**Univerzitet u Nišu, Elektronski fakultet**

**Katedra za računarstvo**

**SEMINARSKI RAD**

**Backup i restore kod MySQL-a**

Student:

Mladen Radenković 1083

Niš 2020. godine

Sadržaj

[Uvod 4](#_Toc40986714)

[Zašto pravimo backup? 4](#_Toc40986715)

[Definisanje zahteva za oporavkom 4](#_Toc40986716)

[Dizajniranje MySQL backup rešenja 5](#_Toc40986717)

[Online i Offline backup 6](#_Toc40986718)

[Logički i sirovi backup 7](#_Toc40986719)

[Logički backup 7](#_Toc40986720)

[Sirovi backup 8](#_Toc40986721)

[Za šta praviti backup? 9](#_Toc40986722)

[Inkrementalni i diferencijalni backup 10](#_Toc40986723)

[Upravljanje i pravljenje rezervnih kopija binarnih logova 11](#_Toc40986724)

[Format binarnog loga 11](#_Toc40986725)

[Sigurno čišćenja starih binarnih logova 12](#_Toc40986726)

[Pravljenje rezervnih kopija podataka 12](#_Toc40986727)

[Pravljenje logičkog backup-a 12](#_Toc40986728)

[SQL dumps 12](#_Toc40986729)

[Delimitirani backup fajlovi 13](#_Toc40986730)

[Snapshot-ovi fajl sistema 14](#_Toc40986731)

[Kako rade LVM snapshot-ovi 15](#_Toc40986732)

[Preduslovi i konfiguracija 16](#_Toc40986733)

[Kreiranje, montiranje i uklanjanje LVM snapshot-a 16](#_Toc40986734)

[LVM snapshotovi za online backup-ove 17](#_Toc40986735)

[InnoDB backup-ovi bez zaključvanja sa LVM snapshot-ovima 18](#_Toc40986736)

[Oporavak iz backup-a 18](#_Toc40986737)

[Vraćanje sirovih fajlova 19](#_Toc40986738)

[Startovanje MySQL-a posle vraćanja sirovih fajlova 19](#_Toc40986739)

[Vraćanje logičkih backup-ova 20](#_Toc40986740)

[Učitavanje SQL fajlova 20](#_Toc40986741)

[Učitavanje delimitiranih fajlova 21](#_Toc40986742)

[Oporavak u određenom trenutku 22](#_Toc40986743)

[Alati za backup i oporavak 24](#_Toc40986744)

[MySQL Enterprice Backup 24](#_Toc40986745)

[Percona XtraBackup 24](#_Toc40986746)

[mylvmbackup 24](#_Toc40986747)

[Zmanda Recovery Manager 25](#_Toc40986748)

[mydumper 25](#_Toc40986749)

[mysqldump 25](#_Toc40986750)

[Zaključak 26](#_Toc40986751)

[Literatura 27](#_Toc40986752)

# Uvod

Pre nego što počnemo, razjasnićemo neke ključne termine koje ćemo koristiti u ovom radu. Često možemo čuti za takozvane hot, warm i cold backup-ove. Ljudi generalno koriste ove termine za označavanje uticaja backup-ova: „hot“ backup-ovi ne treba da zahtevaju bilo kakav prekid sistema. Međutim, neki alati čak koriste reč „hot“ i kada definitivno ne podrazumevaju ono što mi smatramo hot backup-ovima. Mi ćemo u ovom radu izbegavati da koristimo ove termine i umesto toga rećićemo koliko tehnika ili alat prekidaju naš server.

Druge dve zbunjujuće reči su vraćanje (*eng. restore*) i oporavak (*eng. recover*). Vraćanje označava pribavljanje podataka iz backup-a i bilo njihovo učitavanje u MySQL ili postavljanje fajlova tamo gde MySQL očekuje da budu. Oporavak obično označava ceo proces spasavanja sistema ili dela sistema, nakon što se desila neka katastrofa. Ovo uključuje vraćanje podataka iz backup-ova, kao i sve neophodne korake da bi server ponovo upotpunosti funkcionisao, kao što je restartovanje MySQL-a, promena konfiguracije i tako dalje.

Za mnoge ljude, oporavak samo označava popravku oštećenih tabela posle pada sistema. Ovo nije isto kao i oporavak celog servera. Oporavak od pada uređaja za skladištenje usklađuje njegove podatke i log fajlove. Ovo osigurava da fajlovi podataka sadrže samo promene urađene od izvršenih transakcija, i ovo reprodukuje transakcije iz log fajlova koje još nisu primenjene na fajlove podataka. Ovo bi moglo biti deo celokupnog procesa oporavka. Međutim, ovo nije isto kao i oporavak koji treba da uradimo nakon slučajne naredbe DROP TABLE, na primer.

# Zašto pravimo backup?

Evo nekoliko razloga zašto su backup-ovi važni:

* Oporavak od katastrofe – Oporavak od katastrofe je ono što radimo kada hardver padne, neka greška ošteti podatke ili server i njegovi podaci postanu nedostupni ili neupotrebljivi iz nekog drugog razloga. Moramo biti spremni na sve, od nekog ko se slučajno poveže na pogrešan server i uradi ALTER TABLE, do zlonamernih napadača ili MySQL greške, iako su šanse za bilo koje posebno katastrofalno stanje prilično male.
* Ljudi menjaju svoja mišljenja – Bićemo iznenađeni koliko često ljudi namerno obrišu podatke i onda ih žele nazad.
* Revizija – Ponekad moramo da znamo kako su naši podaci ili šema izgledali u nekom trenutku u prošlosti. Na primer, možda ćemo otkriti grešku u našoj aplikaciji i biće potrebno da pogledamo šta je kod nekada koristio.
* Testiranje – Jedan od najlakših načina za testiranje sa realnim podacima je periodično osvežavanje test servera sa najnovijim podacima iz produkcije. Ako pravimo backup-ove, to je lako, samo vratimo backup na test server.

# Definisanje zahteva za oporavkom

Ako sve ide dobro, nikada nećemo morati da razmišljamo o oporavku. Ali kada se nešto desi, ni najbolji backup sistem na svetu nam neće pomoći. Umesto toga, trebaće nam dobar sistem oporavka.

Nažalost, lakše je učiniti backup sisteme da rade neometano nego napraviti dobre procese i alate oporavka. Evo i zašto:

* Backup-ovi dolaze prvi. Ne možemo da izvršimo oporavak ako prvo ne napravimo rezervnu kopiju, odnosno backup. Iz tog razloga naša pažnja treba da bude usresređena na pravljenje rezervnih kopija kada pravimo sistem.
* Backup-ovi se automatizuju pomoću skripti. Lako je potrošiti vreme finim podešavanjem backup procesa, često ne razmišljajući o njima.
* Backup-ovi su rutine, ali oporavak je obično krizna sitaucija.
* Jedna osoba može da planira, dizajnira i implementira backup-ove. Ta osoba može da ne bude dostupna kada se desi katastrofa. Potrebno je obučiti nekoliko osoba i planirati pokrivanje, tako da ne tražimo nekvalifikovanu osobu da povrati naše podatke.

Postoje dva veoma važna zahteva koji su korisni za razmatranje kada planiramo backup i strategiju oporavka. To su cilj tačke oporavka (*eng. recovery poin objective - RPO*) i cilj vremena oporavka (*eng. recovery time objective - RTO*). Oni definišu koliko podataka možemo da izgubimo i koliko dugo ćemo čekati da ih povratimo. Treba da odgovorimo na sledeće vrste pitanja kada definišemo naše RPO i RTO:

* Koliko podataka možemo da izgubimo bez ozbiljnih posledica? Da li je potreban oporavak u određenom trenutku (*eng. point-in-time recovery*) ili je prihvatljivo izgubiti svaki posao koji se degodio posle našeg poslednjeg regularnog backupa?
* Koliko brz oporavak mora da bude? Koje vrste zastoja su prihvatljive? Koje udare (na primer, delimična nedostupnost) mogu da prihvate naša aplikacija i korisnici i kako ćemo omogućiti nastavak funkcionisanja kada se ovi scenariji dogode?
* Šta je potrebno da se oporavi? Uobičajeni zahtevi su da se oporavi ceo server, jedna baza podataka, jedna tabela ili samo određene transakcije ili naredbe.

Česta greška koju možemo videti je: „Replika je backup“. Replika nije backup. Da bismo videli zašto, razmotrimo sledeće: Da li će ona da nam pomogne da vratimo sve podatke ako slučajno izvršimo DROP DATABASE nad našom bazom u produkciji? Replike ne prolaze čak ni ovaj jednostavan test. Ne samo da nisu rezervne kopije, već nisu ni zamena za rezervne kopije. Ništa drugo sem backup-a ne ispunjava potrebu za rezervnim kopijama.

