestore –oplogReplay,* omogućava *restore*-ovanje *backup*-a koji zapravo uključuje promene do trenutka kada je *mongodump* kompletirao proces kreiranja *dump* fajla.

# 5.2 *mongorestore restore*

*mongorestore* vraća binarnu rezervnu kopiju koja je kreirana *mongodump*-om. *Mogorestore* podrrazumevano traži rezervnu kopiju baze podataka u *dump/* direktorijumu. Ovaj alat *restore*-uje podatke tako što se direktno povezuje na pokrenuti *mongod*  i može da vrati celu rezervnu kopiju baze podataka ili neki njen podskup.

**Osnovne *mongorestore* operacije**

Za korišćenje *monogorestore*-a neophodno je povezati se na aktivni *mongod* korišćenjem sledeće komande:

  
Slika 13 – Komanda za povezivanje *mongorestore*-a na aktivni *mongod*

  
Slika 14 – Primer komande za povezivanje *mongorestore*-a na aktivni *mongod*

Ovim primerom komande *mongorestore* importuje *backup* baze podataka u folder *dump-2012-10-25* na *mongod* instanci koja je pokrenuta na *localhost*-u na podrazumevanom portu 27017. Ako želimo da definišemo drugi *host* i *port*, to možemo učiniti na sledeči način:

  
Slika 15 – Primer komande za povezivanje *mongorestore-*a na udaljeni *mongod*

Ako se vrši *restore*-ovanje instance koja nameće kontrolu pristupa, treba uzeti u obzir i *–username* i *–authentivationDatabase* opcije, dok opciju *–pasword* treba izostaviti jer će je sam *prompt* zatražiti.

***Restore*-ovanje rezervnih kopija korišćenjem *Oplog*-ova**

U slučaju kada je *dump* baze podataka kreiran korišćenjem *–oplog* opcije, da bi se obezbedio *point-in-time snapshot*, treba izvršiti *mongorestore* sa opcijom *–oplogReplay*.

Takođe, *mongorestore –objcheck* opcija obezbeđuje mogućnost provere integriteta objekata pri upisu u bazu. Još jedna mogućnost koju pruža ovaj alat je *mongorestore –drop* za izbacivanje svake kolekcije iz baze podataka pre *restore*-ovanja iz rezervnih kopija.

# 6. *Restore*-ovanje seta replika iz *MongoDB backup*-a

Ovaj pristup omogućava proces preuzimanja *MongoDB* podataka i *restore*-ovanje tih podataka u novi set replika. Najčešće se koristi za pokretanje testnih implementacija iz *production* *backup*-ova ili kao deo oporavka podataka.

# 6.1 *Restore*-ovanje baze podataka u jednom *node*-u seta replika

Ovaj proces može se podeliti u pet osnovna koraka:

**Pribavljanje *MongoDB*** ***backup* fajlova**

Fajlovi rezervne kopije mogu se pribaviti iz *snapshot*-ova *filesystem*-a. *MongoDB Cloud Manager* generiše *MongoDB* fajlove baze podataka za već skladištene I trenutne *snapshot-*ove.

***Drop*-ovanje lokalne baze podataka ako ona postoji u *backup*-u**

Ako se process *restore*-ovanja vrši iz *filesystem backup*-a ili bilo kog *backup*-a sa lokalnom bazom podataka, neophodno je pre svega *drop*-ovati lokalnu bazu. Za potrebe ovog procesa neophodno je pokrenuti samostalni *mongod* koristeći *backup* fajlove kao putanju podataka:

  
Slika 16 – Pokretanje samostalnog *mongod*-a za potrebe procesa *drop*-ovanja

Sledeći korak je samo *drop*-ovanje lokalne baze podataka tako što se *mongosh* poveže na *mongod* instancu i *drop*-uje baza podataka.

  
Slika 17 – Sekvenca komandi za povezivanje *mongosh* na *mongod*  i samo *drop*-ovanje

***Start*-ovanje novog *single-node* seta replika**

Pokretanje *mongod* instance kao novi *single-node* skup replika može se postići tako što se navede putanja do *backup* fajlova opcijom *–dbpath* i setovati ime skupa replika uz pomoć opcije *–replSet*. Za konfigurisanje server skupa replika koristi se opcija *–configsvr*.

