Seminarski rad

Backup/Restore kod MongoDB-a

Sistemi baza podataka

Stefan Mladenov 1205

# 

[**1. Uvod**](#_tl1v1kxm99sl) **3**

[**2. Interno skladištenje podataka kod MongoDB-a**](#_evjsdcjfdxjo) **4**

[**3. Kreiranje bekapa**](#_h0d2mzdc8490) **5**

[3.1 Logički](#_di6oktowfa1r) 6

[3.1.1 Mongodump/Mongorestore](#_1iuqbvot3q7c) 6

[3.1.2 Mongoexport/Mongoimport](#_el2ny6aka22q) 8

[3.2 Fizički](#_nhlw43u6hhmj) 11

[**4. Zaključak**](#_h5angz280nj8) **14**

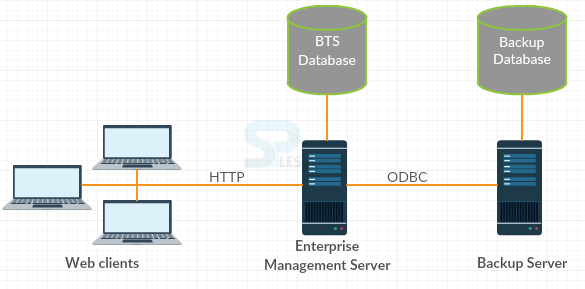
[**5. Reference**](#_zefgu78cyvus) **15**

# 

# 1. Uvod

Baze podataka danas predstavljaju jednu od glavnih komponenti sistema koji se bave određenom obradom nekih informacija. MongoDB predstavlja kros-platformsku *Document-Oriented* bazu napisanu u C++ jeziku, prvi put objavljenu 2009.godine. Namenjena je za skladištenje podataka u JSON formatu, koji se interno skladište u BSON format. Jedna je od danas najpopularnijih NoSQL baza zbog skalabilnosti, pouzdanosti i lakoće korišćenja.

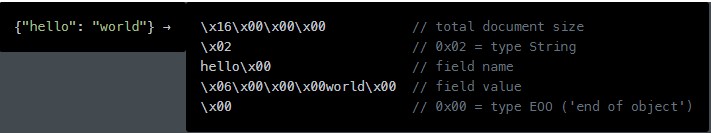
Podaci koji se čuvaju u bazi se mogu koristiti u okviru razlilčitih aplikacija gde je jako bitno da podaci budu dostupni, sinhronizovani i konzistentni što je više moguće. Za mnoge sisteme, gubitak podataka, nedostupnost istih ili narušavanje njihove verodostojnosti bi predstavljao pravu katastrofu ukoliko se radi o velikoj količini podataka, prikupljanih i čuvanih duži vremenski period. Kako bi se izbegao ovakav scenario usled nekih nepredviđenih okolnosti kao što su otkaz memorijskog medijuma, zlonamerni proboj spolja, neautorizovani pristup podacima ili neka prirodna katastrofa koja može dovesti do prestanka napajanja mašine na kojoj se baza nalazi, dobru praksu predstavlja kreiranje rezervnih kopija podataka takozvanih *Backup*-ova, gde se oporavak od otkaza svodi na vraćanje podataka u neko od pređašnjih stanja.



Slika 1. Backup server

# 2. Interno skladištenje podataka kod MongoDB-a

MongoDB je kreiran sa ciljem da bude jedna od najkorišćenijih baza podataka modernih aplikacija. S obzirom na ekspanziju web-a, JSON(engl. *JavaScript Object Notation*) kao jedan od najkorišćenijih formata se nametnuo kao rešenje za prezentaciju podataka u okviru dokumenata. On je baziran na tekstu, a kako je parsovanje teksta sporo i može zauzeti i više prostora nego što je potrebno zbog čitljivosti, došlo se na ideju da se kreira optimizovaniji format za čuvanje informacija, BSON(engl. *Binary JSON*). Binarno enkodovan JSON, kreiran sa namenom da optimizuje proces parsiranja i omogući veću fleksibilnost, pored tipova koje i sam JSON podržava, omogućava korišćenje *Date* i *BinData* tipova. Takođe postoji podrška za veći broj numerčkih tipova poput *decimal128, longs* ili *double* čime se omogućavaju različite matematičke operacije nad podacima. Uz pomoć ovog formata, ubrzava se proces pretrage dokumenata s obzirom da podržava neke od glavnih operacija sa nizovima, čime se podstiče korišćenje indeksa.