# Dizajniranje MySQL backup rešenja

Pravljenje rezervnih kopija je teže nego što izgleda. U osnovi backup je samo kopija podataka, ali potrebe naše aplikacije, arhitektura MySQL-ovog mehanizama za skladištenje i konfiguracija sistema mogu da otežaju pravljenje kopija podataka.

Pre nego što uđemo u detalje o svim dostupnim opcijama, evo nekih preporuka za kreiranje backup-a:

* Sirove rezervne kopije su praktično neophodne za velike baze podataka: logički backup-ovi su jednostavno previše spori i zahtevaju dosta resursa, a oporavak iz logičkog backup-a traje predugo. Backup-ovi zasnovani na snapshot-u, Percona XtraBackup i MySQL Enterprise Backup, su najbolje opcije. Za male baze podataka, logički backup-ovi mogu da rade dobro.
* Čuvati nekoliko generacija backup-a.
* Periodično izdvajati logičke backup-ove.
* Čuvati binarne logove *(eng. binary logs)* za oporavak u određenom trenutku *(eng. point-in-time recovery).* Možemo postaviti expire\_logs\_day dovoljno dugo da se oporavi iz bar dve generacije sirovih backup-ova, tako da možemo kreirati repliku i pokrenuti je iz pokrenutog mastera bez primene bilo kog binarnog loga na nju. Treba napraviti kopiju binarnih logova nezavisno od postavljenog isteka i čuvati ih u backup dovoljno dugo da bi se oporavili iz bar najnovijeg logičkog backup-a.
* Nadgledati vaše backup-ova i procese izrade backup-ova nezavisno od samih alata za izradu backup-a. Potrebna je spoljašnja potvrda da su uredu.
* Treba testirati backup-ove i proces oporavka tako što ćemo proći kroz ceo postupak oporavka. Treba da izmerimo resurse neophodne za oporavak (CPU, prostor na disku, vreme, mrežni protok i tako dalje)
* Dobro razmisliti o sigurnosti. Šta se dešava ako neko kompromituje naš server – da li on može onda da pristupi i backup-u servera ili obrnuto?

RPO i RTO će voditi strategiju pravljenja backup-a. Da li nam je potrebna mogućnost oporavka u određenom trenutku ili je dovoljno da se oporavimo do sinoćne rezervne kopije i izgubimo sve što je od tada urađeno? Ako nam je potreban oporavak u određenom trenutku, možemo da napravimo regularnu rezervnu kopiju i da omogućimo binarni log, tako da možemo da vratimo tu rezervnu kopiju i vratimo se u željenu tačku reprodukcijom binarnog loga.

Uopšteno, što više dozvolimo da možemo da izgubimo, to je lakše da napravimo backup. Ako imamo veoma stroge zahteve, teže je da osiguramo da sve možemo da povratimo. Takođe postoje različite vrste oporavaka u određenom trenutku. „Soft“ zahtev za oporavak u određenoj tački u vremenu označava da bi želeli da ponovo kreiramo svoje podatke tako da budu dovoljno blizu gde su bili kada se problem desio. „Hard“ zahtev označava da nikada ne možemo da tolerišemo gubitak izvršene transakcije, čak iako se nešto strašno dogodi (kao što je kada se zapali server). Ovo zahteva specijalne tehnike, kao što su čuvanje našeg binarnog loga na posebnoj SAN (Strorage area network) jedinici ili korišćenje DRBD (Distributed Replicated Block Device) replike diska.

## Online i Offline backup

Isključivanje MySQL-a da bi se uradio backup je najlakši, najsigurniji i uopšte najbolji način da dobijemo konzistentnu kopiju podataka uz minimalan rizik od oštećenja ili nekonzistentnosti. Ako isključimo MySQL, možemo da kopiramo podatke bez ikakvih komplikacija od stvari kao što su prljavi baferi u InnoDB baferu ili drugim keševima. Ne moramo da brinemo da će naši podaci biti modifikovani dok pokušavamo da napravimo rezervnu kopiju, i pošto server nije pod opterećenjem od aplikacije, možemo uraditi backup veoma brzo.

Međutim, stavljanje servera offline je skuplje nego što se čini. Čak i ako možemo da minimizujemo vreme zastoja, isključivanje i ponovno pokretanje MySQL-a može da traje dugo pod zahtevnim opterećenjima i velikom količinom podataka. Kao rezultat toga, gotovo sigurno će trebati da napravimo naše backup-ove tako da oni ne zahtevaju da server u produkciji bude offline, odnosno da se isključi iz mreže. Međutim, u zavisnosti od naših zahteva za konzistencijom, pravljenje rezervne kopije dok je server online, može još da znači i prekidanje usluge.

Jedan od najvećih problema kod mnogih backup metoda je njihova upotreba naredbe: FLUSH TABLES WITH READ LOCK. Ovo kaže MySQL-u da zatvori i zaključa sve tabele, upiše MyISAM-ove fajlove sa podacima na disk (ali ne InnoDB-ove) i očistiti keš upita. Ovo može da traje vremenski dugo. Tačno koliko dugo je nepredvidivo, biće čak i još duže ako globalni ključ za čitanje mora da čeka da se završi naredba koja se dugo izvršava, ili ako imamo mnogo tabela. Dok se ključ ne otpusti, ne možemo da menjamo nikakve podatke na serveru i sve će se blokirati i staviti u red. FLUSH TABLES WITH READ LOCK nije toliko skupo koliko isključenje servera, zato što većina naših keševa je još uvek u memoriji i server je još uvek „warm“.

Najbolji način za izbegavanje bilo kakve upotrebe FLUSH TABLES WITH READ LOCK je koriščenje samo InnoDB tabela. Ne možemo izbeći upotrebu MyISAM tabela za privilegije i druge informacije o sistemu, ali ako se ti podaci često menjaju (što je normalno), možemo očistiti i zaključati samo te tabele bez prouzrokovanja problema.

Faktori vezani za performanse koje treba razmotriti kada planiramo pravljenje rezervnih kopija:

* Vreme zaključavanja – Koliko vremena nam je potrebno da držimo ključeve, dok pravimo rezervne kopije?
* Vreme backup-a – Koliko vremena je potrebno za kopiranje rezervne kopije na odredište?
* Učitavanje backup-a – Koliko to utiče na performanse servera da kopira backup na odredište?
* Vreme oporavka – Koliko traje kopiranje backup imidža sa njegove lokacije za skladištenja na MySQL server, ponovno reprodukovanje binarnih logova i tako dalje?

Najveći kompromis je vreme backup-a u poređenju sa učitavanjem backup-a. Često možemo da poboljšamo jedan o trošku drugog, na primer, možemo da damo prednost backup-u izazivajući veću degradaciju performansi na serveru.

## Logički i sirovi backup

Postoje dva glavna načina za pravljenje rezervnih kopija MySQL-ovih podataka: sa logičkim backup-om (koji se takođe naziva „dump“) i kopiranjem sirovih fajlova. Logički backup sadrži podatke u obliku koji MySQL može da interpretira ili kao SQL ili kao delimitiran tekst. Sirovi fajlovi su fajlovi onakvi kakvi postoje na disku. Svaki tip backup-a ima prednosti i nedostatke.

### Logički backup

Logički backup-ovi imaju sledeće prednosti:

* Oni su obični fajlovi kojima možemo da manipulišemo i pregledavamo ih pomoću editora i command line alata, kao što su grep i sed. Ovo može da bude veoma korisno prilikom vraćanja podataka ili kada samo želimo da pregledamo podatke bez vraćanja.
* Jednostavni su za vraćanje. Možemo samo da ih prenesemo u mysql ili koristimo mysqlimport.
* Možemo da napravimo rezervne kopije i vratimo ih preko mreže – to jest, na drugoj mašini od MySQL hosta.
* Mogu da budu veoma fleksibilni, zato što mysqldump, alat koji većina ljudi koristi za pravljenje rezervnih kopija, može da prihvati mnogo opcija, poput WHERE klauzule za restrikciju redova koje smeštamo u rezervne kopije.
* Oni su nezavisni od mehanizma skladištenja. Pošto ih kreiramo izdvajanjem podataka iz MySQL servera, oni apstrahuju razlike u osnovnom skladištu podataka. Tako, možemo da napravimo rezervne kopije iz InnoDB tabela i vratimo ih u MyISAM tabele sa veoma malo posla. To ne možemo da radimo sa sirovim kopijama fajla.