  
Slika 18 – Komanda za pokretanje *mongod* instance kao novi *single-node* skup replika

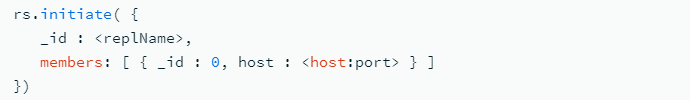
**Povezivanje *mongosh* na *mongod* instance**

Sa iste mašine na kojoj je pokrenut jedan od *mongod*-a pokrenuti *mongosh.* Kako bi se ostvatila konekcija na *mongod* koji podrazumevano sluša na portu 27017 jednostavno izvršiti naredbu *mongosh.*

U zavisnosti od trenutne putanje, možda će biti potrebno navesti putanju do binarnog *mongosh* fajla. Takođe, ako *mongod* ne osluškuje na podrazumevanom portu, neophodno je navesti ga uz pomoć opcije *–port* za *mongosh.*

**Inicijalizovanje novog skupa replika**

Metodu *rs.initiate()* izvršiti samo nad jednim članom skupa replika:

  
Slika 19 – Inicijalizovanje novog skupa replika

*MongoDB* inicijalizuje skup koji se sastoji od trenutnog člana i koristi podrazumevanu kofiguraciju skupa replika.

# 6.2 Dodavanje članova skupu replika

*MongoDB* pruža dve mogućnosti za *restore*-ovanje sekundarnih članova skupa replika:

1. Ručno kopiranje fajlova baze podataka u svaki direktorijum podataka
2. Inicijalna sinhronizacija koja automatski distribuira podatke

U prvom slučaju, neophodno je ugasiti *mongod* instancu koju smo *restore-*ovali koristeći *–shutdown* ili *db.shutdownServer()* da bi se obezbedilo bezbedno gašenje. Nakon toga, kopirati folder primarnih podatana u *dbPath* ostalih članova replika. Pokrenuti *mongod* instancu koju je *restore*-ovana i dodati sekundare u skupu replika. U *mongosh* sesiji koja je povezana na primar, dodati sekundare skupu replika korišćenjem metode *rs.add().*

U slučaju inicijalne sinhronizacije, na samom početku neophodno je proveriti za svaki član skupa replika da ima *storage.dbPath* ili *–dbpath* za /data/db i da su ti direktorijumi prazni. Pokrenuti svaki član seta replike i na samom kraju dodati redom svaki član seta replike tako što se ostvari konekcija sa primarom korišćenjem *mongo shell-*a i dodati svaki sekundar metodom *rs.add().* Kada se doda član skupu replika, *Initial sync* kopira podatke sa primara na novi član.

# 7. Proces *backup-ovanja* i *restore-*ovanja *sharded* klastera

# 7.1 *Backup*-ovanje korišćenjem *snapshot*-ova fajl sistema

Ovaj pristup kreira *backup* svih komponenti *sharded* klastera koristeći *snapshot*-ove fajl sistema da bi kreirali kopiju *mongod* instance. Postupak ovakvog načina kreiranja rezervne kopije možemo podeliti u 6 osnovnik koraka:

**Onemogućavanje rada *balancer-a***

Proces balansiranja je odgovoran za ravnomernu redistribuciju delova *sharded* kolekcije. Podrazumevano proces balansiranja je uvek omogućen. Pre svega, neophodno je povezati *mongosh* na klaster *mongos* instance, a zatim metodom *sh.stopBalancer()* zaustaviti balanser. Ako je proces balansiranja u toku, operacija čeka da se balansiranje završi pre nego što zaustavi balanser.

  
Slika 20 – Komanda za zaustavljanje balansera

Počevši od *MongoDB* verzije 4.2, ovom naredbom se takođe onemogućuje *auto-spliting* *sharded* klastera.

**Zaključavanje jednog sekundarnog člana svakog skupa replika, ako je to neophodno**

U slučaju da sekundaru nije omogućeno vođenje dnevnika ili su njegovi fajlovi dnevnika i podataka na različitim logičkim diskovima, neophodno je zaključati sekundar *mongod* instance pre kreiranja rezervne kopije podataka, u suprotnom, ovaj korak se može preskočiti. Takođe, treba proveriti da li *oplog* ima dovoljan kapacitet da omogući ovim sekundarima da sustignu stanje primara nakon završetka procesa *backup*-ovanja.