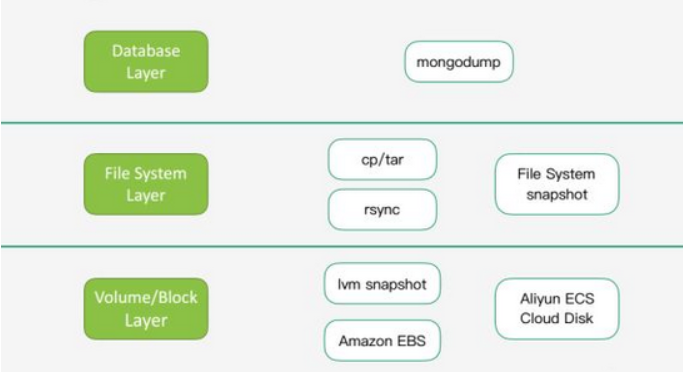
Slika 2. BSON

# 

# 3. Kreiranje bekapa

Importovanje i eksportovanje baze podataka predstavlja rad sa podacima koji su kompatibilni u okviru različitih softverskih proizvoda. Sa druge strane, *backup* i *restore* predstavljaju procese kreiranja ili manipulacije nad binarnim podacima specifičnim za samu bazu, koji pored tabela ili dokumenata u slučaju Mongo-a, prave kopije i nekih karakterističnih atributa. S obzirom da kod ove baze nije potrebno eksplicitno kreirati istu, već će ona po automatizmu biti napravljena ukoliko ne postoji, import-export predstavlja kopiranje svih podataka, što može biti veoma sporo i neefikasno. Sa druge strane, Backup-ovi se prave u određenom, jasno definisanom vremenskom intervalu, gde se kreira tačka(engl. *Restore point*) koja predstavlja stanje baze u trenutku kreacije iste i ona može sadržati samo promene napravljene između dva Restore point-a, čime se kreiraju takozvani inkrementi. Na ovaj način je oporavak od otkaza mnogo brži, jer je količina podataka manja.

Na sledećoj slici se mogu videti slojevi na kojima je moguće kreirati bekapove o čemu će više reči biti u nastavku.



Slika 3. Slojevi za kreiranje bekapa kod MongoDB

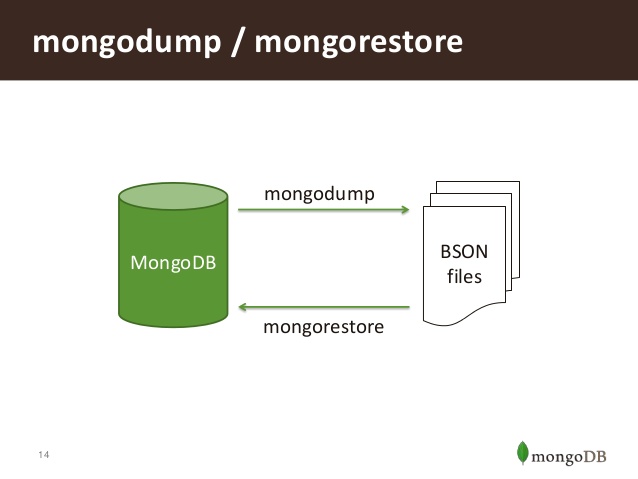
Kod MongoDB-a, kao i kod većine drugih baza, postoje dva tipa bekapa, logički i fizički.