Logički backup-ovi imaju i svoje nedostatke:

* Server mora da ih generiše, tako da oni koriste više CPU ciklusa.
* Logički backup-ovi mogu da budu veći od osnovnih fajlova u nekim slučajevima. U praksi, logički backup-ovi su obično manji od sirovih backup-ova, ali nisu uvek. ASCII reprezentacija podataka nije uvek efikasna kao način na koji mehanizam sladištenja čuva podatke. Na primer, integer zahteva 4 bajta za sladištenje, ali kada je napisan u ASCII, može zahtevati više do 12 karaktera. Često možemo da kompresujemo fajlove i dobijemo manji backup, ali ovo koristi više CPU resursa. Logički backup-ovi su obično manji od sirovih backup-ova ako postoji mnogo indeksa.
* Odlaganje i vraćanje podataka nije uvek zagarantovano da će rezultirati istim podacima. Problemi sa predstavljanjem pokretnog zareza, greške i tako dalje mogu prouzrokovati probleme, mada je ovo retko.
* Vraćanje iz logičkog backup-a zahteva da MySQL učita i interpretira naredbe, konvertuje ih u format sladištenja i ponovo izgradi indekse, a sve ovo je veoma sporo.

Najveći nedostaci su troškovi izbacivanja podataka iz MySQL-a i troškovi učitavanja podataka nazad putem SQL naredbi. Ako koristimo logičke backup-ove, važno je da testiramo potrebno vreme za povratak podataka.

Alat mysqldump može nam pomoći kada radimo sa InnoDB tabelama, zato što on formira izlaz tako da će iskoristiti prednosti InnoDB-ovog koda za brzo kreiranje indeksa nakon što ga ponovo učita. Ovo može dosta da smanji vreme vraćanja. Što više ima indeksa, veće su koristi.

### Sirovi backup

Sirovi backup-ovi imaju sledeće prednosti:

* Sirovi backup-ovi fajlova jednostavno zahtevaju da se kopiraju željeni fajlovi negde drugde za backup. Sirovi fajlovi ne zahtevaju nikakav dodatni posao za generisanje.
* Vraćanje sirovih backup-ova može da bude jednostavnije, u zavisnosti od mehanizma skladištenja. Za MyISAM može da bude jednostavno kao kopiranje fajlova na odredišta. Međutim, InnoDB zahteva da se zaustavi server i eventualno preduzmu drugi koraci.
* Sirovi backup-ovi InnoDB i MyISAM podataka su veoma prenosivi kroz platforme, operativne sisteme i MySQL verzije.
* Može biti brže vraćanje sirovih backup-ova, zato što MySQL server ne mora da izvršava bilo koji SQL ili stavlja indekse. Zapravo, jedna od najgorih stvari oko logičkih backup-ova je njihovo nepredvidivo vreme vraćanja.

Neki od nedostataka sirovih backup-ova su:

* InnoDB-ovi sirovi fajlovi su često mnogo veći od odgovarajućih logičkih backup-ova. InnoDB-ov prostor za tabele obično ima dosta neiskorišćenog prostora. Mali deo prostora se takođe koristi u druge svrhe, osim za skladištenje podataka iz tabele.
* Sirovi backup-ovi nisu uvek prenosivi kroz platforme, operativne sisteme i MySQL verzije. Osetljivost imena fajla na velika i mala slova i formati pokretnog zareza su mesta na kojima možemo da naiđemo na probleme. Možda nećemo moći da premestimo fajlove na sistem koji ima drugačiji format pokretnog zareza (međutim, velika većina procesora koristi IEEE format pokretnog zareza).

Sirovi backup-ovi su generalno jednostavniji i efikasniji. Ne treba se oslanjati na sirove backup-ove za dugoročno čuvanje podataka, u tom slučaju treba periodično praviti logičke backup-ove.

Ne treba smatrati da je backup (posebno sirovi backup) dobar, dok se ne testira. Za InnoDB, ovo znači pokretanje MySQL instance i puštanje InnoDB oporavka, a onda pokrenje CHECK TABLES. Ovo možemo preskočiti ili samo pokrenuti innochecksum na fajlove. Za MyISAM, treba pokrenuti CHECK TABLES ili koristiti myisamchk. Možemo pokrenuti CHECK TABLES na sve tabele pomoću mysqlcheck komande.

Preporučuje se spoj dva pristupa: pravljenje sirovih kopija, onda pokretanje instance MySQL servera sa dobijenim podacima i pokretanje mysqlcheck. Zatim, bar periodično, pravljenje kopije podataka pomoću mysqldump da bi dobili logički backup. Ovim dobijamo prednosti oba pristupa, bez nepotrebnog opterećenja servera u produkciji tokom odlaganja (izbacivanja) podataka.

## Za šta praviti backup?

Zahtevi za oporavak diktiraju za šta treba praviti backup. Najjednostavnija strategija je samo praviti rezervne kopije za naše podatke i definicije tabela, ali ovo je minimalni pristup. Obično je potrebno mnogo više da bi se server oporavio za korišćenje u produkciji. Evo nekih stvari koje možemo da razmatramo da uključimo u naše MySQL backup-ove:

* Neočigledni podaci – Ne treba zaboraviti podatke koje je lako zanemariti, na primer, binarni logovi i InnoDB transakcioni logovi.
* Kod – Savremeni MySQL server može da čuva mnogo koda, kao što su trigeri i stored procedure.
* Konfiguracija replikacije – Ako vršimo oporavak servera koji je uključen u replikaciju, potrebno je uključiti u backup sve replikacione fajlove koji će biti potrebni za to – na primer, binarni logovi, log indeks fajlovi i .info fajlovi. Minimalno treba uključiti izlaz iz SHOW MASTER STATUS i/ili SHOW SLAVE STATUS.
* Konfiguracija servera – Ako moramo da izvršimo oporavak od realne katastofe – recimo, ako pravimo server iz početka u novom data centru nakon zemljotresa – potrebno je imati konfiguracione fajlove servera uključene u backup.
* Odabrani fajlovi operativnog sistema – Kao i kod konfiguracije servera, važno je napraviti rezervne kopije bilo koje spoljašnje konfiguracije koja je neophodna za server u produkciji.

Ove preporuke se jednostavno prevode u „praviti rezervne kopije svega“ u mnogim scenarijima. Međutim, ako imamo mnogo podataka, ovo može da bude skupo i možda ćemo morati da budemo pametniji u pogledu pravljenja rezervnih kopija. Možda ćemo želeti da pravimo rezervne kopije različitih podataka u različite backup-ove. Na primer, možemo praviti rezervne kopije podataka, binarnih logova i konfiguracionih fajlova operativnog sistema odvojeno.

## Inkrementalni i diferencijalni backup

Uobičajena strategija za rad sa previše podataka je pravljenje redovnih inkrementalnih ili diferencijalnih backup-ova. Razika između ova dva termina je sledeća: diferencijalni backup je backup svega što se promenilo od poslednjeg potpunog backup-a, dok inkrementalni backup sadrži sve što se promenilo od poslednjeg backup-a bilo koje vrste.

Na primer, pretpostavimo da pravimo potpuni backup svake nedelje. U ponedeljak, radimo diferencijalni backup svega što se promenilo od nedelje. U utorak, imamo dva izbora: možemo praviti rezervne kopije svega što se promenilo od nedelje (diferencijalni backup) ili možemo praviti rezervne kopije samo podataka koji su se promenili od backup-a iz ponedeljka (inkrementalni backup).

I diferencijalni i inkrementalni backup su delimični backup-ovi: oni obično ne sadrže kompletan skup podataka, zato što se neki podaci gotovo sigurno nisu promenili. Delimični backup-ovi su često poželjni zbog smanjenja režijskih troškova na serveru, vremena i prostora backup-a. Ipak, neki delimični backup-ovi zapravo ne smanjuju režijske troškove na serveru. Na primer, Percona XtraBackup i MySQL Enterprise Backup i dalje skeniraju svaki blok podataka na serveru, tako da oni ne smanjuju mnogo režijske troškove, iako štede vreme i prostor na disku.

Preporuke vezane za pravljenje backup-ova:

* Treba praviti rezervne kopije binarnih logova. Takođe može se koristiti FLUSH LOGS za započinjanje novog binarnog loga posle svakog backup-a, a onda praviti rezervne kopije samo novih binarnih logova.
* Ne treba praviti rezervne kopije za tabele koje se nisu promenile. Neki mehanizmi skladištenja, kao što je MyISAM, beleže kada se desila poslednja promena svake tabele. Ovo vreme možemo videti pregledom fajlova na disku ili izvršenjem SHOW TABLE STATUS. Ako koristimo InnoDB, triger nam može pomoći da pratimo poslednje promene beleženjem promena u malu tablelu „last changed time“. Ovo treba da radimo samo za tabele koje se retko menjaju, tako da troškovi budu minimalni. Korisnička skripta može jednostavno da odredi koje su tabele promenjene.

Ako imamo „lookup“ tabele koje sadrže podatke kao što su liste naziva meseca na različitim jezicima ili skraćenice država ili regiona, dobra ideja je da ih stavimo u posebnim bazama podataka, tako da ne moramo stalno da pravim rezervne kopije za njih.

