**Univerzitet u Nišu, Elektronski fakultet**

**Katedra za računarstvo**

**SEMINARSKI RAD**

**Replikacija kod MySQL-a**

Student:

Mladen Radenković 1083

Niš 2020. godine

Sadržaj

[Uvod 4](#_Toc43761863)

[Osnove replikacije 4](#_Toc43761864)

[Problemi koje rešava replikacija 5](#_Toc43761865)

[Kako replikacija radi? 5](#_Toc43761866)

[Podešavanje replikacije 7](#_Toc43761867)

[Kreiranje naloga za replikaciju 7](#_Toc43761868)

[Konfigurisanje mastera i replike 7](#_Toc43761869)

[Pokretanje replike 9](#_Toc43761870)

[Inicijalizacija replike sa drugog servera 11](#_Toc43761871)

[Preporučena konfigacija replikacije 12](#_Toc43761872)

[Replikacija ispod haube 14](#_Toc43761873)

[Logička replikacija 14](#_Toc43761874)

[Sirova replikacija 14](#_Toc43761875)

[Poređenje logičke i sirove replikacije 15](#_Toc43761876)

[Fajlovi replikacije 17](#_Toc43761877)

[Slanje replikacionih događaja drugim replikama 17](#_Toc43761878)

[Filteri replikacije 18](#_Toc43761879)

[Replikacione topologije 19](#_Toc43761880)

[Master i više replika 20](#_Toc43761881)

[Master-master u aktivan-aktivan modu 21](#_Toc43761882)

[Master-master u aktivan-pasivan modu 22](#_Toc43761883)

[Master-master sa replikama 22](#_Toc43761884)

[Prsten replikacija 23](#_Toc43761885)

[Master, distribucioni master i replikacije 24](#_Toc43761886)

[Stablo ili piramida 25](#_Toc43761887)

[Administracija i održavanje replikacije 26](#_Toc43761888)

[Monitoring replikacije 26](#_Toc43761889)

[Merenje zaostajanja replikacije 27](#_Toc43761890)

[Ponovna sinhronizacija replike sa masterom 27](#_Toc43761891)

[Promena mastera 28](#_Toc43761892)

[Planirane promocije 28](#_Toc43761893)

[Neplanirane promocije 29](#_Toc43761894)

[Pronalaženje željenih pozicija loga 29](#_Toc43761895)

[Promena uloga u master-master konfiguraciji 31](#_Toc43761896)

[Zaključak 32](#_Toc43761897)

[Literatura 32](#_Toc43761898)

# Uvod

MySQL-ova ugrađena replikacija je osnova za izgradnju velikih aplikacija visokih performansi, korišćenjem takozvanih „scale-out“ arhitektura. Replikacija nam omogućava da konfigurišemo jedan ili više servera kao replike drugog servera, čuvajući njihove podatke sinhronizovano sa glavnom (master) kopijom. Ovo ne samo da je korisno za aplikacije visokih performansi, već je takođe i osnova mnogih strategija za visoku dostupnost, skalabilnost, oporavak od katastrofa, pravljenje backup-ova, analize, skadištenje podataka i mnoge druge zadatke. Zapravo, skalabilnost i visoka dostupnost su povezane teme, i mi ćemo se njima baviti kroz ovaj rad.

U ovom radu proučićemo neke aspekte replikacije. Počećemo sa pregledom kako replikacija funkcioniše, onda ćemo videti osnovno podešavanje servera, dizajniranje naprednijih konfiguracija replikacije i administriranje i monitoring repliciranih servera. Fokusiraćemo se uglavnom na performanse, ali podjednako ćemo se baviti i tačnošću i pouzdanošću kada je u pitanju replikacija.

# Osnove replikacije

Osnovni problem koji replikacija rešava je čuvanje podataka jednog servera sinhronizovanih sa podacima drugog. Mnoge replike se mogu povezati sa jednim masterom i ostati sinhronizovane sa njim, a replika, za uzvrat, može da se ponaša kao master.

MySQL podržava dve vrste replikacija: logičke replikacije (*eng. statement-based replication*) i sirove replikacije (*eng. row-based replication*). Replikacije zasnovane na naredbi ili logičke replikacije su dostupne od MySQL 3.23. Sirove replikacije su dodate u MySQL 5.1. Obe vrste rade tako što snimaju promene u binarnom logu mastera i reprodukuju log na repliku, i obe su asinhrone – to jest, nije zagarantovano da kopija podataka na replici bude ažurna u datom trenutku. Ne postoji garancija koliko velike latencije mogu biti na replici. Veliki upiti mogu učiniti da replika zaostaje sekundama, minutima ili čak satima iza mastera.

MySQL-ova replikacija je uglavnom kompatibilna unazad. Odnosno, noviji server obično može da bude replika starijeg servera bez problema. Međutim, starije verzije servera često ne mogu da služe kao replike novije verzije: one možda ne razumeju nove funkcije ili SQL sintaksu novijeg servera, a mogu postajati i razlike u formatima fajla koje replikacija koristi.

Replikacija uglavnom ne dodaje mnogo troškova na masteru. Ona zahteva da binarno logovanje bude omogućeno na masteru, što može imati značajne troškove, ali to nam je svakako potrebno za ispravno pravljenje backup-ova i oporavak u određenom trenutku. Pored binarnog logovanja, svaka dodata replika dodaje i malo opterećenja na masteru tokom normalnog rada. Ako replike čitaju stare binarne logove sa mastera, a ne samo prate najnovije događaje, troškovi mogu biti mnogo veći zbog I/O zahteva za čitanjem starih logova.

Replikacija je relativno dobra za skaliranje čitanja, koja možemo da usmerimo ka replici, ali nije dobar način za skaliranje upisa ako je ne dizajniramo ispravno. Pridruživanje više replika masteru prouzrokuje da se više puta obavlja upis, po jednom na svaku repliku. Ceo sistem je ograničen brojem upisa koji najslabiji deo može da izvede.

Replikacija je takođe uzaludan posao sa više od nekoliko replika, jer usuštini duplicira mnogo podataka bespotrebno. Na primer, jedan master sa 10 replika ima 11 kopija istih podataka i duplicira većinu istih podataka u 11 različitih keševa. Ovo nije ekonomična upotreba hardvera.

## Problemi koje rešava replikacija

Ovo su neke uobičajene upotrebe replikacije:

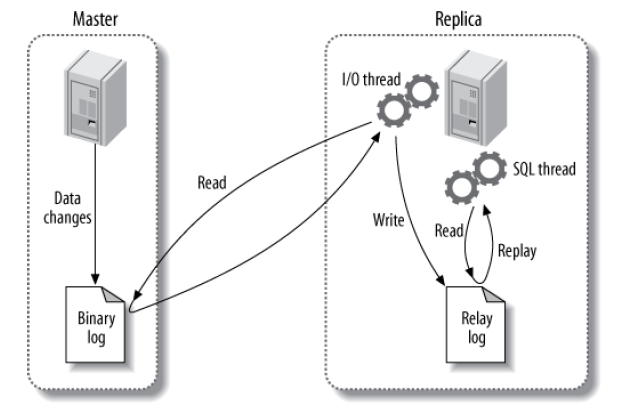
* Distribucija podataka – MySQL replikacija obično ne zahteva veliku propustnost, iako ćemo videti kasnije da sirova replikacija može da koristi mnogo veću propustnost od tradicionalnih logičkih replikacija. Takođe možemo da je zaustavimo i pokrenemo kada želimo. Stoga, replikacija je korisna za održavanje kopije podataka na geografski udaljenoj lokaciji, kao što je drugi data centar. Udaljena replika može čak da radi i kada dođe do prekida veze. Međutim, ako želimo da naše replike imaju veoma mali zastoj replikacije, biće nam potrebna stabilna veza sa malim kašnjenjem.
* Balansiranje opterećenja – MySQL replikacija može da nam pomogne da distribuiramo upite za čitanje na više servera. Možemo da uradimo osnovno balansiranje opterećenja sa nekoliko jednostavnih promena u kodu. Možemo da koristimo pojednostavljene, ali i sofisticiranije pristupe. Standarna rešenja za balansiranje opterećenja, kao što su mrežni proizvodi za balansiranje opterećenja, mogu dobro da funkcionišu za distribuciju opterećenja među MySQL serverima.
* Pravljenje backup-ova – Replikacija je dragocena tehnika za pomoć pri pravljenju backup-ova. Međutim, replika nije backup niti je zamena za backup-ove.
* Visoka dostupnost i prevazilaženje grešaka - Replikacija nam može pomoći da izbegnemo da MySQL postane jedinstvena tačka neuspeha u našoj aplikaciji. Dobar sistem za prevazilaženje grešaka koji uključuje replikaciju može značajno da smanjenji vreme zastoja.
* Testiranje MySQL upgrade-ova – Uobičajena je praksa da postavimo repliku sa upgrade-ovanom MySQL verzijom i da je koristimo kako bismo osigurali da naši upiti rade onako kako se očekuje, pre upgrade-a svake instance.

# Kako replikacija radi?

Pre nego što objasnimo detalje podešavanja replikacije, pogledaćemo kako MySQL zapravo replicira podatke. Na visokom nivou, replikacija je jednostavan proces koji se sastoji iz tri dela:

1. Master snima promene svojih podataka u svom binarnom logu. (Ovi zapisi se nazivaju događaji binarnog loga.)
2. Replika kopira događaje binarnog loga mastera u svoj binarni log (eng. relay log).
3. Replika reprodukuje događaje iz binarnog loga, primenjujući promene na svoje sopstvene podatke.

Svaki od ovih koraka je složen. Na slici 1 je detaljnije ilustrovan proces replikacije.



Slika 1

Prvi deo procesa je binarno logovanje na masteru. Neposredno pre nego što se završi svaka transakcija koja ažurira podatke na masteru, master snima promene u svom binarnom logu. MySQL serijski upisuje transakcije u binarnom logu, čak i ako su naredbe u transakcijama isprepletene tokom izvršenja. Nakon što upiše događaje u binarnom logu, master kaže mehanizmu za skladištenje da izvrši transakcije.

Sledeći korak je da replika kopira binarni log mastera na svoj sopstveni hard uređaj, u takozvani relay log (*eng. relay log*). Za početak, ona pokreće radnu nit, koja se naziva U/I radna nit *(eng. I/O slave thread).* U/I nit otvara uobičajenu klijentsku konekciju ka masteru, a zatim pokreće poseban *binlog dump* proces. *Binlog dump* proces čita događaje iz binarnog loga mastera. Ako sustigne mastera, on spava i čeka da master signalizira kada dođu novi događaji. U/I nit upisuje događaje u relay log replike.