## 3.1 Logički

Logički bekapovi predstavljaju serijalizovane podatke iz baze koji se mogu sačuvati u okviru fajla sa ekstenzijom .bson, .json ili .csv. Ukoliko je neko polje enkriptovano, ostaće tako i u samom bekapu. MongoDB obezbeđuje dva seta operacija za manipulaciju logičkim bekapovima:

* Mongodump/Mongorestore
* Mongoexport/Mongoimport

### 3.1.1 Mongodump/Mongorestore

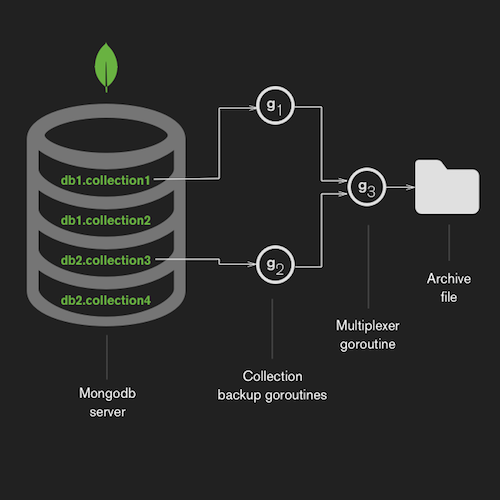


Slika 4. Mongodump-Mongorestore

Mongodump komanda kreira bekap baze, kolekcije ili upita u BSON formatu. Ovom komandom je moguće eksportovati *standalone* bazu, replike podataka ili distribuirane klastere podataka. Sama komanda je prilično slična onoj kod MySQL baze - mysqldump. Sam fajl nije moguće otvoriti u okviru nekog tekst editora i sami podaci u fajlu nisu lako čitljivi. Restore baze podataka uz pomoć ovako napravljenog bekapa je moguć korišćenjem logičkih izraza u okviru ovog fajla, takozvanih *oplog*-ova. Mongorestore komanda se koristi u ovu svrhu, kako bi se podaci deserijalizovali i dodali u ciljnu bazu. Nakon restore-a podataka, kreiraju se indeksi.

Logički bekapovi kopiraju same podake. Ne dolazi do kopiranja fizičkih fajlova koji se tiču samih informacija poput log fajlova, kontrolnih fajlova, itd. Oni služe da verifikuju strukturu baze, preuzmu i zapakuju podatke i prenesu ih na različite operativne sisteme ili okruženja. Na primer, ukoliko nam je potrebno da prenesemo bazu na drugi server, koristili bismo ovaj tip bekapa.

Ovde može doći do određenih problema, poput onog kada je baza veća nego sama memorija sistema, što može dovesti do neželjenih grešaka. Takođe, ukoliko se ova komanda izvršava direktno nad bazom, a ne nad nekom replikom, može doći do nekonzistentnosti samih podataka. Zato se dobrom praksom smatra globalno zaključavanje baze tokom kreiranja bekapova. Još jedan detalj koji može uticati na ishod kopiranja baze jeste različita verzija BSON-a, gde može doći do neočekivanih rezultata.



Slika 5. Bekapovi kolekcija

Same komande je moguće okinuti preko nekog CLI-a(engl. *Command Line Interface*) uz pomoć kojeg je moguće namestiti i automatsko kreiranje bekapova, ali je za to potrebno podesiti još par stvari. Ukoliko se baza nalazi na Linux serveru, moguće je preko CLI-a pozivom komande kreirati sam bekap. Takođe je moguće na isti način i restore-ovati samu bazu.

Mongodump ima određene opcije koje mogu biti navedene u okviru komande:

* --host [host-name] - Ime ili IP servera gde se nalazi baza
* --username [username]
* --password [password]
* --port [port] - Port na kome se hostuje baza
* --db [database-name] - Ime baze
* --out [backup-directory] - Direktorijum u okviru kojeg se kreira bekap

Pored ovih standardnih, dostupni su još neki flegovi kojima je moguće, na primer definisati namespace za koji želimo da napravimo bekap (--nsInclude).

Primer kreiranja bekapa baze podataka sa imenom myDb u folder opt:

mongodump --db myDb --out /opt/

Ukoliko se ne navede --db opcija, sve baze će biti bekapovane. Da bi se kopirala samo određena kolekcija iz određene baze, potrebno je navesti --collection [collection-name] odmah nakon mongodump komande, dok se u ostatku navodi ime baze i folder kao u prvonavedenom primeru.