* Ne treba praviti rezervne kopije za redove koji se nisu promenili. Ako se samo upisuje u tabelu, možemo dodati TIMESTAMP kolonu i praviti rezervne kopije samo za redove koji su dodati posle poslednjeg backup-a.
* Ne treba praviti rezervne kopije nekih podataka uopšte. Ponekad ovo ima dosta smisla – na primer, ako imamo skladište podataka koje je izgrađeno iz drugih podataka, možemo samo da napravimo rezervnu kopiju podataka koji se koriste za izgradnju skladišta, umesto samog skladišta podataka. Ovo može biti dobra ideja čak i ako je veoma spor oporavak zbog ponovne izgradnje skladišta iz originalnih fajlova. Izbegavanje backup-ova može tokom vremena da sakupi mnogo veće uštede od potencijalno bržeg vremena oporavka koje bismo dobili potpunim backup-om. Takođe možemo da ne pravimo rezervne kopije privremenih podataka, kao što su tabele koje čuvaju podatke o sesiji web sajta.

Nedostaci inkrementalnih backup-ova uključuju povećanu složenost oporavka, povećan rizik i duže vreme oporavka. Ako možemo da uradimo potpuni backup, treba da ga uradimo radi jednostavnosti. Definitivno je potrebno povremeno raditi potpuni backup – bar jednom nedeljno. Ne možemo očekivati oporavak od mesečnih inkrementalnih backup-ova. Čak i nedeljni predstavlja mnogo posla i rizika.

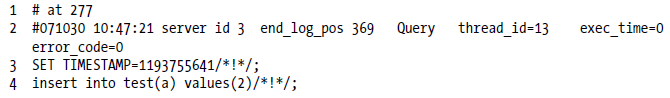
# Upravljanje i pravljenje rezervnih kopija binarnih logova

Binarni logovi servera su jedna od najvažnijih stvari za koje možemo da pravimo rezervne kopije. Oni su neophodni za oporavak u određenom vremenskom trenutku i pošto su oni obično manji od naših podataka, jednostavniji su za češće pravljenje rezervnih kopija. Ako imamo backup naših podataka u nekom trenutku i sve binarne logove od tada, možemo da reprodukujemo binarne logove i prebacimo se na promene napravljene nakon poslednjeg potpunog backup-a. MySQL takođe koristi binarni log za replikaciju.

Ako izgubimo naše podatke, ne želimo da izgubimo i binarne logove. Da bi minimizovali šanse da se ovo desi, možemo ih držati odvojeno. Radi dodatne sigurnosti, možemo ih čuvati na SAN-u ili ih replicirati na drugi uređaj pomoću DRBD-a. Dobra je ideja da često pravimo rezervne kopije za binarne logove. Na primer, ako ne možemo da izgubimo podatke koji vrede više od 30 minuta, treba da pravimo rezervne kopije na svakih 30 minuta.

## Format binarnog loga

Binarni log se sastoji iz niza događaja. Svaki događaj ima zaglavlje fiksne veličine koje sadrži razne informacije, kao što su tekući timestamp i zadata baza podataka. Možemo da koristimo mysqlbinlog alat za pregledavanje sadržaja binarnih logova i da štampamo neke od informacija iz zaglavlja. Na slici 1 je dat primer izlaza.



Slika 1

Linija 1 sadrži pomeraj u bajtovima u log fajlu (u ovom slučaju 277). Linija 2 sadrži sledeće stavke:

* Datum i vreme događaja.
* ID izvornog servera, koji je neophodan za sprečavanje beskonačne petlje u replikaciji i drugih problema.
* end\_log\_pos – koji predstavlja pomeraj u bajtovima od sledećeg događaja. Ova vrednost je netačna za većinu događaja u transakcijama sa više naredbi. MySQL kopira događaje u bafer na masteru tokom takvih transakcija, ali on ne zna poziciju sledećeg loga događaja kada to radi.
* Tip događaja. Tip u ovom primeru je Query, ali postoje i drugi različiti tipovi.
* threadID – nit koja je izvršila događaj na izvornom serveru, što je važno za reviziju.
* exec\_time – koji je razilka između timestamp-a naredbe i vremena kada je upisana u binarni log.
* Bilo koji kod greške koja se desila na izvornom serveru.

Sve dalje linije sadrže podatke neophodne da se reprodukuje promena. Korisnički definisane promenljive i bilo koja druga specijalna podešavanja, kao što je timestamp koji je na snazi kada se naredba izvrši, takođe se pojavljuju ovde.

## Sigurno čišćenja starih binarnih logova

Potrebno je odlučiti o politici isteka loga kako bi obezbedili da MySQL ne popuni disk binarnim logovima. Koliko će naši logovi da rastu zavisi od radnog opterećenja i formata loga. Treba čuvati logove onoliko dugo koliko su korisni, ako je moguće. Njihovo čuvanje je korisno za postavljanje replika, analiziranje radnog opterećenja servera, reviziju i oporavak u određenom trenutku od poslednjeg potpunog backup-a. Treba da razmotrimo sve ove potrebe kada odlučujemo koliko dugo želimo da čuvamo logove.

Uobičajeno se koristi expire\_logs\_days promenljiva kojom se kaže MySQL-u da očisti logove nakon nekog vremena. Postavka expire\_logs\_days stupa na snagu pri pokretanju servera ili kada MySQL rotira binarne logove, tako da ako se binarni log nikada ne popuni i rotira, server neće čistiti starije ulaze. On odlučuje koji fajlovi će se čistiti, pregledavajući njihova vremena modifikacije, a ne njihov sadržaj.

# Pravljenje rezervnih kopija podataka

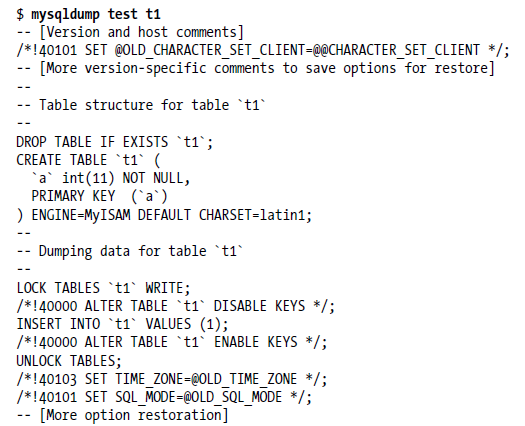
Postoje bolji i gori načini da se naprave rezervne kopije, a očigledni načini ponekad nisu tako dobri. Cilj je da se maksimizuje mreža, disk i kapacitet CPU-a kako bi se napravile rezervne kopije što je brže moguće. Ovde se radi o balansiranju i moramo da eksperimentišemo da bi postigli željeni rezultat.

## Pravljenje logičkog backup-a

Postoje dve vrste logičkih backup-ova: SQL dumps i delimitirani fajlovi.

### SQL dumps

SQL dumps je ono sa čim je većina ljudi upoznata, jer njih mysqldump kreira po defaultu. Na primer, odlaganje male tabele sa defaultnim opcijama će proizvesti rezultat kao na slici 2.



Slika 2

Dump fajl sadrži i strukturu tabele i podatke, sve napisano kao validne SQL naredbe. Fajl započinje komentarima koji postavljaju različite MySQL opcije. One su prisutne ili da bi posao povratka bio efikasniji ili zbog kompatibilnost i ispravnosti. Zatim možemo videti strukturu tabele, a onda i njene podatke. Na kraju, skripta resetuje opcije koje je promenila na početku dump-a.

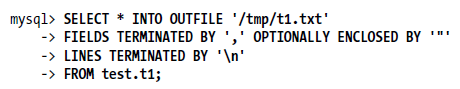
Izlaz dumpa se izvršava prilikom operacije povratka. Ovo je praktično, ali defaultne opcije mysqldump-a nisu sjajne za pravljenje ogromnog backup-a i one će detaljnije biti objašnjene kasnije u radu.

mysqldump nije jedini alat za pravljenje SQL logičkih backup-ova. Takođe ih možemo kreirati i sa mydumper ili phpMyAdmin. Ovo su glavni nedostaci pravljenja SQL dump logičkih backup-ova, nezavisno od alata koji se koristi:

* Šema i podaci se čuvaju zajedno – Iako je ovo zgodno ako želimo povratak iz jednog fajla, ali ovo otežava situaciju ako treba da vratimo samo jednu tabelu ili želimo da vratimo samo podatke. Možemo da ublažimo ovaj problem odlaganjem dva puta – jednom podatke, a drugi put šemu.
* Ogromne SQL naredbe – Veliki je posao za server da parsira i izvršava sve SQL naredbe. Ovo je veoma spor način za učitavanje podataka.
* Jedan ogroman fajl – Većina tekst editora ne mogu da menjaju velike fajlove ili fajlove sa veoma dugim linijama. Iako ponekad možemo da koristimo command-line editore, kao što su sed ili grep, da bismo izvukli podatke koji su nam potrebni, poželjno je da fajlovi budu mali.
* Logički backup-ovi su skupi – Postoje efikasniji načini da se podaci izvuku iz MySQL-a od pribavljanja iz mehanizma za skladištenje i slanja preko klijent/server protokola kao skup rezultata.