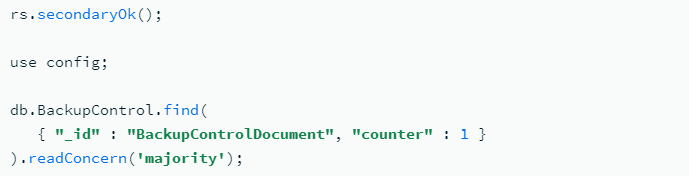
U slučaju da je zaključavanje neophodno, treba zaključati jedan sekundar svakog *shard*-a i jedan sekundar skupa replika koniguracionog servera. U procesu zaključavanja sekundara za svaki skup replika u *sharded* klasteru prvenstveno treba proveriti da li član ima replikovane podatke do neke kontrolne tačke. Za ovu proveru treba povezati *mongosh* sa *shard* primarom i izvršiti operaciju upisa na sledeći način:

  
Slika 21 – Operacija upisa sa *majority write concern-*om nad kontrolnom kolekcijom

Ova operacija bi kao rezultat izvršenja trebalo da vrati modifikovani ili dodati kontrolni dokument kao u nastavku:

  
Slika 22 – Rezultujući kontrolni dokument

Povezivanjem *mongosh* na *shard* sekundar možemo ga zaključati i kreirati upit za pribavljanje kontrolnog dokumenta koristeći *db.collection.find()*:

  
Slika 23 – Metoda za pronalaženje kontrolnog dolkumenta na sekundaru

Ako sekundar sadrži najnoviji kontrolni dokument, bezbedno je zaključati taj član. U suprotnom, treba sačekati dok taj član ne pribavi najnoviji kontrolni dokument ili izabrati neki drugi član koji sadrži ovakav dokument. Zaključavanje sekundarnog člana vrši se na sledeći način:

  
Slika 24 – Metoda za zaključavanje članova seta replika

Pošto proces zaključavanja podrazumeva i zaključavanje sekundara seta replika konfiguracionog servera to bi bio sledeći po istom principu.

***Backup*-ovanje jednog od konfiguracionih servera**

*Backup*-ovanje konfiguracionog servera podrazumeva kreiranje rezervne kopije metapodataka *sharded* klastera. Neophodno je *backup*-ovati samo jedan konfiguracioni server, jer svi oni drže iste podatke. Za kreiranje *filesystem* *snapshot*-a konfiguracionog servera koristi se standardni pristup kreiranja *snapshot*-a.

***Backup*-ovanje člana skupa replika za svaki *shard***

U slučaju da je ovaj član zaključan, izvršiti ovaj korak iako je zaključan. *Backup*-ovanje se može izvršiti paralelno za sve *shard*-ove, kreiranjem *snapshot*-ova za svaki od njih na standardni način.

**Otključavanje svih zaklučanih članova seta replika**

Za otključavanje članova seta replika u *mongosh* iskoristiti metodu:

  
Slika 25 – Metoda za otključavanje članova seta replika

***Omogućavanje procesa rada balansera***

Za ponovno omogučavanje rada balansera, povezati *mongosh* na *mongos* i izvršiti metodu:

  
Slika 26 – Metoda za startovanje balansera

# 7.2 Planiranje *backup*-ovanja *sharded* klastera

U slučaju automatskog rasporeda *backup*-ovanja, postoji mogućnost da se sve operacije balansiranja onemoguće na određeno vreme. Na primer:

  
Slika 27 – Konfiguracija balansera za planirano *backup*-ovanje

Ovom metodom konfiguriše se balanser da radi između 06:00 i 23:00 prema serverskom vremenu. Van ovog vremena možemo isplanirati ili zakazati izvršenje operacije kreiranja rezervne kopije, ali moramo biti sigurni da će se ova operacija završiti pre nego što balanser krene ponovo sa radom.