Mongorestore komanda služi za importovanje baze iz fajla kreiranog mongodump komandom, gde ukoliko ne postoji baza pod navedenim imenom, po automatizmu se kreira ista.

Ukoliko baza već postoji, podaci se dodaju, s tim što se ovom komandom ne prepisuju podaci preko dokumenata koji već postoje sa istom vrednošću jedinstvenog polja, na primer \_id-ja. Uz pomoć ove komande je moguće i dodati podatke koji se nalaze u okviru zipovanog bekapa. Ona je slična prethodnoj uz par razlika kao što je opcija --drop umesto --out gde se navodi lokacija bekapa i uklanja se ukoliko postoji. Takođe postoji mogućnost da se promeni ime baze ili kolekcije koja se vraća navođenjem --nsFrom i --nsTo flegova, kao i da se neke kolekcije uključe (--nsInclude) ili isključe(--nsExclude) prilikom importa.

Da bi se proces kreiranja bekapova automatizovao, mogu se koristiti neki takozvani *task scheduler*-i ili *cron* poslovi, koji će okidati zadatu komandu u tačno određenom vremenskom roku. Danas postoje neke open-source biblioteke za kreiranje bekapova kao što su MongoDB Backup, Node.js aplikacija koja pruža CLI i API za okidanje ovih komandi ili Mongob napisan u Python-u koji pruža mogućnost kreiranje zipovanog bekapa ili direktno kopiranje baze na drugu instancu. Veoma je koristan, jer pruža mogućnost kreiranja inkrementalnih bekapova koji su relativno mali i sadrže samo promene od prethodno kreiranog, što je posebno važno u slučaju kontinualne integracije i razvoja nekog softvera koji se oslanja na Mongo bazu podataka.

### 3.1.2 Mongoexport/Mongoimport

Prilikom korišćenja ovih komandi, bekap se kreira u okviru JSON fajla koji je moguće otvoriti u bilo kom tekst editoru ili CSV fajla. Podaci su čitljivi, ali je veličina fajla veća nego u prethodno opisanom slučaju. Samo importovanje ne zavisi od verzija JSON-a, tako da se ove komande koriste u slučaju nekompatibilnih verzija.

Ove komande su granularnije i moguće je zadati neke specifične opcije prilikom exportovanja, gde je moguće označiti koje kolone želimo da exportujemo za svaki red u kolekciji. U okviru Mongoexport-a je moguće i filtriranje i sortiranje elemenata, kao i paginacija i preskakanje određenog broja redova.

Primer:

mongoexport --db=myDb --username=user --password=userPass --collection=myCollection --fields=name,price --sort=”{price:1}” --skip=10 --limit=5 --jsonArray --pretty --sort --out=”backup.json”

Mongoimport komanda služi kako bi se dodali podaci iz JSON, CSV ili TSV fajla. Glavna razlika u odnosu na mongorestore predstavlja fleg --mode uz pomoć kojeg je moguće specificirati mod rada same komande u slučaju da se importuje dokument koji predstavlja duplikat, koji ima istu vrednost za jedinstveno polje kao neki dokument u bazi, u okviru koje se dodaju podaci. Ovim je moguće prevazići problem koji se javlja kod mongorestore-a. Modovi rada su:

* Insert
* Upsert
* Merge

Insert mod logovaće grešku ukoliko se pokuša dodavanje dokumenta koji sadrži duplikat, tj. istu vrednost polja koje je definisano kao jedinstveno, sa jedinstvenim indeksom poput \_id-ja. Ovaj mod očekuje da u okviru baze nema dokumenata koji imaju iste vrednosti polja naznačenih kao jedinstveni.

Upsert mod menja postojeći dokument u bazi dokumentom iz import fajla, čime će podaci iz fajla pregaziti sve podatke u okviru tog dokumenta sa istim podacima za polja koja imaju jedinstvene indekse. Ukoliko dokument u bazi ima sledeće podatke:

{

“\_id”:{

“$oid”:”5d9788aedfe48c7bea5fgsss”

}

“price”:1125,

“name”:”Smart Watch”,

“slug”:”smart-watch”

}

a dokument u import fajlu sadrži:

{

“\_id”:{

“$oid”:”5d9788aedfe48c7bea5fgsss”

}

“price”:999,

“name”:”Smart Watch replaced”,

“slug”:”smart-watch replaced”,

“newField”:”This is new field which did not exist”

}

U samoj bazi će se naći importovani dokument (drugonavedeni u primeru), gde će doći do jednostavne zamene.