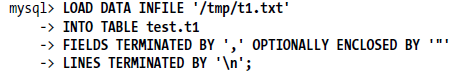
### Delimitirani backup fajlovi

Za kreiranje logičkog backup-a naših podataka u delimitiranom formatu fajla možemo koristiti SELECT INTO OUTFILE SQL naredbu. Delimitirani fajlovi sadrže sirove podatke predstavljene u ASCII, bez SQL-a, komentara i naziva kolona. Na slici 3 je dat primer koji odlaže podatke u formatu u kome su vrednosti odvojene zarezima (CSV).



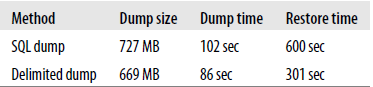
Slika 3

Rezultujući fajl je kompaktniji i lakši za manipulisanje pomoću command-line alata od SQL dump fajla, ali najveća prednost ovog pristupa je brzina pravljenja rezervne kopije i vraćanje. Možemo učitati podatke nazad u tabelu pomoću LOAD DATA INFILE, istim opcijama koje se koriste za njihovo odlaganje, kao što je prikazano na slici 4.



Slika 4

Kako bi demonstrirali razlike u brzini pravljenja backup-a i vraćanja između SQL fajlova i delimitiranih fajlova, kao primer je uzeta tabela sa 15 miliona redova koja koristi oko 700MB na disku. Tabela na slici 5 upoređuje performanse dva načina pravljenja rezervnih kopija i vraćanja.



Slika 5

SELECT INTO OUTFILE metod ima nekoliko ograničenja:

* Možemo da pravimo rezervne kopije samo u fajlove na mašini na kojoj je pokrenut MySQL server.
* MySQL mora da ima dozvolu za upisivanje u folder u koji je fajl upisan, zato što MySQL server, a ne korisnik koji pokreće SQL naredbu, upisuje fajl.
* Iz sigurnosnih razloga ne možemo da prepisujemo postojeći fajl bez obzira na dozvole fajla.
* Ne možemo direktno da odložimo u kompresovan fajl.
* Neke stvari, kao što su nestandardni skupovi karaktera, teško su ispravni u koraku eksporta ili importa.

## Snapshot-ovi fajl sistema

Snapshot-ovi fajl sistema su odličan način za pravljenje online rezervnih kopija. Fajl sistemi koji omogućavaju snapshot mogu da kreiraju konzistentni imidž njihovog sadržaja u trenutku, koji zatim možemo da koristimo za pravljenje rezervne kopije. Fajl sistemi i uređaji koji omogućuju snapshot uključuju fajl sistem FreeBSD-a, fajl sistem ZFS-a, GNU/Linux-ov Logical Volume Manager (LVM) i mnoge SAN sisteme i rešenja za skladištenje fajlova, kao što je NetApp uređaj skladištenja.

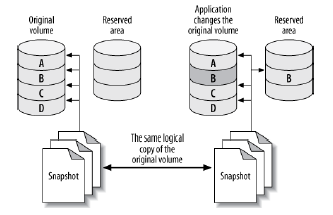
Ne treba mešati snapshot sa backup-om. Pravljenje snapshot-a je jednostavan način za smanjenje vremena za koje mora da se drže ključevi; nakon otpuštanja ključeva, moramo kopirati fajlove u rezervnu kopiju. Na InnoDB-u možemo opciono da napravimo snapshotove čak i bez pribavljanja ključeva.

Snapshot može da bude odličan način za pravljenje rezervnih kopija za specifične namene. Jedan primer je rezervna kopija u slučaju problema tokom upgrade-a. Možemo napraviti snapshot, uraditi upgrade i ako postoji problem, samo da se vratimo na snapshot. Možemo da uradimo istu stvar za bilo koju operaciju koja je neizvesna i rizična, kao što je ažuriranje ogromne tabele.

### Kako rade LVM snapshot-ovi

LVM koristi tehnologiju copy-on-write da napravi snapshot t.j. logičku kopiju celog volumena (*eng. volume)* u vremenskom trenutku. Ovo nije fizička kopija. Logička kopija sadrži sve iste podatke kao volumen za koji smo uradili snapshot, ali u početku ne sadrži podatke. Umesto kopiranja podataka na snapshot, LVM beleži vreme kada je snapshot kreiran, a onda čita podatke iz originalnog volumena kada zahtevamo iz snapshota.

Kada se menjaju podaci u originalnom volumenu, LVM kopira promenjene blokove u oblast rezervisanu za snapshot pre nego što upiše bilo koje promene. LVM ne čuva više starih verzija podataka, tako da dodatni upisi u blokove koji su promenjeni u originalnom volumenu ne zahtevaju bilo koji dodatni rad za snapshot. Drugim rečima, samo prvi upis u svaki blok prouzrokuje copy-on-write u rezervisanoj oblasti. Sada, kada zahtevamo te blokove u snapshot, LVM čita podatke iz kopiranih blokova, umesto iz originalnog volumena. Ovo omogućava da idalje vidimo iste podatke u snapshot-u bez blokiranja bilo čega na originalnom volumenu. Ovo je prikazano na slici 6.



Slika 6

Snapshot kreira novi logički uređaj u /dev direktorijumu i možemo da montiramo ovaj uređaj kao što bismo montirali bilo koji drugi.

Teorijski možemo da napravimo snapshot za ogroman volumen i zauzmemo veoma mali fizički prostor ovom tehnikom. Međutim, potrebno je da izdvojimo dovoljno prostora da bismo mogli da čuvamo sve blokove za koje očekujemo da se ažuriraju u originalnom volumenu dok držimo snapshot otvoren. Ako ne rezervišemo dovoljno copy-on-write prostora, snapshot-u će nestati prostor i uređaj će postati nedostupan.

### Preduslovi i konfiguracija

Potrebno je obezbediti da je sistem konfigurisan na takav način da možemo da dobijemo konzistentnu kopiju svih fajlova za koje želimo da napravimo backup u jednom trenutku. Prvo, treba proveriti da li naš sistem ispunjava sledeće uslove:

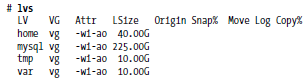
* Svi InnoDB fajlovi (InnoDB fajlovi za tabele i InnoDB transakcioni logovi) moraju biti na jednom logičkom volumenu. Potrebna nam je apsolutna trenutna konzistencija i LVM ne može da kreira konzistentne snapshot-ove više od jednog volumena u trenutku.
* Ako je potrebno da pravimo rezervne kopije i za definicije tabela, direktorijum MySQL podataka mora da bude u istom logičkom volumenu.
* Moramo da imamo dovoljno slobodnog prostora u grupi volumena da bismo kreirali snapshot. Kada podesimo naš sistem, ostaće malo nezauzetog prostora kako bismo imali prostora za snapshotove kasnije.

LVM ima koncept grupe volumena, koji sadrži jedan ili više logičkih volumena. Možemo da vidimo volumene grupe na našem sistemu kao što je prikazano na slici 7. Ovaj izlaz prikazuje grupu volumena koja ima četiri logička volumena distribuirana kroz jedan fizički volumen, sa oko 250GB slobodno. Naredba vgdispaly nam daje više detalja ako nam je potrebno.



Slika 7

Na slici 8 su prikazani logički volumeni na sistemu. Izlaz prikazuje da mysql volumen ima 225GB prostora. Naziv uređaja je /dev/vg/mysql.



Slika 8

### Kreiranje, montiranje i uklanjanje LVM snapshot-a

Snapshot možemo da kreiramo jednom komandom. Samo je potrebno da odlučimo gde da ga smesimo i koliko prostora da zauzmemo za copy-on-write. Možemo da koristimo više prostora nego što mislimo da će nam trebati. LVM ne koristi prostor koji smo odmah specificirali, on ga samo rezerviše za buduću upotrebu, tako da nema štete u rezervisanju previše prostora. Na slici 9 je prikazano kreiranje snapshot-a sa 16GB prostora za copy-on-write i sa nazivom backup\_mysql.



Slika 9

Na slici 10 možemo da vidimo status novokreiranog volumena. Atributi snapshota se razlikuju od atributa originalnih uređaja, a prikaz prikazuje i dodatne informacije: njegovo poreklo i koliko od zauzetih 16GB se trenutno koristi za copy-on-write.