SQL radna nit *(eng. SQL slave thread)* upravlja poslednjim delom procesa. Ova nit čita i reprodukuje događaje iz relay loga, ažurirajući tako podatke replike tako da se podudaraju sa podacima mastera. Sve dok ova nit drži korak sa U/I niti, relay log obično ostaje u kešu operativnog sistema. Događaji koje SQL nit izvršava mogu opciono da pređu u binarni log replike, što je korisno za scenarije koje ćemo spomenuti kasnije u radu.

Slika 1 prikazuje samo dve niti za replikaciju koje se izvršavaju na replici, ali takođe postoji i nit na masteru, kao bilo koja konekcija sa MySQL serverom, konekcija koju replika otvara ka masteru kako bi pokrenula nit na masteru.

Ova arhitektura replikacije razdvaja procese pribavljanja i reprodukovanja događaja na replici, što im omogućava da budu asinhroni. Odnosno, U/I nit može da radi nezavisno od SQL niti. Ona takođe postavlja ograničenja u procesu replikacije, od kojih je najvažnije da se replikacija izvršava serijski na replici. Ovo znači da ažuriranja koja se možda izvršavaju paralelno (u različitim nitima) na masteru ne mogu da budu paralelizovana na replici, jer se izvršavaju u jednoj niti. Postoje neka rešenja za ovo, ali većina korisnika i dalje podležu jednonitnom ograničenju.

# Podešavanje replikacije

Podešavanje replikacije je prilično jednostavan proces u MySQL-u, ali postoji mnogo varijacija u osnovnim koracima, u zavisnosti od scenarija. Najosnovniji scenario je sveže instalirani master i replika. Na visokom nivou, postupak je sledeći:

1. Postaviti naloge za replikaciju na svakom serveru.
2. Konfigurisati master i repliku.
3. Dati instrukciju replici da se poveže i replicira sa mastera.

Ovo pretpostavlja da će biti dovoljna podrazumevana podešavanja, što je tačno ako smo upravo instalirali master i repliku i oni imaju iste podatke (podrazumevana mysql baza podataka). Pokazaćemo kako da uradimo svaki korak pod pretpostavkom da se naši serveri zovu server1 (IP adresa 192.168.0.1) i server2 (IP adresa 192.168.0.2). Zatim ćemo objasniti kako da inicijalizujemo repliku sa servera koji je već pokrenut i istražimo preporučenu konfiguraciju replikacije.

## Kreiranje naloga za replikaciju

MySQL ima nekoliko posebnih privilegija koje dopuštaju pokretanje procesa replikacije. Radna U/I nit, koja se izvršava na replici, uspostavlja TCP/IP konekciju sa masterom. Ovo znači da moramo da kreiramo korisnički nalog na masteru i da mu damo odgovarajuće privilegije, tako da U/I nit može da se poveže kao taj korisnik i čita binarni log mastera. Na slici 2 je prikazano kako se kreira korisnički nalog, koji ćemo nazvati repl.



Slika 2

Korisnički nalog kreiramo i na masteru i na replici. Ograničili smo korisnika na lokalnu mrežu, jer nalog za replikaciju ima mogućnost čitanja svih promena na serveru, što ga čini privilegovanim nalogom. (Iako nema mogućnost za SELECT ili promenu podataka, on još uvek može da vidi neke podatke u binarnom logu.)

Korisniku replikacije je zapravo potrebna samo REPLICATION SLAVE privilegija na masteru, a ne i REPLICATION CLIENT privilegija na bilo kom drugom serveru. Postoje dva razloga zašto odobravamo ove privilegije na oba servera:

* Nalogu koji koristimo za nadgledanje i upravljanje replikacijom biće potrebna REPLICATION CLIENT privilegija, i lakše je koristiti isti nalog za obe namene, nego kreirati poseban korisnički nalog za ovu svrhu.
* Ako postavimo nalog na masteru i onda kloniramo repliku od njega, replika će biti postavljena tačno da deluje kao master, u slučaju da želimo da replika i master zamene uloge.

## Konfigurisanje mastera i replike

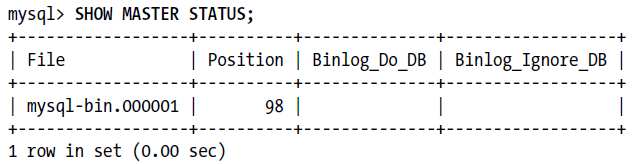
Sledeći korak je omogućivanje nekoliko postavki na masteru, koji pretpostavimo da je nazvan server1. Moramo da omogućimo binarno logovanje i specificiramo ID servera. Treba uneti linije, prikazane na slici 3, u my.cnf fajlu mastera.



Slika 3

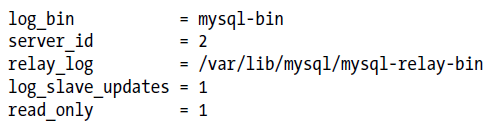
Moramo eksplicitno da specificiramo jedinstveni ID servera. Izabrali smo da korisitimo 10 umesto 1, jer 1 je podrazumevana vrednost koju će server izabrati kada nijedna vrednost nije navedena. (Ovo zavisi od verzije, neke MySQL verzije neće raditi). Stoga, upotreba 1 može jednostavno da prouzrokuje konfuziju i konflikte sa serverima koji nemaju eksplicitne ID-jeve servera. Uobičajena praksa je da se koristi poslednji oktet IP adrese servera, pod pretpostavkom da se ne menja i da je jedinstven. Treba da izaberemo neku konvenciju koja nam ima smisla i da je sledimo.

Ako binarno logovanje nije već specificirano u konfiguracionom fajlu mastera, moraćemo da restartujemo MySQL. Kako bismo proverili da li je binarni log fajl kreiran na masteru, izvršićemo SHOW MASTER STATUS i proveriti da li smo dobili izlaz sličan kao na slici 4. MySQL će dodati neke cifre nazivu fajla, tako da nećemo videti fajl sa tačnim nazivom koji smo naveli.



Slika 4

Replika zahteva konfiguraciju u svom my.cnf fajlu, koja je slična kao na masteru, što je prikazano na slici 5, i takođe moramo da restartujemo MySQL na replici.



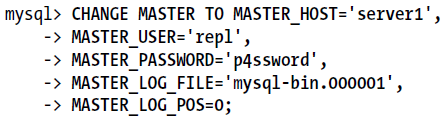
Slika 5

Neke od ovih opcija na slici 5 tehnički su nepotrebne, a za neke samo eksplicitno postavljamo podrazumevane vrednosti. Zapravo, na replici je samo potreban server\_id parametar, ali smo takođe omogućili i log\_bin i dali ekslpicitan naziv fajlu binarnog loga. Po default-u on se naziva po imenu hosta servera, ali ovo može da izazove probleme ako se naziv hosta promeni. Mi korisitimo isti naziv i za master i za replike da bi pojednostavili, ali po želji mogu da se izaberu i različiti.

Takođe smo dodali i dva opciona konfiguraciona parametra: relay\_log (za specificiranje lokacije i naziva relay loga) i log\_slave\_updates (da bi replika evidentirala replicirane događaje u svom sopstvenom binarnom logu). Poslednja opcija prouzrokuje dodatni posao za replike, ali kao što ćemo videti kasnije, imamo dobre razloge za dodavanje ovih opcionih podešavanja na svakoj replici. Ako je moguće, bolje je koristiti read\_only konfiguracionu postavku, koja sprečava bilo šta, osim privilegovanih niti, da menjaju podatke. Međutim, read\_only često nije praktična postavka, posebno za aplikacije koje treba da imaju mogućnosti kreiranja tabela na replikama.

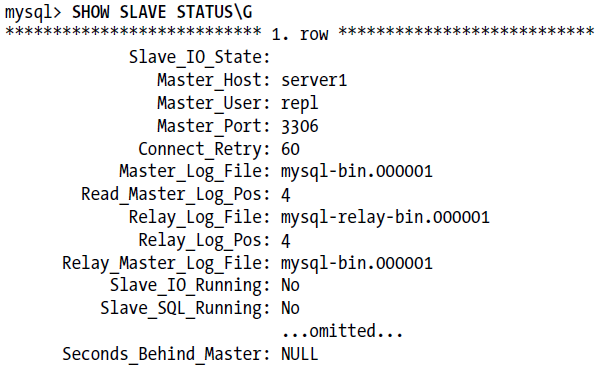
## Pokretanje replike

Sledeći korak je da kažemo replici kako da se poveže sa masterom i da započne reprodukciju njegovih binarnih logova. Ne bi trebalo da koristimo my.cnf fajl za ovo, već CHANGE MASTER TO naredbu. Ova naredba kompletno zamenjuje odgovarajuće my.cnf postavke. Takođe omogućava da u buduće usmerimo repliku prema drugom masteru, bez zaustavljanja servera. Na slici 6 je data osnovna naredba koju mora da pokrenemo na replici kako bismo započeli replikaciju.



Slika 6

Parametar MASTER\_LOG\_POS je postavljen na 0, jer je ovo početak loga. Nakon što pokrenemo ovo, možemo da proverimo izlaz SHOW SLAVE STATUS i vidimo da li su ispravna podešavanja replike, kao što je prikazano na slici 7. Kolone Slave\_IO\_State, Slave\_IO\_Running i Slave\_SQL\_Running pokazuju da se procesi replikacije ne pokreću. Takođe možemo da primetimo da je pozicija loga 4, umesto 0. Ovo je zato što 0 nije u stvari pozicija loga: ona samo znači početak log fajla. MySQL zna da je prvi događaj zapravo na poziciji 4.



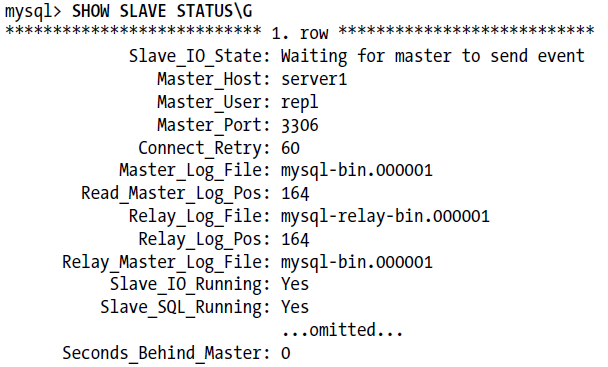
Slika 7

Da bismo započeli replikaciju, treba da izvršimo naredbu kao na slici 8.