Merge mod predstavlja spajanje dva dokumenta koji imaju iste vrednosti jedinstvenih polja u jedan, rezultujući dokument. Ovim modom se omogućava ažuriranje vrednosti polja sa istim imenom iz import fajla i dodavanje svih polja čija se imena(key-evi) razlikuju. Ukoliko se u bazi nalazi dokument:

{

“\_id”:{

“$oid”:”5d9788aedfe48c7bea5fgsss”

}

“price”:1125,

“name”:”Smart Watch”,

“slug”:”smart-watch”

}

dok dokument u import fajlu sadrži:

{

“\_id”:{

“$oid”:”5d9788aedfe48c7bea5fgsss”

}

“price”:999,

“name”:”Smart Watch replaced”,

“descr”:”Descr”,

“newField”:”This is new field which did not exist”

}

rezultujući dokument će izgledati:

{

“\_id”:{

“$oid”:”5d9788aedfe48c7bea5fgsss”

}

“price”:999,

“name”:”Smart Watch replaced”,

“slug”:”smart-watch”,

“descr”:”Descr”,

“newField”:”This is new field which did not exist”

}

## 3.2 Fizički

Fizički bekapovi predstavljaju kreiranje bekapova na nivou fajl-sistema ili memorijskog resursa. Predstavljaju takozvane *snapshot*-ove podataka u konkretnom trenutku, koji zapravo sadrže samo promenjene podatke. Spajanjem svih snapshot-ova(takozvano *mount*-ovanje) kreira se celokupna kopija baze. Fizički bekapovi se obično sastoje od *redo* logova, kontrolnih fajlova i dokumenata koji sadrže podatke.

Ovde generalno postoje dva pristupa prilikom koriščenja ovakve tehnike kopiranja:

* Hladno kopiranje
* Toplo kopiranje

Pod pojmom hladno kopiranje podataka se podrazumeva proces gašenja računara na kome se nalazi baza, kreiranje kopije i propratnih fajlova neophodnih za rekonstrukciju baze i ponovno paljenje mašine na kojoj su originalni podaci. Karakteristike ovakvog pristupa su:

* Kopije su konzistentne
* Nije potrebno kreirati *redo*-logove
* Kreira se fizička kopija
* Retko se vrši u realnom vremenu jer je preskupo za velike baze
* Zahteva gašenje mašine na kojoj se nalazi baza tako da podaci nisu dostupni u ovom vremenskom intervalu

Oporavak kod ovakvog pristupa je nepotpun. Preslikavanjem podataka iz hladne kopije u originalnu bazu se gube sve promene nastale od trenutka kreiranja kopije do trenutka otkaza, ukoliko do otkaza dođe.

Sa druge strane, vruće kopiranje ne zahteva gašenje mašine na kojoj se baza nalazi, kreiraju se logovi koji predstavljaju promene koje se vrše nad podacima. S vremena na vreme se prave i kopije podataka koje predstavljaju osnovu na koju se dodaju određene promene iz logova prilikom oporavka. Ovakav pristup karakterišu činjenice da:

* Kopije nisu konzistentne
* Prave se redo-logovi
* Nema gašenja mašine

Na nivou fajl sistema, moguće je kreirati kopije fajlova koristeći alate poput *cp* i *rsync*. S obzirom da kopiranje više fajlova nije atomična operacija, sve upise u bazi tokom tog vremenskog intervala treba stopirati, jer če u suprotnom kopije biti nekonzistentne.

Ovakav način kreiranja bekapa ne pruža podršku kada je u pitanju oporavak replika u određenom vremenskom trenutku i teško je izvodljivo u slučaju distribuiranih klastera. Ovakvi bekapovi su veliki jer sadrže indekse i dupliciraju informacije o fragmentaciji. Na primer, mongodump kreira mnogo manje bekapove.