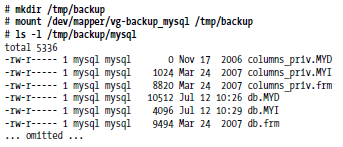
Slika 10

Dobra je ideja da nadgledamo to dok pravimo backup-ove, kako bismo videli ako se uređaj napuni i pokvari. Možemo da nadgledamo status uređaja interaktivno ili pomoću sistema za praćenje, kao što je Nagios, što je prikazano na slici 11.



Slika 11

mysql volumen sadrži fajl sistem. Ovo znači da snapshot volumen možemo, takođe, montirati i koristiti kao i bilo koji drugi fajl sistem, što je prikazano na slici 12.



Slika 12

Možemo da odmontiramo i uklonimo snapshot pomoću lvremove komande, kao na slici 13.



Slika 13

### LVM snapshotovi za online backup-ove

Sada, kada smo videli kako da kreiramo, montiramo i obrišemo snapshot-ove, možemo ih koristiti za pravljenje backup-ova. Prvo ćemo videti kako da napravimo rezervnu kopiju InnoDB baze podataka bez stopiranja MySQL servra, ali sa globalnim ključem za čitanje. Povezaćemo se na MySQL server i isprazniti tabele na disku sa globalnim ključem za čitanje, a onda ćemo da nabavimo koordinate binarnog loga, kao što je prikazano na slici 14.



Slika 14

Snimićemo izlaz iz naredbe SHOW MASTER STATUS i voditi računa da zadržimo otvorenu MySQL konekciju tako da se ključevi ne otpuštaju. Onda možemo da napravimo LVM snapshot i odmah otpustimo ključ za čitanje, bilo pomoću UNLOCK TABLES ili zatvaranjem konekcije. Na kraju, ćemo montirati snapshot i kopirati fajlove na rezervnu lokaciju.

Glavni problem s ovim pristupom je taj što može dugo da traje pribavljanje ključa za čitanje, posebno ako postoje upiti koji se dugo izvršavaju. Svi upiti će biti blokirani dok konekcija čeka za globalnim ključem za čitanje i nemoguće je predvideti koliko će to trajati.

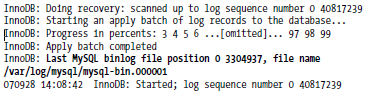
### InnoDB backup-ovi bez zaključvanja sa LVM snapshot-ovima

Backup-ovi bez zaključvanja su samo malo drugačiji. Razlika je u tome što ne radimo FLUSH TABLES WITH READ LOCK. Ovo znači da nećemo imati nikakvu garanciju da će naši MyISAM fajlovi biti konzistentni na disku, ali ako koristimo samo InnoDB, ovo neće biti problem. I dalje ćemo imati neke MyISAM tabele u mysq sistemskoj bazi podataka, ali one se verovatno neće promeniti u trenutku kada radimo snapshot. Ako mislimo da se mysql sistemske tabele mogu promeniti, možemo da ih zaključamo, a onda očistimo, kao što je prikazano na slici 15.



Slika 15

Ne pribavljamo globalni ključ za čitanje, tako da nam ne treba ni SHOW MASTER STATUS. Međutim, kada pokrenemo MySQL na snapshotu (da bi potvrdili integritet backupa), videćemo nešto kao na slici 16 u log fajlu. InnoDB beleži poziciju MySQL binarnog loga koja odgovara tački do koje se oporavio. Ovo je pozicija binarnog loga koja može da se koristi za oporavak u određenom trenutku.



Slika 16

# Oporavak iz backup-a

Način oporavka podataka zavisi od toga kako smo napravili backup. Potrebno je da preduzmemo neke ili sve od sledećih koraka:

* Zaustaviti MySQL server
* Voditi beleške o konfiguraciji servera i dozvolama za fajlove
* Premestiti podatke iz backup-a u MySQL direktorijum podataka
* Promeniti konfiguraciju
* Promeniti dozvole za fajlove
* Restartovati server sa ograničenim pristupom i sačekati da se upotpunosti pokrene
* Učitati logičke backup fajlove
* Ispitati i reprodukovati binarne logove
* Proveriti šta je povraćeno
* Restartovati server sa potpunim pristupom

Za vreme oporavka, često je bitno da se MySQL učini nedostupnim za sve, osim za proces oporavka. Treba startovati MySQL sa opcijama –skip-networking i –socket=/tmp/mysql\_recover.sock kako bismo se osigurali da ne bude dostupan postojećim aplikacijama dok ga ne proverimo i vratimo online. Ovo je posebno važno za logičke backup-ove, koji se učitavaju u delovima.

## Vraćanje sirovih fajlova

Vraćanje sirovih fajlova je obično jednostavno, zato što nema mnogo opcija. Ovo može da bude dobro ili loše, u zavisnosti od naših potreba za oporavkom. Uobičajeni postupak je jednostavno kopiranje fajlova na svoje mesto.

Da li je potrebno isključiti MySQL zavisi od mehanizma za skladištenje. MyISAM-ovi fajlovi su uglavnom međusobno nezavisni i jednostavno kopiranje svakog .frm, .MYI i .MYD fajla tabele dobro funkcioniše, čak i ako je server pokrenut. Server će pronaći tabelu čim je neko zahteva u upitu ili na neki drugi način natera server da je traži (na primer, izvršenjem SHOW TABLES). Ako je tabela otvorena kada kopiramo u te fajlove, verovatno će uzrokovati grešku, tako da pre nego što ovo učinimo, treba ili obrisati ili preimenovati tabelu, ili koristiti LOCK TABLES i FLUSH TABLES da bismo je zatvorili.

Ako vraćamo tradicionalnu InnoDB postavku, gde su sve tabele uskladištene u jednom prostoru tabele, moraćemo da isključimo MySQL, kopiramo ili premestimo fajlove na mesto, a onda ponovo pokrenemo MySQL. Takođe je potrebno obezbediti da se InnoDB-ovi fajlovi logova transakcija podudaraju sa fajlovima prostora tabela. Ako se fajlovi ne podudaraju – na primer, ako premestimo fajlove tabele, ali ne i fajlove transakcionih logova – InnoDB će odbiti da se pokrene. Ovo je razlog zašto je bitno praviti rezervne kopije fajlova transakcionih logova zajedno sa podacima.

Ako koristimo funkciju InnoDB fajl po tabeli (innodb\_file\_per\_table), InnoDB čuva podatke i indekse za svaku tabelu u .ibd fajlu, koji je kao kombinacija MyISAM-ovih .MYI i .MYD fajlova. Možemo da napravimo backup i da izvršimo vraćanje pojedinačnih tabela, kopiranjem ovih fjalova, i to možemo uraditi dok je server pokrenut, ali ovo nije tako jednostavno kao sa MyISAM-om. Pojedinačni fajlovi nisu nezavisni od InnoDB-a u celini. Svaki .ibd fajl ima interne infomacije koje kažu InnoDB-u kako je fajl povezan sa glavnim (deljenim) prostorom tabele.

### Startovanje MySQL-a posle vraćanja sirovih fajlova

Postoje nekoliko stvari koje treba uraditi pre pokretanja MySQL servera koji se oporavlja.

Prva i najvažnija stvar, a koja se najlakše zaboravlja, je da se proveri konfiguracija servera i osigura da fajlovi koji se vraćaju imaju ispravnog vlasnika i dozvole, pre nego što pokušamo da pokrenemo MySQL server. Ovi atributi moraju da budu sasvim tačni ili se MySQL neće pokrenuti. Atributi se razlikuju od sistema do sistema, tako da treba da proverimo beleške kako bi videli tačno šta je potrebno da se podesi. Obično želimo da mysql korisnik i grupa poseduju fajlove i direktorijume, koje želimo da budu dostupni za čitanje i upis od strane tog korisnika i grupe ali ne i drugih.

Takođe, treba nadgledati MySQL log grešaka dok se server startuje. Zatim možemo pokrenuti MySQL server i gledati greške. Ako sve dobro prođe, imaćemo dobro oporavljen server nakon što se MySQL startuje.

Gledanje loga grešaka je čak važnije u novijim MySQL verzijama. Starije verzije se neće pokrenuti ako InnoDB ima grešku, ali u novijim verzijama server će se pokrenuti u svakom slučaju, a samo će se onemogućiti InnoDB. Čak i ako izgleda da se server startovao bez greške, treba da izvršimo SHOW TABLE STATUS u svakoj bazi podataka, a onda da proverimo ponovo log greške.

## Vraćanje logičkih backup-ova

Ako vraćamo logičke backup-ove, umesto sirovih fajlova, treba da koristimo MySQL server za učitavanje podataka u tabele, za razliku od korišćenja operativnog sistema za jednostavno kopiranje fajlova na svoje mesto.