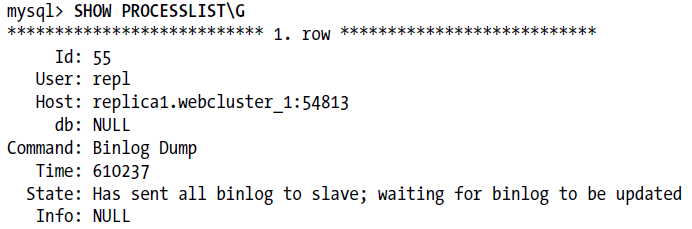
Slika 8

Ova naredba ne bi trebalo da proizvede nikakve greške ili izlaz. Sad ćemo ponovo proveriti SHOW SLAVE STATUS, što je prikazano na slici 9. Sa slike vidimo da su i radna U/I nit i SQL nit pokrenute, a Seconds\_Behind\_Master nije više NULL. U/O nit čeka događaj od mastera, što znači da je pribavila sve binarne logove mastera. Pozicija loga je inkrementirana, što znači da su neki događaji pribavljeni i izvršeni. Ako napravimo promenu na masteru, trebalo bi da vidimo drugačiji fajl i uvećenu poziciju na replici. Takođe trebalo bi se videti i promene u bazi podataka na replici.



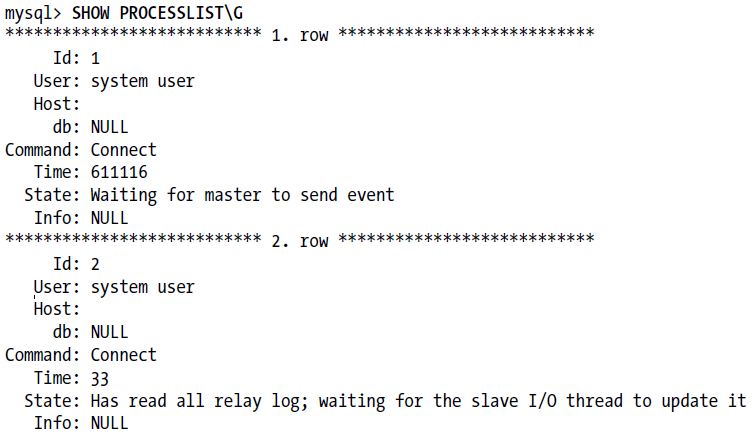
Slika 9

Takođe možemo videti niti replikacije u listi procesa i na masteru i na replici. Na masteru bi trebalo da vidimo konekciju kreiranu od strane U/I niti replike, što je prikazano na slici 10.



Slika 10

Na replici, trebalo bi da vidimo dve niti. Jedna je U/I nit, a druga je SQL nit, što je prikazano na slici 11.



Slika 11

Primer izlaza koji smo prikazali dolazi od servera koji su pokrenuti duže vreme, zbog toga kolona Time za U/I nit na masteru i replici ima veliku vrednost. SQL nit je na replici bio neaktivan 33 sekunde, što znači da se događaji nisu reprodukovali 33 sekunde.

Ovi procesi će se uvek izvršavati pod „system user“ korisničkim nalogom, ali ostale vrednosti kolona mogu da variraju. Na primer, kada SQL nit reprodukuje događaj na replici, kolona Info će prikazivati upit koji se izvršava.

## Inicijalizacija replike sa drugog servera

Prethodne instrukcije za podešavanje su se odnosile kada pokrećemo master i repliku sa podrazumevanim inicijalnim podacima nakon sveže instalacije, tako da smo implicitno imali iste podatke na oba servera i znali smo koordinate binarnog loga mastera. To nije uobičajen slučaj. Obično ćemo imati master koji je postavljen i pokrenut neko vreme i treba da sinhronizujemo novoinstaliranu repliku sa masterom, čak iako nema podatke mastera.

Postoji nekoliko načina za inicijalizaciju ili kloniranje, replike od drugog servera. Oni uključuju kopiranje podataka sa mastera, kloniranje replike iz druge replike i pokretanje replike iz nedavnog backup-a. Potrebne su tri stvari za sinhronizaciju replike sa masterom:

* Snapshot podataka mastera u nekom trenutku u vremenu.
* Trenutni log fajl mastera i pomeraj u bajtovima unutar tog loga u tačnoj tački u kojoj smo snimili snapshot. Ove dve vrednosti nazivamo koordinatama log fajla, zato što one zajedno identifikuju poziciju binarnog loga. Možemo da saznamo koordinate log fajla mastera pomoću komande SHOW MASTER STATUS.
* Binarni log fajlovi mastera od tog vremena do sada.

Načini za kloniranje replike iz drugog servera:

* Pomoću „cold“ kopije – Jedan od najosnovnijih načina za pokretanje replike je isključivanje mastera i kopiranje njegovih fajlova u repliku. Zatim možemo ponovo da pokrenemo master, koji započinje novi binarni log i upotrebimo naredbu CHANGE MASTER TO da bismo pokrenuli repliku na početku ovog binarnog loga. Nedostatak ove tehnike je što moramo da isključimo master dok pravimo kopiju.
* Pomoću „warm“ kopije – Ako koristimo samo MyISAM tabele, možemo da koristimo mysqlhotcopy ili rsync za kopiranje fajlova dok server još uvek radi.
* Pomoću mysqldump-a – Ako koristimo samo InnoDB tabele, možemo da iskorisitmo komandu prikazanu na slici 12 kako bi odložili sve sa servera, učitali sve to u repliku i promenili koordinate replike u odgovarajuću poziciju u binarnom logu mastera. Opcija –single-transaction prouzrokuje da odlaganje pročita podatke kao što su postojali na početku transakcije. Ako ne korisitmo transakcione tabele, možemo da upotrebimo opciju –lock-all-tables da bi dobili konzistentnu kopiju svih tabela.



Slika 12

* Pomoću snapshot-a ili backup-a – Sve dok znamo odgovarajuće koordinate binarnog loga, možemo da koristimo snapshot od mastera ili backup za inicijalizaciju replike (ako koristimo backup, ovaj način zahteva da čuvamo sve binarne logove mastera od trenutka backup-a). Samo treba da povratimo backup ili snapshot na repliku, a zatim iskoristimo odgovarajuće koordinate binarnog loga u CHANGE MASTER TO. Možemo da koristimo LVM, SAN, EBS ili bilo koje druge snapshot-ove.
* Pomoću Percona XtraBackup – Percona XtraBackup je alat otvorenog koda za pravljenje hot backup-ova. On može da kreira backup-ove bez blokiranja rada servera. Možemo da kreiramo replike kloniranjem mastera ili kloniranjem postojeće replike. Potrebno je samo da kreiramo backup (bilo od mastera ili od postojeće replike) i vratimo ga na ciljnu mašinu. Zatim treba pogledati u backup-u za tačnu poziciju za pokretanje replikacije. Ako uzimamo backup od novog mastera replike, možemo da započnemo replikaciju od pozicije navedene u xtrabackp\_binlog\_pos\_innodb fajlu. Ako uzimamo backup od druge replike, možemo da započnemo replikaciju od pozicije navedene u xtrabackup\_slave\_info fajlu. InnoDB Hot Backup ili MySQL Enterprice Backup su drugi načini za inicijalizaciju replike.
* Iz druge replike – Za kloniranje jedne replike iz druge možemo da koristimo bilo koju od spomenutih tehnika snapshot-a ili kopiranja. Takođe, umesto upotrebe SHOW MASTER STATUS za dobijanje koordinata binarnog loga mastera, treba da koristimo SHOW SLAVE STATUS kako bi pronašli poziciju u kojoj se replika izvršavala na masteru kada je urađen snapshot. Najveći nedostatak kloniranja jedne replike iz druge je taj što ako naša replika postane nesinhronizovana sa masterom, kloniraćemo loše podatke.

## Preporučena konfigacija replikacije

Postoji mnogo parametara replikacije i većina njih ima bar neke uticaje na sigurnost podataka i performanse. U ovom delu ćemo pokazati preporučenu konfiguraciju replikacije koja minimizuje mogućnosti za probleme.

Najvažnija postavka za binarno logovanje na masteru je sync\_binlog, što je prikazano na slici 13. Ovo omogućava da MySQL sinhronizuje sadržaj binarnog loga na disku svaki put kada se izvrši transakcija, tako da ne gubimo događaje loga ako dođe do pada. Ako onemogućimo ovu opciju, server će imati malo manje posla, ali bi ulazi binarnog loga mogli biti oštećeni ili promešeni nakon pada servera. Na replici, koja ne treba da se ponaša kao master, ova opcija stvara nepotrebne režijske troškove. Ovo se primenjuje samo na binarni log, a ne na relay log.



Slika 13

Takođe preporučujemo korišćenje InnoDB-a ako ne možemo da tolerišemo oštećene tabele nakon pada. MyISAM je uredu ako oštećenje tabela nije toliko bitno, ali MyISAM tabele će verovatno biti u nekonzistentnom stanju nakon pada servera replike. Velike su šanse da će naredba biti nepotpuno primenjena na jednu ili više tabela i podaci će biti nekonzistentni čak i nakon što popravimo tabele.

Ako koristimo InnoDB, preporučujemo postavljanje opcija na masteru kao na slici 14. Ovo su podrazumevane postavke u MySQL 5.0 i novijim verzijama.



Slika 14

Takođe preporučujemo eksplicitno specificiranje osnovnog imena binarnog loga, kako bi se kreirali uniformni nazivi binarnih logova na svim serverima i sprečile promene naziva binarnog loga ako se promeni hostname servera. Možda nećemo razmišljati da je problem automatsko imenovanje binarnih logova nakon promene imena hosta servera, ali ovo prouzrokuje mnogo problema kada pomeramo podatke između servera, kloniramo nove replike, vraćamo backup-ove i tako dalje. Da bismo ovo izbegli, treba da navedemo argument log\_bin opciji, opciono sa apsolutnom putanjom, ali svakako sa osnovnim nazivom, kao što je prikazano na slici 15.



Slika 15

Takođe preporučujemo korišćenje apsolutne putanje za lokaciju relay loga, kao na slici 16. Opcija relay\_log sprečava nazivanje replay log fajla u zavisnosti od naziva hosta, čime se izbegavaju isti problemi kao što smo ranije spomenuli koji se mogu dogoditi na masteru, a navođenje apsolutne putanje ka logovima izbegava greške u različitim verzijama MySQL-a koje mogu da izazovu da se relay logovi kreiraju na neočekivanoj lokaciji. Opcija skip\_slave\_start će sprečiti da se replika automatski startuje nakon pada, što može da nam da šansu da popravimo server ako ima probleme. Ako se replika automatski startuje nakon pada i nalazi se u nekonzistentnom stanju, to može da izazove toliko dodatnih oštećenja da ćemo morati da bacimo njene podatke i započnemo sa novim pokretanjem. Opcija read\_only sprečava većinu korisnika da menjaju neprivremene tabele. Izuzeci su replikaciona SQL nit i niti sa SUPER privilegijom.