# 7.3 Proces *restore*-ovanja *sharded* klastera

Proces *restore-*ovanja vraća podatke *sharded* klastera na osnovu postojećih *backup* *snapshot*-ova. Da bi ovaj process bio uspešno obavljen izvorni i ciljani klaster moraju imati isti broj *shard*-ova. *Restore*-ovanje možemo predstaviti u nekoliko koraka:

1. **Pregled konfiguracije skupa replika**

Ovaj korak nije neophodan, opcioni je i omogućava pregled kofiguracije uz pomoć metode *rs.conf()* uz pomoć koje možemo videti konfiguracioni document replike.

1. **Priprema ciljnog *host*-a za restauraciju**

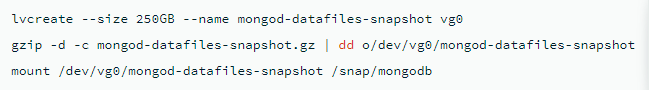
Sledeći korak je priprema *host*-a na kome će se *restore*-ovati što uključuje proveru zahtevane dostupne memorije za skladištenje *restore*-ovanih podataka, postojanje bar jednog *LVM* upravljanog logičkog diska i dovoljno memorije za podatke ektraktovane iz *snapshot*-a. Još jedan od zahteva je da izvorni i ciljni host imaju istu verziju *MongoDB-a.*

Ako se vrši *restore* postojećeg klastera, treba prekinuti izvršavanje *mongod*  i *mongos* procesa na ciljnom *host*-u. U slučaju *host*-ova na kojima se izvršava *mongos*, povezati *mongo shell* na *mongos* i izvršiti komandu *db.shutdownServer()* sa admin baze podataka.

Takođe, još neke od priprema su kreiranje direktorijuma na ciljnom *host*-u za smeštanje *restore-*ovanih podataka, kao I direktorijuma za *mongod* *log* fajlove. A poslednji korak ove pripreme ciljnog *host*-a je konfiguracionog fajla.

1. ***Restore*-ovanje konfiguracionog servera skupa replika**

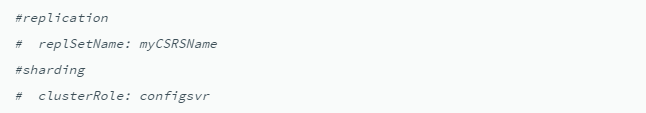
Prvi korak samog *restore*-ovanja je *mount*-ovanje *LVM snapshot*-a na ciljnani *host*:

  
Slika 28 – Proces *mount*-ovanja *LVM snapshot*-a

Sledeći korak je kopiranje *mongod* fajlova iz *mount-*ovanog *snapshot*-a u kreirani direktorijum na ciljanom hostu:

  
Slika 29 – Kopiranje podataka iz *mount-*ovanog *snapshot*-a

Izbaciti ili zakomentarisati sledeće delove konfiguracionog fajla:

  
Slika 30 – Izgled konfiguracionog fajla

Zatim, startovati *mogod* korišćenjem konfiguracionog fajla, navodeći njegovu apsolutnu putanju:

  
Slika 31 – Pokretanje *mogod-*a korišćenjem konfiguracionog fajla

Sledeći korak je *drop*-ovanje lokalne baze podataka:

  
Slika 32 – *Drop*-ovanje lokalne baze podataka

Pre samog procesa inicijalizacije novog skupa replika, neophodno je *restart*-ovati *mongod* kao novi *single-node* skup replika. Prekinuti process izvršavanja *mongod*-a i dodati sledeće opcije u konfiguracioni fajl:

  
Slika 33 – Izmene u konfiguracionom fajlu

Startovati *mongod* sa izmenjenim konfiguracionim fajlom:

  
Slika 34 – Ponovno pokretanje *mongod*-a sa izmenjenom konfiguracijom

Konačno, dolazimo do koraka samog inicijalizovanja novog skupa replika sledećom komandom:

  
Slika 35 – Metoda za instanciranje skupa replika

Nakon kreiranja novog skupa replika, postoji mogučnost dodavanja članova skupa replika. Za svaki clan skupa replika u *CSRS*-u, treba pokrenuti *mongod* na njegovoj host mašini. Tada, treba povezati *mongo shell* na primarni član skupa replike. Sa primara možemo koristiti metodu *rs.add()* za dodavanje svakog člana skupa replika:

  
Slika 36 – Metoda za dodavanje članova skupa replika

Ukoliko želimo da dodamo član koji ima specifičnu konfiguraciju, možemo proslediti konfiguracioni document kao parameter:

  
Slika 37 – Metoda za dodavanje članova skupa replika sa specifičnom konfiguracijom

Svaki novododati član vrši *initial sync* da bi se sinhronizovao sa primarom. Da bi se identifikovao primar može se iskoristiti funkcija *rs.status()*, jer se metoda *rs.add()* može pozvati samo sa primara.