Pre nego što učitamo taj odbačen fajl, treba da razmotrimo koliko je on veliki, koliko treba vremena da se učita i sve šta želimo da uradimo pre pokretanja, kao što je obaveštavanje korisnika ili onemogućavanje dela naše aplikacije. Onemogućavanje binarnih logova može da bude dobra ideja, osim ako ne treba da ponovimo vraćanje na replike: ogroman odbačen fajl je dosta težak za server da se učita, a upisivanje u binarni log dodaje još više režijskih troškova. Učitavanje ogromnih fajlova takođe ima posledice za neke mehanizme skladištenja. Na primer, nije dobra ideja učitati 100 GB podataka u InnoDB jednom transakcijom, zbog ogromnog povratnog segmenta. Treba učitati u segmente i izvršiti transakciju posle svakog segmenta.

Postoje dve vrste vraćanja koje odgovaraju dvema vrstama logičkih backup-ova koje možemo da napravimo.

### Učitavanje SQL fajlova

Ako imamo SQL dump, fajl će sadržati izvršni SQL. Sve što treba da uradimo je da ga pokrenemo. Pretpostavimo da smo napravili backup sakila baze podataka i šeme u jednom fajlu, komandu na slici 17 možemo iskoristiti za vraćanje podataka.



Slika 17

Takođe možemo učitati fajl iz mysql command-line klijenta pomoću SOURCE komande. Iako je ovo drugi način za isto, ovo olakšava neke stvari. Na primer, ako smo administrativni korisnici u MySQL-u, možemo isključiti binarno logovanje naredbi koje ćemo izvršiti iz naše klijentske konekcije, a onda učitati fajl baz potrebe za restartovanjem MySQL servera, kao što je prikazano na slici 18.



Slika 18

Ako smo kompresovali backup, ne treba da odvojeno dekompresujemo i učitamo ga. Umesto toga, treba dekompresovati i učitati u jednoj operaciji, jer je to mnogo brže, kao što je prikazano na slici 19.



Slika 19

Šta ako želimo da povratimo samo jednu tabelu­ (na primer, actor tabelu)? Ako naši podaci nemaju break liniju, nije teško povratiti podatke i ako je šema već na svom mestu (uspostavljena), to možemo uraditi naredbom kao na slici 20. A ako je fajl kompresovan, onda naredbom kao na slici 21.



Slika 20



Slika 21

Ako je potrebno kreirati tabelu i povratiti podatake, a imamo celu bazu podataka u jednom fajlu, moraćemo da editujemo fajl. Zbog ovoga neki ljudi odlažu svaku tabelu u sopstvenom fajlu. Većina editora ne može da radi sa ogromnim fajlovima, posebno ako su kompresovani. Osim toga, ne želimo da menjamo fajl – samo želimo da izdvojimo relevantne linije – tako da ćemo verovatno morati da uradimo neki posao u komandnoj liniji. Lako je koristiti grep da bi izdvojili samo INSERT naredbe za određenu tabelu, kao što smo uradili u prethodnim komandama, ali tako je teže dobiti CREATE TABLE naredbu. Na slici 22 je prikazana sed skripta koja izvlači odlomak koji nam je potreban.



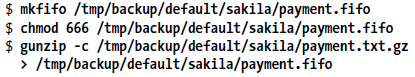
Slika 22

Ako moramo da radimo ovu vrstu posla za povratak podataka, naši bakup-ovi su loše dizajnirani. Samo treba da pravimo backup za svaku tabelu u sopstvenom fajlu ili još bolje da posebno pravimo backup za podatke, a posebno za šemu.

### Učitavanje delimitiranih fajlova

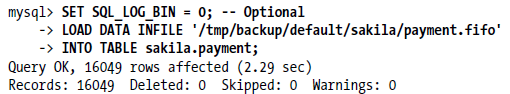
Ako smo odložili podatke pomoću SELECT INTO OUTFILE, moraćemo da korisitmo LOAD DATA INFILE sa istim parametrima da bismo ih vratili. Takođe možemo koristiti mysqlimport, koji je omotač oko LOAD DATA INFILE. On se oslanja na konvencije o imenovanju kako bi odredio gde treba da se učitaju podaci fajla.

Postoji odlična optimizacija koju možemo koristiti sa LOAD DATA INFILE. Ona mora da čita direktno iz fajla, tako da možemo misliti da moramo da dekompresujemo fajl pre nego što ga učitamo, što je veoma sporo i intenzivno za disk. Međutim, postoji način da se to otkloni, bar na sistemima koji podržavaju FIFO „named pipe“ fajlove, kao što je GNU/Linux. Prvo, treba kreirati imenovani pipe i prebaciti dekompresovane podatke u njemu, kao na slici 23. Koristimo karakter veće od (>) za redirekciju dekompresovanog izlaza u fajl payment.fifo – ne pipe simbol, koji kriera anonimne pipe-ove između programa. Fajl payment.fifo je imenovani pipe, tako da nema potrebe za anonimnim.



Slika 23

Pipe će sačekati dok ga neki program ne otvari za čitanje sa drugog kraja. MySQL server može da čita dekompresovane podatke iz pipe-a, kao i bilo koji drugi fajl. Možemo da onemogućimo binarno logovanje ako je potrebno, kao na slici 24.



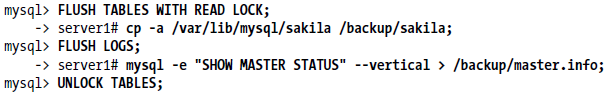
Slika 24

Jednom kada MySQL učita podatke, gunzip će izaći i možemo da obrišemo imenovani pipe. Možemo koristiti ovu istu tehniku za učitavanje kompresovanih fajlova iz command-line klijenta MySQL-a pomoću SOURCE komande.

# Oporavak u određenom trenutku

Najčešći način da uradimo oporavak u određenom trenutku pomoću MySQL-a je da vratimo poslednji potpuni bakcup i onda reprodukujemo binarne logove od tog trenutka unapred. Sve dok imamo binarni log, možemo da se oporavimo u bilo kojoj tački želimo. Glavni nedostatak je što reprodukcija binarnog loga može biti spor proces.

Uobičajeni scenario je poništavanje efekata štetnih naredbi, kao što je DROP TABLE. Razmotrićemo jednostavan primer kako da uradimo ovo korišćenjem samo MyISAM tabela. Pretpostavimo da se u ponoć izvršio backup posao ekvivalentan naredbama na slici 25, koje su kopirale bazu podataka na drugom mestu na istom serveru.



Slika 25

Zatim, kasnije tokom dana, pretpostavimo da je neko izvršio sledeću naredbu, kao na slici 26.



Slika 26

Pretpostavimo da možemo da oporavimo ovu bazu podataka izolovano (t.j. da ni jedna tabela u ovoj bazi podataka nije bila uključena u upite u više drugih baza podataka). Pretpostavimo i da smo naredbu sa greškom primetili tek neko vreme kasnije. Cilj je da oporavimo sve što se desilo sa bazom podataka, osim te naredbe. Odnosno, moramo da sačuvamo sve modifikacije koje su se desile u drugim tabelama, uključujući i nakon što se ova naredba izvršila.

Prvo, stopiramo MySQL da bi sprečili dalje modifikacije i povratimo samo bazu podataka iz backup-a, što je prikazano na slici 27.



Slika 27

Onemogućimo konekcije tokom rada dodavanjem koda na slici 28 u serverovom fajlu my.cnf. Nakon toga je bezbedno pokrenuti server, što je dato na slici 29.

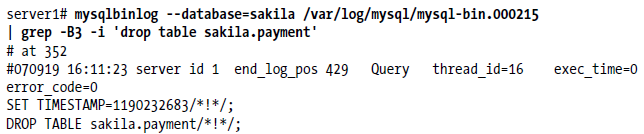


Slika 28



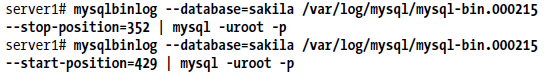
Slika 29

Sledeći zadatak je da pronađemo koje naredbe u binarnom logu želimo da reprodukujemo, a koje želimo da preskočimo. Server je kreirao samo jedan binarni log od backup-a u ponoć. Možemo da pregledamo taj fajl pomoću grep i pronađemo naredbu sa greškom, kao na slici 30.



Slika 30

Naredba koju želimo da preskočimo je na poziciji 352 u log fajlu, a sledeća naredba je na poziciji 429. Možemo da ponovo reprodukujemo log fajl do pozicije 352, a onda od 429 da nastavimo dalje, pomoću komandi prikazanih na slici 31.