Slika 16

Čak iako omogućimo sve opcije koje smo predložili, replika može lako da se pokvari nakon pada, jer relay logovi i fajl master.info nisu bezbedni za padove. Oni se čak po defaultu ni ne čiste na disk, a ne postoji ni konfiguraciona opcija koja bi kontrolisala takvo ponašanje do MySQL 5.5. Treba da omogućimo te opcije kao na slici 17, ako nam ne smetaju troškovi dodatnih poziva fsync().



Slika 17

Ako je replika mnogo zaostala iza svog mastera, radna U/I nit može da upiše mnogo relay logova. Replikaciona SQL nit će ih ukloniti čim završi s njihovom reprodukcijom (ovo možemo da promenimo pomoću opcije realy\_log\_purge), ali ako je ona daleko iza, U/I nit može da napuni disk. Rešenje ovog problema je konfiguraciona promenljiva relay\_log\_space\_limit. Ako ukupna veličina svih relay logova raste više od veličine ove promenljive, U/I nit će se zaustaviti i čekaće da SQL nit oslobodi još malo prostora na disku.

Ovo može da bude skriven problem. Ako replika nije pribavila sve relay logove od mastera, ovi logovi se mogu zauvek izgubiti ako master padne. Ako nismo zabrinuti za prostor na disku, možemo da omogućimo da replika koristi onoliko prostora koliko je potrebno za relay logove. Zbog toga nismo uključili relay\_log\_space\_limit postavku u našu preporučenu konfiguraciju.

# Replikacija ispod haube

U ovom delu ćemo detaljnije objasniti neke osnove replikacije. Videćemo kako replikacija stvarno funkcioniše, koje su prednosti i mane kao rezultat toga i ispitaćemo neke naprednije konfiguracione opcije replikacije.

## Logička replikacija

MySQL 5.0 i starije verzije podržavaju samo replikaciju zasnovanu na naredbi (*eng. statement-based*) (koja se takođe naziva i logičkom replikacijom). Logička replikacija radi tako što snima upit koji menja podatke na masteru. Kada replika čita događaj iz relay loga i izvrši ga, ona ponovo izvršava stvarni SQL upit koji je master izvršio. Ovo ima i prednosti i mane.

Najočiglednija prednost je ta što je prilično jednostavno za implementaciju. Jednostavno logovanje i reprodukovanje bilo koje naredbe koja menja podatke teoretski će održati repliku sinhronizovanom sa masterom. Još jedna prednost logičke replikacije je ta što su događaju binarnog loga prilično kompaktni. Logička replikacija ne koristi veliku propustnost – upit koji ažurira gigabajte podataka može da koristi samo nekoliko desetina bajtova u binarnom logu. Takođe, alat mysqlbinlog je najprikladniji za upotrebu sa logovanjem zasnovanim na naredbi.

Međutim, u praksi logička replikacija nije toliko jednostavna kako se čini, jer mnoge promene na masteru mogu da zavise od drugih faktora pored samog teksta upita. Na primer, naredbe će se izvršavati u različito vreme na masteru i na replici. Kao rezultat toga, format binarnog loga MySQL-a uključuje više od samog teksta upita; on takođe prenosi nekoliko bitova metapodataka, kao što je tekući timestamp. Uprkos tome, postoje neke neredbe koje MySQL ne može da replicira ispravno, kao što su upiti koji koriste funkciju CURRENT\_USER(). Stored procedure i trigeri su takođe problematični kod replikacije zasnovane na naredbi.

Drugi problem kod replikacije zasnovane na naredbi je što modifikacije mora da budu serijalizovane. Za ovo je potrebno značajno više zaključavanja. Ne rade svi mehanizmi za skladištenje sa replikacijom zasnovanom na naredbi.

## Sirova replikacija

MySQL 5.1 ima dodatu podršku za replikaciju zasnovanu na redu (*eng. row-based*), koja snima stvarne promene podataka u binarnom logu i slična je načinu na koji većina drugih baza podataka implementira replikaciju. Ova šema ima nekoliko prednosti i mana. Najveće prednosti su da MySQL može da replicira svaku naredbu ispravno i neke naredbe mogu da se repliciraju mnogo efikasnije.

MySQL može da replicira neke promene efikasnije koristeći sirovu replikaciju, jer replika ne mora da ponavlja upite koji su promenili redove na masteru. Ponavljanje nekih upita može da bude veoma skupo. Na primer, na slici 18 je prikazan upit koji sumira podatke iz veoma velike tabele u manju tabelu.



Slika 18

Pretpostavimo da postoje samo tri jedinstvene kombinacije col1 i col2 u tabeli enormous\_table. Ovaj upit će skenirati mnogo redova u izvornoj tabeli, ali će rezultat biti samo tri reda u odredišnoj tabeli. Repliciranje ovog događaja kao naredbe će učiniti da replika ponavlja ceo ovaj posao kako bi generisala nekoliko redova, ali repliciranje pomoću sirove replikacije će biti jeftino na replici. U ovom slučaju, sirova replikacija je mnogo efikasnija.

Događaj prikazan na slici 19 je mnogo jeftiniji da se replicira pomoću logičke replikacije. Korišćenje sirove replikacije za ovaj upit bilo bi skuplje jer on menja svaki red: svaki red bi morao da bude upisan u binarnom logu, što događaj binarnog loga čini veoma velikim. Ovo bi mnogo opteretilo master tokom logovanja i replikacije, a sporije logovanje može smanjiti konkurentnost.



Slika 19

Pošto nijedan format nije savršen za svaku situaciju, MySQL može dinamički da prelazi između replikacije zasnovane na naredbi i na redu. Po default-u, on koristi replikaciju zasnovanu na naredbi, ali kada detektuje događaj koji ne može da se pravilno replicira pomoću logičke replikacije, on prelazi na sirovu replikaciju. Takođe i mi možemo po potrebi da kontrolišemo format postavljanjem promenljive sesije binlog\_format.

## Poređenje logičke i sirove replikacije

Teoretski, sirova replikacija je verovatno bolja u svim slučajevima. Ali njena implementacija je dosta nova, tako da se nije mnogo koristila u specijalnim slučajevima kako bi podržala sve operativne potrebe MySQL administratora i zbog toga je za neke ljude još uvek neuobičajena. U nastavku je data potpunija diskusija o prednostima i manama svakog formata kako bi mogli da odlučimo šta je pogodnije za naše potrebe:

* Prednosti logičke replikacije:

Logička replikacija funkcioniše u mnogim slučajevima kada je šema drugačija na masteru i replici. Na primer, može da radi u mnogim slučajevima kada tabele imaju ali kompatibilne tipove podataka, drugačiji raspored kolona, i tako dalje. Ovo olakšava izvršenje promena šeme na replici i zatim promovisanje toga na masteru, smanjujući vreme zastoja.

Logička replikacija uglavnom dopušta više operativne fleksibilnosti.

Proces primene replikacije u logičkoj replikaciji je normalno SQL izvršenje. Ovo znači da se sve promene na serveru odvijaju kroz dobro razumljiv mehanizam i lako je pregledati i odrediti šta se dešava ako nešto ne radi kako se očekuje.

* Nedostaci logičke replikacije:

Lista stvari koje se ne mogu ispravno replicirati pomoću logovanja baziranog na naredbi je toliko velika da će svaka instalacija verovatno naići na bar jednu od njih. Konkretno, bilo je mnogo bagova koji su uticali na stored procedure, trigere i tako dalje u 5.0 i 5.1 verzijama servera – toliko mnogo bagova da se način na koji se oni repliciraju menjao par puta u pokušajima da se učini boljim. Ako koristimo trigere ili stored procedure, ne treba da koristimo logičku replikaciju kako bismo bili sigurni da nećemo naići na probleme.

Takođe postoji mnogo problema sa privremenim tabelama, mešavinama mehanizama za skladištenje, specifičnim SQL konstrukcijama, nedeterminističkim naredbama i tako dalje.

* Prednosti sirove replikacije:

Postoji mnogo malo slučajeva koji ne rade u sirovoj replikaciji. Ona ispravno radi sa svim SQL konstrukcijama, sa trigerima, stored procedurama i tako dalje. Ona samo ne radi u slučajevima kao što su promena šeme na replici.

Ona takođe stvara mogućnosti za smanjeno zaključavanje, jer ne zahteva toliko strogu serijalizaciju.

Sirova replikacija funkcioniše tako što loguje podatke koji su promenjeni, tako da je binarni log zapis onoga što se promenilo na masteru. Ne moramo da gledamo naredbu i nagađamo da li je promenila neke podatke. Na neki način zapravo znamo više o tome šta je promenjeno na serveru i imamo bolju evidenciju o promenama. Takođe, u nekim slučajevima sirovi binarni logovi snimaju koji su podaci nekada bili, tako da oni mogu da budu korisniji za neke vrste oporavka podataka.

Sirova replikacija u mnogim slučajevima može da bude manje intenzivna na procesoru, jer nema potrebe za planiranjem i izvršenjem upita na isti način kao što to radi logička replikacija.

Sirova replikacija može da nam pomogne da brže pronađemo i rešimo nekonzistentnost podataka u nekim slučajevima. Na primer, logička replikacija neće uspeti, ako ažuriramo red na masteru a on ne postoji na replici, ali će sirova replikacija baciti grešku ili će se zaustaviti.

* Nedostaci sirove replikacije:

Naredba nije uključena u log događaja, tako da može da bude teško da se shvati šta je SQL izvršio. Ovo je važno u mnogim slučajevima, pored saznanja o promenama reda.

Promene replikacije se primenjuju na replike na potpuno različite načine – ne izvršava se SQL. Zapravo, postupak primene sirovih promena je kao crna kutija bez vidljivosti o tome šta server radi, tako da kada stvari ne funkcionišu kako treba, može biti teško da se problem reši. Na primer, ako replika izabere neefikasan način za pronalaženje redova, mi to ne možemo da primetimo.

Ako imamo više nivoa replikacionih servera, a svi su konfigurisani za logovanje zasnovano na redu, naredba koja se izvršava dok je promenljiva formata na nivou sesije binlog\_format postavljena na STATEMENT evidentiraće se kao naredba na serveru odakle potiče, ali tada replike prvog nivoa mogu da prenesu događaj u formatu zasnovanom na redu do daljih replika u lancu. Odnosno, naše željeno logovanje zasnovano na naredbi biće promenjeno u logovanje zasnovano na redu tokom propagiranja kroz replikacionu topologiju.