Kreiranje snapshot-ova uz pomoć LVM-a (engl. Logic Volume Manager) na Linux sistemima se može pokrenuti komandom:

lvcreate --size 100M --snapshot --name mdb-snap01 /dev/vg0/mongodb

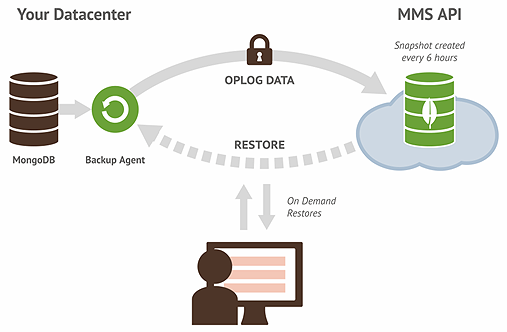
Ovom komandom se kreira LVM snapshot pod nazivom mdb-snap01 veličine 100 megabajta. Ovde je jako bitno obezbediti i rezervisati memoriju potrebnu za pamćenje svih promena u okviru snapshot-a jer ukoliko se desi da nema dovoljno rezervisanog prostora, ova “slika baze” ne može biti korišćena. Snapshot-ove je moguće kompresovati gzip komandom. Kada želimo da restorujemo bazu korišćenjem snapshot-ova, potrebno je sve njih mount-ovati u okviru željene baze. Ukoliko su ove “slike” kompresovane, treba ih prvo dekompresovati.

lvcreate --size 1G --name mdb-new vg0

gzip -d -c mdb-snap01.gz | dd of=/dev/vg0/mdb-new

mount /dev/vg0/mdb-new /srv/mongodb

Snapshot-ove je moguće kreirati i pomoću nekih drugih alata, poput WiredTiger engine-a, MongoDB Atlas-a, MongoDB Cloud Manager-a ili Amazonovog EBS-a.

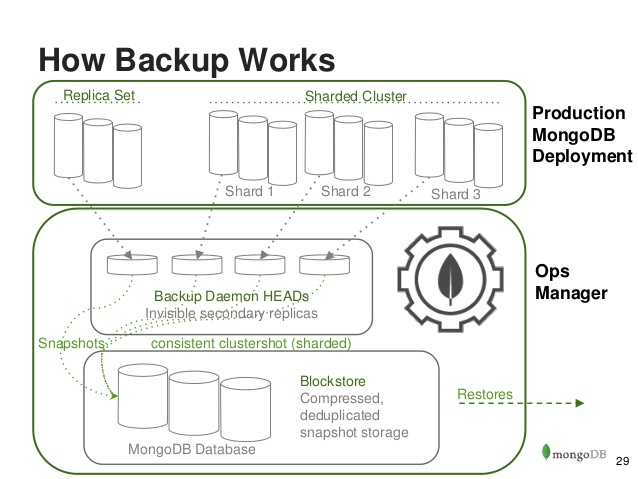


Slika 6. Bekap sa MongoDB Managemet Service

Koriščenjem MongoDB menadžment servisa se snapshot-ovi kreiraju na svakih 6 sati. Predstavlja jedinu opciju za oporavak replika i snapshot-ova klastera na distribuiranim čvorovima.

Korišćenjem MongoDB Atlasa, servisa u oblaku, omogućava se kreiranje snapshot-ova klastera hostovanih na Microsot Azure, AWS-u ili GCP-u. Postoji opcija za uključivanje kreiranja cloud bekapova. Od verzije 3.6 je moguće bekapovanje distribuiranih klastera podataka.

Ops Manager kao još jedan od alata, pruža mogućnost korišćenja MongoDB Cloud Manager-a u okviru korisničke infrastrukture.