Slika 31

Sada sve što treba da uradimo je da samo proverimo podatke kako bi bili sigurni, zaustavimo server i poništimo promene u my.cnf fajlu, i ponovo pokrenemo server.

# Alati za backup i oporavak

Dostupni su raznovrsni alati za pravljenje backup-a. Omiljeni su mylvmbackup za LVM snapshot backup-ovde i Percona XtraBackup (otvorenog koda) ili MySQL Enterprise Backup za „hot“ InnoDB backup-ove. Ne preporučuje se mysqldump za pravljenje backup-a bilo koje značajne količine podataka, zbog uticaja na server i nepredvidivo dugog vremena za povratak.

## MySQL Enterprice Backup

Ovaj alat, ranije poznat kao InnoDB Hot Backup ili ibbackup, je deo pretplate za MySQL Enterprice. Njegovo korišćenje ne zahteva zaustavljanje MySQL-a, postavljanje ključeva ili prekidanje normlne aktivnosti baze podataka (mada će to izazvati dodatno opterećenje na serveru). On podržava funkcije kao što su kompresovani backup-ovi i inkrementalni backup-ovi. Ovo je zvanični backup alat za MySQL.

## Percona XtraBackup

Percona XtraBackup je sličan MySQL Enterprice Backup-u na više načina, ali je on otvorenog koda i besplatan je. Pored osnovnog alata za izradu backupa, takođe postoji wrapper skripta pisana u Perlu koja poboljšava njegove funkcionalnosti za naprednije zadatke. On podržava funkcije kao što su protočne, inkrementalne, kompresovane i višenitne (paralelne) backup operacije. Takođe, on ima razne specijalne karakteristike za smanjenje uticaja backup-ova na opterećenim sistemima.

Percona XtraBackup funkcioniše tako što prati u stopu InnoDB log fajlove u pozadinskoj niti, a onda kopira InnoDB fajlove podataka. Ovaj proces uključuje posebne provere kako bi se obezbedilo da se podaci kopiraju konzistentno. Kada se svi fajlovi podataka kopiraju, nit za kopiranje loga se takođe završava. Rezultat je kopija svih podataka, ali u različitim vremenskim trenucima. Logovi sada mogu da se primene na fajlove podataka, korišćenjem InnoDB rutina oporavka od katastrofa, kako bi se svi fajlovi podataka doveli u konzistentno stanje. To se naziva procesom pripreme. Kada se jednom pripremi, backup je potpuno konzistentan i sadrži sve izvršene transakcije od krajnje tačke procesa kopiranja fajla. Sve ovo se dešava potpuno eksterno na MySQL-u, tako da nije potrebno da se ni na koji način konektujemo ili pristupamo MySQL-u.

## mylvmbackup

mylvmbackup je Perl skripta koja pomaže da se automatizuju MySQL backup-ovi preko LVM snapshot-a. On dobija globalni ključ za čitanje, kreira snapshot i oslobađa ključ. Onda kompresuje podatke pomoću tar i briše snapshot. On ima nekoliko naprednijih opcija, ali uopšteno je jednostavan alat za izvođenje LVM backup-ova.

## Zmanda Recovery Manager

Zmanda Recovery Manager za MySQL ili ZMR dolazi i u besplatnoj i u komercijalnoj verziji. U poslovnom izdanju dolazi sa menadžement konzolom koja pruža grafički web interfejs za konfigurisanje, izradu backupa, proveru, oporavak, izveštavanje i planiranje. Izdanje otvorenog koda nije ni na koji način uskraćeno, ali ne uključuje neke dodatke, kao što je web konzola. ZMR je zapravo menadžer za izradu rezervnih kopija i oporavka, a ne samo jedan alat. On umotava svoju funkcionalnost oko standardnih alata i tehnika, kao što su mysqldump, LVM snapshot-ovi i Percona XtraBackup. On automatizuje mnogo napornog rada pravljenja rezervnih kopija i oporavka.

## mydumper

Nekoliko MySQL inženjera su napravili mydumper kao zamenu za mysqldump. On je skup alata za višenitno (paralelno) pravljenje backupa i oporavka za MySQL sa mnogo dobrih karakteristika. Mnogi ljudi smatraju brzinu višenitne izrade backup-ova i vraćanja kao najatraktivniju karakteristiku ovog alata. Malo ljudi ga koristi u produkciji.

## mysqldump

Većina ljudi koristi programe koji se isporučuju sa MySQL-om, tako da je uprkos nedostacima, mysqldump najčešći izbor za krieranje logičkih backup-ova podataka i šema. Ovo je alat opšte namene koji može da se koristi za mnoge zadatke, kao što je kopiranje tabele sa jednog servera na drugi, što je prikazano na slici 32.



Slika 32

Ranije u ovom radu smo već pokazali kako se kreiraju logički backup-ovi pomoću mysqldump-a. Po defaultu, on šalje skriptu koja sadrži sve komande potrebne za kreiranje tabele i njeno popunjavanje podacima. Evo još nekih primera tipične upotrebe:

* Pravljenje logičkog backup-a svega na serveru u jedan fajl, za sve tabele u svakoj bazi podataka napraviti backup u istom vremenskom trenutku:



Slika 33

* Pravljenje logičkog backup-a samo jedne baze podataka:



Slika 34

* Pravljenje logičkog backup-a samo jedne tabele:



Slika 35

Podrazumevane opcije za mysqldump nisu dobre za većinu namena backup-a. Možemo da specifiramo neke opcije za promenu izlaza. Ovo su opcije koje se često koriste da bi mysqldump bio efikasniji i da bi njegov izlaz bio lakši za upotrebu:

* --opt – Omogućuje grupu opcija koje onemogućavaju baferovanje i upis više podataka u manje SQL naredbi u kopiji kako bi efikasnije učitali podatke. Ako onemogućimo ovu grupu opcija, mysqldump će čuvati svaku tabelu koju odlažemo u memoriju pre nego što je upiše na disk, što je nepraktično za velike tabele.
* --allow-keywords, --quote-names – Omogućavaju odlaganje i vraćanje tabela koje koriste rezervisane reči kao nazive.
* --complete-insert – Omogućava premeštanje podataka između tabela koje nemaju identične kolone.
* --tz-utc – Omogućava premeštanje podataka između servera u različitim vremenskim zonama.
* --lock-all-tables – Koristi FLUSH TABLES WITH READ LOCK za dobijanje globalno konzistentnog backup-a.
* --tab – Odlaže fajlove pomoću SELECT INTO OUTFILE
* --skip-extended-insert – Prouzrokuje da svaki red podataka ima svoju sopstvenu INSERT naredbu. Ovo nam može pomoći da selektivno vraćamo određene redove ako je potrebno.

Ako koristimo mysqldump opcije –databases ili –all-databases, rezultujući odloženi podaci će biti konzistentni u svakoj bazi podataka, jer mysqldump će istovremeno zaključati i odložiti sve tabele u bazi podataka. Međutim, tabele iz različitih baza podataka možda neće biti međusobno konzistentne. Korišćenje opcije –lock-all-tables rešava ovaj problem.

Za pravljenje backup-a InnoDB-a, treba dodati opciju –single-transaction, koja koristi InnoDB-ove MVCC karakteristike za kreiranje konzistentnog backupa u jednom trenutku, umesto da koristi LOCK TABLES. Ako dodamo opciju –master-data, backup će takođe sadržati koordinate binarnog loga servera u trenutku pravljenja backup-a, što je veoma korisno za oporavak u određenom trenutku i podešavanje replika. Međutim, treba da imamo na umu da će on za zamrzavanje servera koristiti FLUSH TABLES WITH READ LOCK da bi mogao da dobije koordinate.

# Zaključak

Backup-ovi su nam potrebni i to oni iz kojih možemo da izvršimo oporavak. Postoje mnogo načina za kreiranje backup-ova koji su u suprotnosti sa našim zahtevima za oporavak. Da bismo izbegli ovaj problem, potrebno je da definišemo i dokumentujemo naše RPO i RTO zahteve i iskoristimo ove zahteve za odabir backup sistema. Takođe je važno testirati oporavak i osigurati da on funkcioniše. Najgori trenutak da otkrijemo koliko će oporavak trajati je kada nam je stvarno potreban. Oporavak iz rezervne kopije koja se pravi nekoliko sati, može trajati nedeljama, u zavisnosti od hardvera, šeme, indeksa i podataka.

Cilj ovog rada je predstavljanje teorijskih osnova backup-a kod MySQL-a, objašnjenje pravljenja različitih tipova backup-ova i povratka na osnovu njih, kao i sagledavanje najčešćih alata za pravljenje backup-a i oporavak.

# Literatura

High performance MySQL - Baron Schwartz, Peter Zaitsev, and Vadim Tkachenko

<https://dev.mysql.com/doc/refman/8.0/en/backup-and-recovery.html>