Sirovo logovanje ne može da upravlja nekim stvarima kojima logičko logovanje može, kao što su promene šeme na replikama.

Sirova replikacija će se ponekad zaustaviti u slučajevima gde bi se logička replikacija nastavila, na primer kada u replici nedostaje red koji bi trebalo da bude promenjen. U svakom slučaju, ovo je podesivo pomoću opcije slave\_exec\_mode.

## Fajlovi replikacije

U ovom odeljku ćemo videti neke fajlove koje replikacija koristi. Već znamo za binarni log i replay log, ali postoji i nekoliko drugih fajlova. Gde ih MySQL smeša uglavnom zavisi od naših podešavanja konfiguracije. Različite MySQL verzije ih smeštaju u različitim folderima po defaultu. Verovatno ćemo ih pronaći ili u direktorijumu podataka ili u direktorijumu koji sadrži .pid fajl servera. Ti fajlovi su:

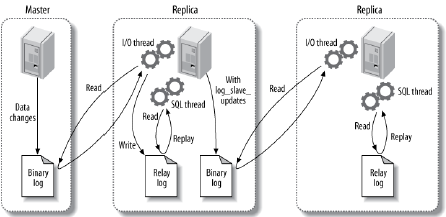
* mysql-bin.index – Server, na kome je omogućeno binarno logovanje, takođe ima fajl koji se zove isto kao binarni log, ali sa sufiksom .index. Ovaj fajl prati fajlove binarnih logova koji postoje na disku. Svaka linija u fajlu sadrži naziv fajla binarnog loga. MySQL se oslanja na ovaj index fajl i neće prepoznati fajl binarnog loga ako nije spomenut ovde.
* mysql-relay-bin.index – Ovaj fajl služi istoj svrsi za relay logove kao index fajl binarnog loga za binarne logove.
* master.info – Ovaj fajl sadrži informacije koje su potrebne replici kako bi se povezala sa masterom. Tekstualnog je formata (jedna vrednost po liniji). Ne treba da ga brišemo, jer naša replika neće znati kako da se poveže sa masterom nekon restarta. Ovaj fajl sadrži šifru korisnika replikacije, tako da možemo da ograničimo njegove dozvole.
* relay-log.info – Ovaj fajl sadrži trenutni binarni log replike i koordinate relay loga (na primer, pozicija replike na masteru). Ne treba da ga brišemo, jer će replika da zaboravi odakle se replicirala nakon restarta i može da pokuša da reprodukuje naredbe koje su već izvršene.

Ovi fajlovi su prilično sirov način za snimanje MySQL-ove replikacije i stanja logovanja. Međutim, oni se ne upisuju sinhrono, tako da ako se ugasi server a ovi podaci nisu još ispražnjeni na disk, mogu da budu netačni kada se server ponovo startuje. Ovo je poboljšano u MySQL 5.5.

.index fajlovi interaguju sa drugim postavkama, kao što je expire\_logs\_days koja specificira kako MySQL čisti zastarele binarne logove. Ako mysql-bin.index fajlovi navode fajlove koji ne postoje na disku, automatsko čišćenje neće raditi u nekim verzijama MySQL-a, čak neće raditi ni naredba PURGE MASTER LOGS. Rešenje za ovaj problem je korišćenje MySQL servera za upravljanje binarnim logovima.

## Slanje replikacionih događaja drugim replikama

Opcija log\_slave\_updates omogućava nam da koristimo repliku kao master drugih replika. Ona daje instrukciju MySQL-u da upiše događaje koje replikaciona SQL nit izvršava u svom binarnom logu, koji onda replike mogu da pribave i izvrše. Slika 20 ilustruje ovo.



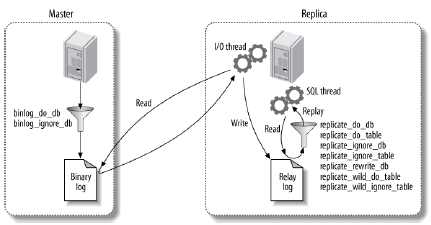
Slika 20

U ovom scenariju, promena na masteru prouzrokuje da se događaj upiše u njegov binarni log. Prva replika onda pribavlja i izvršava događaj. U ovom trenutku je obično život događaja završen, ali pošto je log\_slave\_updates omogućen, replika ga upisuje u svom binarnom logu. Sada druga replika može da pribavi događaj u svom relay logu i da ga izvrši. Ova konfiguracija znači da se promene na originalnom masteru mogu propagirati do replika koje nisu direktno prikačene na njega. Treba postaviti log\_slave\_updates jer omogućava povezivanje replike bez restartovanja servera.

Kada prva replika upiše događaj binarnog loga iz mastera u svoj binarni log, taj događaj će se sigurno nalaziti na različitoj poziciji u logu od pozicije na masteru – odnosno, može da bude u drugačijem log fajlu ili na drugačijoj numeričkoj poziciji unutar log fajla. Ovo znači da ne možemo da pretpostavimo da će svi serveri koji su u istoj logičkoj tački replikacije imati iste koordinate loga. Ovo komplikuje obavljanje nekih zadataka, kao što su promena replika na drugačijem masteru ili promovisanje replike da bude master.

## Filteri replikacije

Opcije filtriranja replikacije omogućavaju da repliciramo samo deo podataka servera. Postoje dve vrste replikacionih filtera: oni koji filtriraju događaje iz binarnog loga na masteru i one koji filtriraju događaje koji dolaze iz relay loga replike. Ova dva tipa su ilustrovana na slici 21.



Slika 21

Opcije koje kontrolišu filtriranje binarnog loga su binlog\_do\_db i binlog\_ignore\_db. Na replici, replicate\_\* opcije filtriraju događaje dok ih replikaciona SQL nit čita iz relay loga. Možemo da repliciramo ili ignorišemo jednu ili više baza podataka, prepišemo jednu bazu podataka u drugu bazu podataka i repliciramo ili ignorišemo tabele u zavisnosti od LIKE sintakse podudaranja šablona.

Najvažnija stvar koju treba da shvatimo oko ovih opcija je da \*\_do\_db i \*\_ignore\_db opcije, bilo na masteru ili na replici, ne rade onako kako bismo mogli da očekujemo. Možemo misliti da one filtriraju po nazivu baze podataka objekta, ali one zapravo filtirraju na trenutnoj zadatoj bazi podataka. Odnosno, ako izvršimo narebe na masteru kao na slici 22. Parametri \*\_do\_db i \*\_ignore\_db će filtrirati DELETE naredbe na bazi podatak test, a ne sakila. Ovo obično nije ono što želimo i može da izazove da se pogrešne naredbe repliciraju ili ignorišu. Ovi parametri se koriste, ali su ograničeni i retki i treba da budemo pažljivi sa njima. Ako koristimo ove parametre, veoma je lako da replikacija ne bude sinhronizovana i da ne uspe.



Slika 22

Uopšteno, replikacioni filteri su problem koji čeka da se desi. Na primer, pretpostavimo da želimo da sprečimo da se privilegovane promene propagiraju na replike, što je ubičajen cilj. Replikacioni filteri na sistemskim tabelama će sigurno sprečiti repliciranje GRANT naredbi, ali će takođe sprečiti i repliciranje događaja i rutina. Takve nepredviđene posledice su razlog zašto treba da budemo pažljivi sa filterima. Bolja je ideja da sprečimo repliciranje određenih naredbi, obično pomoću SET SQL\_LOG\_BIN=0, mada takva praksa ima svoje opasnosti. Generalno, replikacione filtere treba da koristimo veoma pažljivo i samo ako su nam stvarno potrebni, jer oni mogu lako da prekinu replikaciju i izazovu probleme koji mogu da se manifestuju kada nam je to najmanje potrebno, kao što je tokom oporavka od katastrofe.

# Replikacione topologije

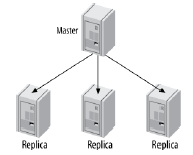
Možemo da postavimo MySQL replikaciju za skoro bilo koju konfiguraciju mastera i replika, uz ograničenje da data instanca MySQL replike može da ima samo jednog mastera. Mnoge složene topologije su moguće, ali čak i jednostavne mogu da budu veoma fleksibilne. Jedna topologija može da ima više različitih upotreba.

Već smo videli kako da podesimo master sa jednom replikom. U ovom odeljku ćemo videti neke druge uobičajene topologije i razmotriti njihove prednosti i mane. Osnovna pravila su:

* Instanca MySQL replike može da ima samo jednog mastera.
* Svaka replika mora da ima jedinstven ID servera.
* Master može da ima više replika.
* Replika može da propagira promene od mastera i da bude master drugim replikama, ako je omogućeno log\_slave\_updates.

## Master i više replika

Pored osnovne postavke dva servera master-replika koju smo već pokazali, master sa više replika je najjednostavnija replikaciona topologija. Zapravo, ona je jednostavna kao i osnovna postavka, jer replike ne interaguju međusobno, one su povezane samo sa masterom. Ovo je ilustrovano na slici 23.



Slika 23

Ova konfiguracija je najkorisnija kada imamo malo upisa i mnogo čitanja. Možemo da propagiramo čitanja na bilo koji broj replika, sve dok replike ne opterećuju previše master ili mrežna propustnost od mastera do replika ne postane problem. Možemo da postavimo više replika odjednom ili da dodajemo replike po potrebi, korišćenjem istih koraka koje smo pokazali ranije.

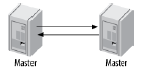
Iako je ovo veoma jednostavna topologija, ona zadovoljava mnoge potrebe, kao što su:

* Korišćenje različitih replika za različite uloge.
* Postavljanje jedne od replika kao master pripravnosti (čekanja), bez saobraćaja osim replikacije.
* Stavljanje jedne od replika u udaljeni data centar za oporavak od katastrofe.
* Vremensko odlaganje jedne ili više replika radi oporavka od katastrofe.
* Korišćenje jedne od replika za pravljenje backup-a ili za razvoj i postavljanje servera.