Slika 7. Ops manager

# 4. Strategije za Backup

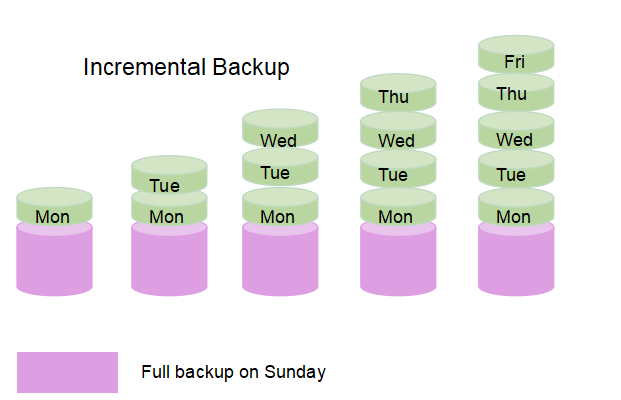
U zavisnosti od same aplikacije i zahteva koje ista treba da ispuni, postoje nekoliko strategija koje se mogu pratiti za kreiranje backup-ova, gde nijedan od njih ne predstavlja idealan scenario. Prilikom odabira strategije, treba voditi računa o količini podataka, replikaciji, vremena potrebnog za oporava, kao i praga tolerancije na gubitak podataka koja je prihvatljiva.

## 4.1 Full backup

Predstavlja najjednostavniju strategiju. Kreiraju se kompletne kopije podataka u trenutku kreacije, čime se omogućava kompletan oporavak iz te jedne “slike” podataka koja se kreira. U ovu svrhu, moguće je koristiti “mongodump” komandu.

## 4.2 Inkrementalni

U okviru inkrementalnih bekapova pamte se podaci koji su promenjeni u odnosu na poslednje napravljeni bekap. Predstavljaju sekvencijalne promene nad podacima i sadrže manju količinu podataka, pa je samim tim oporavak brži. Za kompletan oporavak je potrebno imati i full backup, kako bi se nad njim konstruisao lanac inkrementalnih bekapova u pravom rasporedu.



Slika 8. Inkrementalni bekapovi

## 4.3 Replica set bekapovi

Ukoliko postoje replike podataka, moguće je iskoristiti prednosti koje te replike pružaju u svrhu kreiranja bekapova. Kako bi se smanjio uticaj na performanse servera/baze prilikom kreiranja bekapova, gde u trenutku kreiranja bekapa baza može biti preopterećena, podaci se mogu čitati iz samih replika, pri čemu treba biti oprezan, jer može doći do nekonzistentnosti podataka, tako da ovi bekapovi ne mogu kompletno zameniti neke standardne.

## 4.4 Sharded Cluster bekapovi

U slučaju da se koristi sharding mehanizam, potrebno je napraviti bekapove svih shardova kao i bekap konfiguracije koja sadrži metapodatke o samim delovima. Moguće je koristiti “mongodump” komandu zajedno sa flegom –oplog ili korišćenjem nekog eksternog servisa za konfiguracione podatke.

## 4.5 Cloud based bekapovi

Mnogi cloud provajderi nude mogućnost kreiranja bekapova za MongoDB.

# 

# 4. Zaključak

MongoDB kao jedna od popularnijih baza podataka danas, pruža različite opcije kada je reč o kreiranju bekapova baza i kolekcija. Moguće je kreirati ih manuelno u određenim vremenskim tačkama, a moguće je i automatizovati sam proces. Danas postoje neki *third-party* alati koji olakšavaju sam proces kopiranja podataka između različitih servera i aplikacija, a postoje i oni *native* poput MongoDB Atlasa, koji pružaju mogućnost kreiranja *cloud* bekapova. U novijim verzijama prevaziđeni su određeni problemi prilikom pravljenja kopija distribuiranih klastera, tako da predstavlja odličan izbor u slučajevima gde je bitno maksimizovati skalabilnost i rasteretiti sistem, pa samim tim i smanjiti rizik od otkaza.

# 

# 5. Reference

[1] “Filesystem snapshots” <https://docs.mongodb.com/manual/tutorial/backup-with-filesystem-snapshots/>

[2] ”Cloud backup” <https://docs.atlas.mongodb.com/backup/cloud-backup/overview/#on-demand-snapshots>

[3] ”Backup and Restore” <https://www.mongodb.com/basics/backup-and-restore>

[4] “MongoDB backup” <https://www.splessons.com/lesson/mongodb-backup/>