Za dodatnu rekonfiguraciju metoda *rs.reconfig()* ažurira konfiguraciju skupa replika na osnovu konfiguracionog dokumenta koji joj je prosleđen kao parameter.

1. ***Restore*-ovanje svakog *shard* skupa replika**

Postupak *restore*-ovanja svakog *shard* skupa replika je gotovo identičan prethodno opisanom postupku *restore*-ovanja skupa replika konfiguracionog servera.

# 8. Oporavak baze podataka nakon neočekivanog isključivanja

Oštećenje ili gubitak podatka na nivou diska ili fajlova koji nedostaju, mogu sprečiti pokretanje *mongod* instance, a fajlovi vođenja dnevnika nedovoljni za automatski oporavak. U takvim slučajevima, *dbPath* sadrži *mongod.lock* fajl koji nije prazan. U tom slučaju naredbom *mongod –repair* , počev od *MongoDB* verzije 4.4, postiže se:

* *Rebuild*-ovanje svih indeksa kolekcija sa jednim ili više nekonzistentnih indeksa
* Odbacivanje oštećenih podataka
* Kreiranje praznih/*stub* fajlova za podatke/metpodatke koji nedostaju

Postupak oporavka podataka ogleda se u dva koraka:

**Kreiranje *backup*-a podataka**

Kreiranje rezervne kopije fajlova podataka obavlja se u *–dbpath.*

**Startovanje *mongod* sa opcijom *–repair***

Oporavljanje podataka postiže se startovanjem *mongod* instance sa opcijom *–repair* izvršavanjem sledeće naredbe:

  
Slika 28 – Naredba za oporavak podataka nakon neočekivanog gašenja

Nakon uspešnog izvršavanja ove naredbe, *dbpath* treba da sadrži oporavljene podatke i prazan *mongod.lock* fajl.

# Zaključak

Kreiranje rezervnih kopija podataka kao i oporavak podataka na osnovu ovih kopija zauzima značajno mesto u razvoju bilo koje vrste aplikacija, od malih korisničkih aplikacija pa sve do ogromnih komercijalnih. Bilo kako bilo, uloga i značaj ovih procesa je neprikosnoven. Kako količina podataka u globalnom smislu raste, što podataka za skladištenje što za obradu i analizu, svakako nikome nije u interesu da te podatke izgubi, već da ih na neki način osigura.

*MongoDB* predstavlja ogroman sistem koji može da smesti i obradi nezamislivo veliku količinu najraznovrsnijih podataka, s tim da korišćenjem *backup* i *restore* metoda pruža veliku sigurnost i otpornost sistema. Kako je *MongoDB* glavni predstavnik *document oriented* baza podataka i glavna alternativa tabelarnom pristupu, tako i sam pruža dosta različitih metoda za kreiranje kako lokalnih tako i udaljenih rezervnih kopija podataka i metoda za *restore*-ovanje.

U procesu razvoja aplikacija u globalu jedan od glavnih koraka u domenu obrade podataka trebalo bi da se bavi čuvanjem i zaštitom od gubitaka podataka, jer oni mogu biti uzrokovani na veoma različite načine, kao što su zastoj hardvera, oštećenje softvera, maliciozni napadi, iznenadno gašenje i mnogi drugi. Zato su metode za kreiranje rezervnih kopija kao i metode za restauraciju podataka na osnovu tih kopija veoma važne u svakoj vrsti razvoja aplikacija i obradi podataka.

# Literatura

<https://spanning.com/blog/types-of-backup-understanding-full-differential-incremental-backup/>

<https://www.advantage.tech/service/advantage-backup-and-recovery>

<http://www.rossbackup.com/blog/why-backing-up-is-essential-the-top-five-benefits-to-data-backup.htm>

<https://rewind.com/blog/advantages-data-backup-recovery/>

<https://www.mongodb.com>