Jedan od razloga zašto je ova topologija popularna je taj što ona izbegava mnoge kompleksnosti koje dolaze sa drugim konfiguracijama. Na primer, lako je uporediti jednu repliku sa drugom u odnosu na poziciju binarnog loga na masteru, jer će sve biti iste. Drugim rečima, ako zaustavimo sve replike u istoj logičkoj tački u replikaciji, sve će one čitati sa iste fizičke pozicije u logovima mastera, Ovo je dobra osobina koja pojednostavljuje mnoge administrativne zadatke, kao što je promovisanje replike da bude master. Ovo svojstvo ima samo među braćama replika. Komplikovanije je poređenje pozicije logova između servera koji nisu u direktnoj master-replika relaciji ili nisu braća.

## Master-master u aktivan-aktivan modu

Replikacija master-master (poznata kao i dvosmerna replikacija) uključuje dva servera, oba konfigurisana i kao master i kao replika drugog – drugim rečina, par komastera. Postavka je prikazana na slici 24.



Slika 24

Replikacija master-master u aktivan-aktivan mod se uglavnom koristi za specijalne namene. Jedna moguća upotreba je za geografski odvojene kancelarije, gde svaka kancelarija treba da ima sopstvenu kopiju podataku na kojoj može lokalno da upisuje.

Najveći problem sa takvom konfiguracijom je kako izaći na kraj sa konfliktnim promenama. Velika je lista mogućih problema prouzrokovanih posedovanjem dva mastera koji mogu da upisuju. Problemi obično nastaju kada upit menja isti red istovremeno na oba servera ili dodaje u tabelu sa AUTO\_INCREMENT kolonom istovremeno na oba servera.

MySQL 5.0 je dodao: podešavanja auto\_increment\_increment i auto\_increment\_offset. Ova podešavanja omogućavaju serverima da automatski generišu nekonfliktne vrednosti za INSERT upite. Međutim, dopuštanje upisa oba mastera je idalje veoma opasno. Ažuriranja koja se dešavaju u različitom redosledu na dve mašine mogu još izazvati da podaci postanu nesinhronizovani. Na primer, pretpostavimo da imamo jednu kolonu i jedan red u tabeli koji sarži vrednost 1. Sada pretpostavimo da se na prvom komasteru izvršava naredba kao na slici 25, a na drugom naredba kao na slici 26.



Slika 25

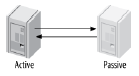


Slika 26

Jedan server će imati vrednost 4, a drugi 3, a ipak uopšte nema grešaka u replikaciji. To što podaci postaju nesinhronizovani je samo početak. Šta ako se zaustavi normalna replikacija sa greškom, ali aplikacije nastavljaju da upisuju na oba servera? Ne možemo samo da kloniramo jedan server od drugoga, jer svaki od njih će imati promene koje moramo da kopiramo drugom. Ovoj problem će se veoma teško rešiti. Ako pažljivo podesimo master-master aktivan-aktivan konfiguraciju, možda sa dobro particionisanim podacima i privilegijama, i ako stvarno znamo šta radimo, možemo da izbegnemo neke od ovih problema. Uopšteno, omogućavanje upisa na oba servera prouzrokuje više problema nego koristi.

## Master-master u aktivan-pasivan modu

Ovo je varijacija master-master replikacije koja izbegava zamke o kojima smo diskutovali i veoma je snažan način za dizajniranje sistema koji su tolerantni na otkaze i visoko dostupni. Glavna razlika je u tome što je jedan od servera read-only pasivan server, kao što je prikazano na slici 27.



Slika 27

Ova konfiguracija nam omogućava da lako i jednostavno menjamo aktivne i pasivne uloge servera, jer su konfiguracije servera simetrične. Ovo nam omogućava lako prevazilaženje grešaka. Takođe nam omogućava održavanje, optimizaciju tabela, nadogradnju operativnog sistema (ili aplikacije ili hardvera) i obavljanje drugih zadataka bez ikakvih zastoja.

Na primer, pokretanje ALTER TABLE naredbe zaključava celu tabelu, blokirajući čitanja i upise u njoj. Ovo može da traje dugo i poremeti uslugu. Međutim, master-master konfiguracija nam omogućava da zaustavimo replikacione niti na aktivnom serveru (tako da ne obrađuje ažuriranja sa pasivnog servera), izmenimo tabelu na pasivnom serveru, zamenimo uloge i restartujemo replikaciju na prethodno aktivnom serveru. Taj server onda čita svoj relay log i izvršava istu ALTER TABLE naredbu.

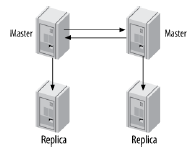
Master-master par ćemo konfigurisati izvršenjem sledećih koraka na oba servera, tako da oni imaju simetričnu konfiguraciju:

* Osigurati da serveri imaju potpuno iste podatke.
* Omogućiti binarno logovanje, izabrati jedinstvene ID-jeve servera i dodati replikacione naloge.
* Omogućiti logovanje ažuriranja replike. Ovo je važno za prevazilaženje i oporavak od greške.
* Opciono konfigurisati pasivan server samo za čitanje kako bi se sprečile promene koje mogu da budu u konfliktu sa promenama na aktivnom serveru.
* Startovati svaku instancu MySQL servera.
* Konfigurisati svaki server kao repliku drugog, počevši od novokreiranog binarnog loga.

Kada dođe do promene na aktivnom serveru, promena se upisuje u njegov binarni log i prenosi se preko replikacije u relay log pasivnog servera. Pasivni server izvršava upit i upisuje događaj u svom binarnom logu, jer je omogućeno log\_slave\_updates. Zatim aktivan server ignoriše događaj, jer se ID servera u događaju poklapa sa njegovim.

## Master-master sa replikama

U ovoj konfiguraciji se dodaje jedna ili više replika svakom ko-masteru, kao što je prikazano na slici 28.



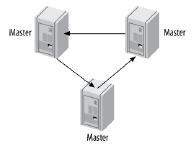
Slika 28

Prednost ove konfiguracije je dodatna redundantnost. U geografski distribuiranoj replikacionoj topologiji, ovo uklanja jedinstvenu tačku neuspeha na svakoj lokaciji. Takođe možemo da učitamo upite sa intenzivnim čitanjem u replike.

Ako lokalno koristimo master-master topologiju za brzo prevazilaženje grešaka, ova konfiguracija je još uvek korisna. Moguće je promovisanje jedne od replika za zamenu palog mastera, iako je malo složenije. Isto važi i za premeštanje jedne od replika da ukazuje na drugi master.

## Prsten replikacija

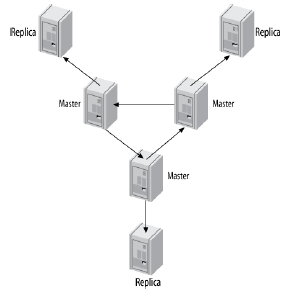
Master-master konfiguracija je samo specijalan slučaj prsten replikacione konfiguracije, koja je prikazana na slici 29. Prsten ima tri ili više mastera. Svaki server je replika servera pre njega u prstenu, a master servera nakon njega. Ova topologija se takođe naziva i kružna replikacija.



Slika 29

Prsteni nemaju neke od ključnih prednosti master-master postavke, kao što su simetrična konfiguracija i lako prevazilaženje greške. Oni takođe upotpunosti zavise od svakog čvora u prstenu koji je dostupan, što mnogo povećava verovatnoću pada celog sistema. Ako uklonimo jedan od čvorova iz prstena, svi događaji replikacije koji su nastali na tom čvoru mogu da uđu u beskonačnu petlju: oni će zauvek kružiti kroz lanac servera, jer jedini server koji će filtrirati događaj na osnovu njegovog ID-a servera je server koji ga je kreirao. Generalno, prstene treba izbegavati.

Neke od rizika prsten replikacije možemo da ublažimo dodavanjem replika kako bismo obezbedili redundantnost na svakoj lokaciji, kao što je prikazano na slici 30. Ovo samo štiti od rizika od neuspeha servera. Gubitak struje ili bilo koji drugi problem koji utiče na bilo kakvu vezu između lokacija će idalje prekinuti ceo prsten.



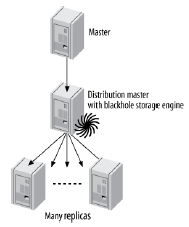
Slika 30

## Master, distribucioni master i replikacije

Replike mogu dosta da opterete master ako ih ima mnogo. Svaka replika kreira novu nit na masteru, koja izvršava posebnu binlog dump komandu. Ova komanda čita podatke iz binarnog loga i šalje ih replici. Ovaj posao se ponavlja za svaku repliku.

Ako postoji mnogo replika i postoji posebno veliki događaj binarnog loga, opterećenje mastera može značajno da se poveća. Masteru čak može i da ponestane memorije i padne jer sve replike traže isti ogroman događaj u isto vreme. Sa druge strane, ako sve replike zahtevaju različite događaje binarnog loga koji više nisu u kešu fajl sistema, to može da izazove mnoštvo traženja diska, što može takođe da ometa performanse mastera.

Iz tog razloga, ako nam je potrebno mnogo replika, često je dobra ideja ukloniti učitavanje sa mastera i koristiti distribucioni master. Distribucioni master je replika čija je jedina namena da čita i servira binarne logove od mastera. Više replika može da se poveže sa distribucionim masterom, koji originalni master izoluje od opterećenja. Da bi uklonili posao izvršenja upita na distribucioni master, možemo da promenimo njegove tabele u Blackhole mehanizmu skladištenja, što je prikazano na slici 31.



Slika 31

Teško je reći tačno koliko replika master može da hendluje pre nego što mu je potreban distribucioni master. Grubo pravilo je da ako master radi skoro punim kapacitetom, ne treba staviti više od 10 replika na njemu. Ako postoji veoma malo aktivnosti upisa ili repliciramo samo deo tabela, master može da opslužuje mnogo više replika. Uz to, ne moramo da se ograničimo na samo jednog distribucionog mastera. Možemo da koristimo nekoliko ako treba da repliciramo ogroman broj replika ili čak možemo da koristimo piramidu distribucionih mastera.

Takođe možemo da koristimo distribucioni master za druge namene, kao što su primena filtera ili prepisivanje pravila u događaje binarnog loga. Ovo je mnogo efikasnije od ponavljanja logovanja, prepisivanja i filtriranja na svakoj replici.

Ako koristimo Blackhole tabele na distribucionom masteru, on će moći da uslužuje više replika nego što bi inače mogao. Distribucioni master će izvršavati upite, ali upiti će biti veoma jeftini, jer Blackhole tabele neće imati nikakve podatke. Mana Blackhole tabela je što one imaju greške, kao što je da u nekim okolnostima zaborave da stave autoinkrementalne ID-jeve u svoje binarne logove, tako da treba da budemo oprezni sa Blackhole tabelama ako ih koristimo.

Možemo da osiguramo da sve tabele na distribucionom masteru koriste Blackhole mehanizam skladištenja postavljanjem storage\_engine opcije servera, kao na slici 32. Ovo će uticati samo na CREATE TABLE naredbe koje eksplicitno ne specificiraju mehanizam skladištenja. Možemo da onemogućimo InnoDB pomoću opcije skip\_innodb, ali ne možemo da onemogućimo MyISAM.

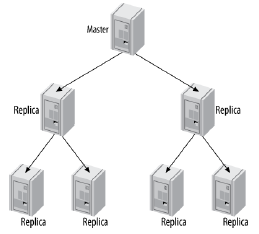


Slika 32

Drugi veliki nedostatak su potreškoće prilikom zamene mastera jednom od replika. Teško je promovisati jednu od replika na njegovo mesto, jer master osigurava da one skoro uvek imaju drugačije koordinate binarnog loga od originalnog mastera.

## Stablo ili piramida

Ako repliciramo master na velikom broju replika, upotreba dizajna piramide može da bude lakša za upravljanje, što je ilustrovano na slici 33.



Slika 33

Prednost ovog dizajna je u tome što smanjuje opterećenje na masteru, kao što je to radio distribucioni master u prethodnom odeljku. Nedostatak je u tome što svaki neuspeh u srednjem nivou će uticati na više servera, što se ne bi desilo da su sve replike direktno povezane na master. Takođe, što više imamo srednjih nivoa, teže je i komplikovanije rešavanje neuspeha.

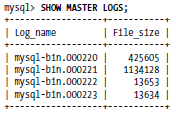
# Administracija i održavanje replikacije

Kada jednom budemo postavili replikaciju, monitoring i administriranje naše replikacione topologije će biti redovan posao, bez obzira na to koliko imamo servera. Trebalo bi da pokušamo da automatizujemo ovaj posao koliko je to moguće.

## Monitoring replikacije

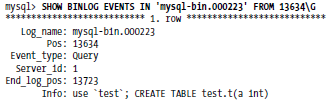
Replikacija povećava složenost monitoringa MySQL-a. Iako se replikacija zapravo odvija i na masteru i na replici, najveći deo posla se vrši na replici i upravo se tu javljaju najčešći problemi. Da li rade sve replike? Da li je neka replika imala greške? Kolko je zaostala najsporija replika? MySQL pruža većinu informacija koje su nam potrebne da odgovorimo na ova pitanja, ali automatizacija procesa nadgledanja i izrade robustne replikacije je ostavljena nama.

Na masteru možemo da koristimo naredbu SHOW MASTER STATUS kako bismo videli trenutnu poziciju binarnog loga mastera i konfiguraciju. Takođe možemo da pitamo mastera koji sve binarni logovi postoje na disku pomoću naredbe SHOW MASTER LOGS, kao što je prikazano na slici 34.



Slika 34

Takođe možemo da vidimo replikacione događaje u binarnom logu pomoću komande SHOW BINLOG EVENTS. Na primer, nakon pokretanja prethodne komande, kreirali smo tabelu na drugom neiskorišćenom serveru. Pošto smo znali da je ovo bila jedina naredba koja je promenila podatke, znali smo da je pomeraj naredbe u binarnom logu 13634, tako da smo mogli da vidimo to kao na slici 35.



Slika 35

## Merenje zaostajanja replikacije

Jedna od najčešćih stvari koju treba da nadgledamo je koliko replika zaostaje iza mastera. Iako kolona Seconds\_behind\_master u SHOW SLAVE STATUS teorijski pokazuje zaostajanje replikacije, ovo nije uvek tačno, iz više razloga:

* Replika izračunava Seconds\_behind\_master tako što upoređuje trenutni timestamp servera sa snimljenim timstampom u binarnom logu događaja, tako da replika ne može čak ni da prijavi svoje zaostajanje ako se ne obrađuje upit.
* Replika će obično prijaviti NULL ako se procesi replikacije ne izvršavaju.
* Neke greške (na primer, neusklađeno podešavanje max\_allowed\_packet između mastera i replike ili nestabilne mreže) mogu da prekinu replikaciju ili zaustave replikacione niti, ali Seconds\_behind\_master će prijaviti 0 umesto da ukaže na grešku.
* Replika ponekad ne može da izračuna zaostajanje čak i ako se procesi replikacije izvšavaju. Ako se ovo dogodi, replika može da prijavi ili 0 ili NULL.
* Veoma duga transakcija može da izazove promenu prijavljenog kašnjenja. Na primer, ako imamo transakciju koja ažurira podatke, ostaje otvorena sat vremena, a zatim se izvrši, ažuriranje će ući u binarni log sat vremena nakon što se zapravo dogodilo. Kada replika obradi naredbu, ona će privremeno da prijavi da je sat vremena iza mastera, a zatim će skočiti nazad na nula sekundi.
* Ako zaostaje distribucioni master i ima svoje replike, replike će prijaviti da su zaostale nula sekundi, čak i ako postoji kašnjenje u odnosu na krajnjeg mastera.

Rešenje za ove probleme je zanemariti Seconds\_behind\_master i nadgledati zaostajanje replike nečim što možemo direktno posmatrati i meriti. Najbolje rešenje je heartbeat zapis, koji je timestamp koga ažuriramo jednom u sekundi na masteru. Kako bi izračunali kašnjenje, jednostavno možemo da oduzmemo heartbeat od trenutnog timestampa na replici. Ova metoda je imuna na sve probleme koje smo spomenuli i ima dodatnu prednost kreiranja timestampa koji pokazuje u kom trenutku su trenutno podaci replike.

## Ponovna sinhronizacija replike sa masterom

Tradicionalni savet za popravku nesinhronizovane replike je da se ona zaustavi i ponovo klonira iz mastera. Ako je nekonzistentna replika kritičan problem, potrebno je da je zaustavimo i uklonimo iz produkcije čim je pronađemo. Zatim možemo da je ponovo kloniramo ili povratimo iz mastera.

Mana ovog pristupa je kada imamo mnogo podataka. Ako možemo da saznamo koji se podaci razlikuju, verovatno možemo to da učinimo efikasnije nego ponovnim kloniranjem čitavog servera. I ako nekonzistentnost koju smo otkrili nije kritična, možda ćemo moći da ostavimo repliku online i ponovno sinhronizujemo samo pogođene podatke.

Najjednostavnija popravka je odložiti i ponovo učitati samo pogođene podatke pomoću mysqldump-a. Ovo može dobro da funkcioniše ako se podaci ne menjaju dok to radimo. Možemo jednostavno da zaključamo tabelu na masteru, odložimo tabelu, sačekamo da replika sustigne mastera i onda importujemo tabelu na replici. Moramo da sačekamo da replika sustigne mastera kako ne bismo uneli još nekonzistentnosti u druge tabele.

Iako ovo funkcioniše prihvatljivo za mnoge scenarije, često je nemoguće raditi ovo na zauzetom serveru. Takođe ima manu kada se manjaju podaci replike van replikacije. Promena podataka replike kroz replikaciju (pravljenjem promena na masteru) je obično najsigurnija tehnika, jer se izbegavaju uslovi trke i druga iznenađenja. Ako je tabela veoma velika ili je propustnost mreže ograničena, odlaganje i ponovno učitavanje je izuzetno skupo.

## Promena mastera

Često će biti potrebno da usmerimo repliku na novog mastera. Možda je došlo do kvara i mora da promovišemo repliku da bude master ili možda samo realociramo kapacitet. Bez obzira na razlog, moramo da obavestimo repliku o njenom novom masteru.

Kada je proces isplaniran, to je lako. Jednostavno treba da izvršimo komandu CHANGE MASTER TO na replici, korišćenjem odgovarajućih vrednosti, od kojih su većina opcione. Replika će odbaciti svoju trenutnu konfiguraciju i relay logove i počeće repliciranje sa novog mastera. Ona će takođe da ažurira master.info fajl novim parametrima, tako da će promena biti perzistentna tokom ponovnog pokretanja replike. Najteži deo ovog procesa je pronaći željenu poziciju na novom masteru, tako da replika počinje u istoj logičkoj poziciji u kojoj je stopirana na starom masteru.

Promovisanje replike da bude master je malo teže. Postoje dva osnovna scenarija za zamenu mastera jednom od njegovih replika. Prvi je kada je planirana promocija, a drugi kada je neplanirana.

### Planirane promocije

Promovisanje replike da bude master uključuje sledeće korake:

1. Stopirati upise na starom masteru.
2. Opciono dopustiti da njegove replike sustignu replikaciju.
3. Konfigurisati repliku da bude novi master.
4. Usmeriti replike i saobraćaj ka novom masteru, zatim omogućiti upise na njemu.

Međutim, problem su detalji. Moguća su nekoliko scenarija u zavisnosti od replikacione topologije. Na primer, koraci su malo drugačiji u master-master topologiji od koraka u master-replika postavci. Ovo su neki koraci koje ćemo verovatno morati da preduzmemo za većinu postavki:

1. Zaustaviti sve upise na trenutnom masteru. Možda ćemo morati da prisilimo sve klijentske programe da se zatvore.
2. Opciono zaustaviti sve aktivnosti upisa na masteru pomoću FLUSH TABLES WITH READ LOCK. Takođe možemo da postavimo da master bude samo za čitanje pomoću opcije read\_only. Od ovog trenutka, treba zabraniti sve upise na masteru koji će uskoro biti zamenjen, jer jednom kada više nije master, pisanje na njemu znači gubitak podataka. Međutim, postavka read\_only ne sprečava izvršenje postojećih transakcija. Treba ubiti sve otvorene transakcije.
3. Izabrati jednu od replika da bude novi master i osigurati da je ona upotpunosti sustigla replikaciju (odnosno dopustiti da završi izvršenje svih relay logova koje je pribavila od starog mastera).
4. Opciono proveriti da li novi master sadrži iste podatke kao i stari master.
5. Izvršiti STOP SLAVE na novom master.
6. Izvrštiti CHANGE MASTER TO MASTER\_HOST=’’, a zatim RESET SLAVE na novom masteru, kako bismo se diskonektovali od starog mastera i odbacili informacije o konekciji u njegovom master.info fajlu.
7. Treba saznati koordinate binarnog loga novog mastera pomoću SHOW MASTER STATUS.
8. Proveriti da li su sve ostale replike sustigle replikaciju.
9. Isključiti stari master.
10. U MySQL 5.1 i novijim verzijama, aktivirati događaje na novom masteru ako je potrebno.
11. Dopustiti klijente da se povežu sa novim masterom.
12. Izvršiti komandu CHANGE MASTER TO na svakoj replici, ukazujući na novog mastera. Korisititi koordinate binarnog loga koje smo pribavili iz SHOW MASTER STATUS.

### Neplanirane promocije

Ako master padne i moramo da promovišemo repliku da ga zameni, proces možda neće biti lak. Ako postoji samo jedna replika, samo ćemo iskoristiti repliku. Ali ako ih imamo više, moraćemo da uradimo nekoliko dodatnih koraka kako bismo promovisali repliku da bude novi master.

Takođe postoji i dodatni problem zbog potencijalno izbugljenih replikacionih događaja. Moguće je da neka ažuriranja koja su se dogodila na masteru još uvek nisu replicirana ni na jednoj od njegovih replika. Moguće je čak i da je naredba izvršena i onda poništena na masteru, ali nije poništena na replici – tako da bi replika zapravo bila ispred pozicije logičke replikacije mastera. Ako možemo da povratimo podatke mastera u nekom trenutku, možda ćemo moći da povratimo izgbljene naredbe i primenimo ih ručno.

Ovo je procedura za promociju replike u master-replike topologiji:

1. Odrediti koja replika ima najnovije podatke. Proveriti izlaz iz SHOW SLAVE STATUS na svakoj replici i izabrati onu čije su najnovije koordinate Master\_Log\_File/ Read\_Master\_Log\_Pos
2. Dopustiti da sve replike završe izvršenje relay logova koje su pribavile od starog mastera pre nego što je pao. Ako promenimo master replike pre nego što se izvrši relay log, odbaciće se preostali događaji loga i nećemo znati gde se zaustavio.
3. Izvrštiti korake 5-7 iz liste u predhodnom odeljku.
4. Uporediti Master\_Log\_File/ Read\_Master\_Log\_Pos koordinate svake replike sa koordinatama novog mastera.
5. Izvršiti korake 10-12 iz liste u predhodnom odeljku.

### Pronalaženje željenih pozicija loga

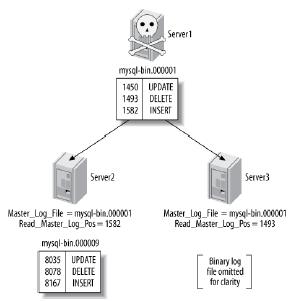
Ako neka replika nije na istoj poziciji kao novi master, moraćemo da pronađemo poziciju u binarnim logovima novog mastera koja odgovara poslednjem događaju koji je replika izvršila, i da je iskoristimo u CHANGE MASTER TO. Možemo koristiti mysqlbinlog alat za pregledavanje zadnjeg upita koji je replika izvršila i pronalaženje ovog istog upita u binarnom logu novog mastera.

Da bismo ovo ilustrovali, pretpostavićemo da događaji loga inkrementiraju ID i da je najnovija replika – novi master – upravo pribavio događaj 100 kada je stari master pao. Pretpostavimo da postoje još dve replike, replica2 i replica3; replica2 je pribavila događaj 99, a replica3 je pribavila događaj 98. Ako obe replike usmerimo na trenutnu poziciju binarnog loga novog mastera, one će početi da repliciraju događaj 101, tako da neće biti sinhronizovane. Međutim, sve dok je omogućen binarni log novog mastera pomoću log\_slave\_updates, možemo da pronađemo događaje 99 i 100 u binarnom logu novog mastera, tako da možemo da vratimo replike u konzistentno stanje.

Zbog restartovanja servera ili različitih konfiguracija isti događaji mogu da postoje u različitim bajt pomerajima na različitim serverima. Pronalaženje događaja može da bude sporo, ali obično nije teško. Samo treba da pregledamo poslednji događaj koji se izvršio na svakoj replici, tako što ćemo pokrenuti mysqlbinlog u binarnom logu ili relay logu replika. Zatim treba da pronađemo isti upit u binarnom logu novog mastera, takođe pomoću mysqlbinlog; ovo će odštampati pomeraj upita u bajtovima i možemo da koristimo taj pomeraj u CHANGE MASTER TO upitu.

Ovaj proces možemo da ubrzamo tako što ćemo oduzeti pomeraje u bajtovima na kojima su se zaustavili novi master i replika, što će nam dati razliku u njihovim pozicijama. Ako onda oduzmemo tu vrednost od trenutne pozicije binarnog loga novog mastera, velike su šanse da će željeni upit biti na toj poziciji. Na taj način smo pronašli poziciju na kojoj treba da startujemo repliku.

Prikazaćemo ovo na konkretnom primeru. Pretpostavimo da je server 1 master serverima 2 i 3, i on se srušio. Prema Master\_Log\_File/Read\_Master\_Log\_Pos u SHOW SLAVE, server 2 je uspeo da replicira sve događaje koji su bili u binarnom logu servera 1, ali server 3 nije ažuriran. Slika 36 ilustrira ovaj scenario.



Slika 36

Prema slici 36, možemo da budemo sigurni da je server 2 replicirao sve događaje u binarnom logu mastera jer se njegovi Master\_Log\_File i Read\_Master\_Log\_pos poklapaju sa poslednjim pozicijama na serveru 1. Stoga, možemo da promovišemo server 2 da bude novi master, a da server 3 bude njegova replika.

Server 3 se zaustavio na pomeraju 1493, što je 89 bajta iza pomeraja 1582, poslednje komande koju je server 2 izvršio. Server 2 trenutno upisuje na poziciju 8167 u svom binarnom logu. 8167 – 89 = 8078, tako da teoretski treba da ukažemo server 3 na taj pomeraj u logovima servera 2. Međutim, dobra je ideja da istražimo događaje loga oko ove pozicije i proverimo da server 2 stvarno ima ispravne događaje na tom pomeraju u svojim logovima. Možda će tamo biti nešto drugo, na primer zbog ažuriranja podataka koja su se desila samo na serveru 2. Pod pretpostavkom da su događaji isti nakon pregleda, komanda na slici 37 će server 3 prebaciti da bude replika servera 2.



Slika 37

Šta ako je server 1 završio izvršenje i logovanje još jednog događaja, izvan pomeraja 1582, kada se srušio? Pošto je server 2 pročitao i izvršio samo do pomeraja 1582, možda smo zauvek izgubili jedan događaj. Međutim, ako disk starog mastera nije oštećen, idalje možemo da povratimo događaj koji nedostaje iz njegovog binarnog loga pomoću mysqlbinlog.

Ako je potrebno da povratimo događaje koji nedostaju iz starog mastera, to treba da uradimo nakon što postavimo novi master, ali pre nego što dopustimo klijentske konekcije. Na ovaj način nećemo morati da izvršavamo događaje koji nedostaju na svakoj replici; replikacija će se pobrinuti o tome za nas. Ako je master potpuno nedostupan, možda ćemo morati da sačekamo da to uradimo kasnije.

Varijacija ovog postupka je korišćenje pouzdanog načina za skladištenje fajlova binarnog loga mastera, kao što su SAN ili DRBD. Čak i ako se master upotpunosti srušio, i dalje ćemo imati njegove binarne logove. Možemo da postavimo server loga, da replike usmerimo ka njemu i zatim ih pustimo da dođu do tačke u kojoj se master srušio. Ovo olakšava promociju jedne od replika da postane novi master – ovo je suštinski isti postupak kao što smo pokazali za planiranu promociju.

## Promena uloga u master-master konfiguraciji

Jedna od prednosti master-master replikacije je da možemo lako da promenimo aktivne i pasivne uloge, zbog simetrične konfiguracije. U ovom odeljku ćemo pokazati kako da ih zamenimo.

Kada menjamo uloge u master-master konfiguraciji, najbitnija stvar je da osiguramo da samo jedan od ko-mastera upisuje u jednom trenutku. Ako se upisi jednog od mastera prekinu upisima od drugog, upisi mogu da se sukobe. Drugim rečima, pasivan server ne sme da prima bilo koje događaje binarnog loga od aktivnog servera nakon što se uloge zamene. Ovo možemo da garantujemo da se neće desiti tako što ćemo osigurati da replikaciona SQL nit pasivnog servera sustigne aktivnog servera pre nego što mu omogućimo upis.

Sledeći koraci menjaju uloge bez opasnosti od sukobljenih ažuriranja:

1. Zaustaviti sve upise na aktivnom serveru.
2. Izvršiti SET GLOBAL read\_only = 1 na aktivnom serveru i postaviti read\_only opciju u njegovom konfiguracionom fajlu zbog bezbednosti u slučaju restartovanja. Ovo neće zaustaviti korisnike sa SUPER privilegijom da prave promene. Ako želimo da spečimo promene od svih korisnika, koristićemo FLUSH TABLES WITH READ LOCK. Ako ovo ne uradimo, moramo da ubijemo sve klijentski konekcije da bismo bili sigurni da ne postoje naredbe koje se dugo izvršavaju ili neizvršene transakcije.
3. Izvršiti SHOW MASTER STATUS na aktivnom serveru i zapamtiti koordinate binarnog loga.
4. Izvršiti SELECT MASTER\_POS\_WAIT() na pasivnom serveru sa koordinatama binarnog loga aktivnog servera.
5. Izvršiti SET GLOBAL read\_only = 0 na pasivnom serveru, ovo će ga učiniti aktivnim serverom.
6. Ponovno konfigurisati aplikacije da upisuju na novom aktivnom serveru.

U zavisnosti od konfiguracije naše aplikacije, možda ćemo trebati da uradimo još i druge zadatke, uključujući promenu IP adrese na dva servera.

# Zaključak

Osnovni problem koji replikacija rešava je čuvanje podataka jednog servera sinhronizovanih sa podacima drugog. Cilj ovog rada je bio da predstavimo osnove replikacije kod MySQL-a, vidimo kako replikacija zapravo funkcioniše i objasnimo kako MySQL replicira podatke. Pored toga, prikazali smo detalje podešavanja replikacije. Podešavanje replikacije je prilično jednostavan proces u MySQL-u, ali postoji mnogo varijacija u osnovnim koracima, u zavisnosti od scenarija. Možemo da postavimo MySQL replikaciju za skoro bilo koju konfiguraciju mastera i replika, uz ograničenje da data instanca MySQL replike može da ima samo jednog mastera. Mnoge složene replikacione topologije su moguće, ali čak i jednostavne mogu da budu veoma fleksibilne. U radu smo objasnili nekoliko replikacionih topologija koje mogu da imaju više različitih upotreba. U poslednjem delu rada objasnili smo i monitoring i administriranje replikacione topologije, jer će to biti redovan posao, bez obzira na to koliko imamo servera.

# Literatura

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

<https://dev.mysql.com/doc/refman/8.0/en/replication.html>