UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

Replikacija podataka kod MySQL baze podataka

Seminarski rad

Studijski program: Računarstvo i informatika

Modul: Softversko inženjerstvo

Mentor:

Doc. dr Aleksandar Stanimirović

Student:

Vuk Cvetković 1667

Niš, Jun 2024.

Sadržaj

Uvod 2
Replikacija kod MySQL 4
Konfiguracija replikacije 5
Replikacija zasnovana na binarnom log-u 5
Konfiguracija izvorišnog servera 6
Postavljanje odredišnog servera (replike) 6
Kreiranje korisnika za replikaciju 8
Dobijanje informacija o binarnom log-u sa izvora 8
Snimanje podataka 9
Konfiguracija odredišnog servera 10
Replikacija sa globalnim transakcionim identifikatorima 12
GTID format i skladištenje 13
GTID životni ciklus 16
GTID automatsko pozicioniranje 16
Podešavanje replikacije preko GTID 16
Korišćenje GTID-ova za Failover i Scaleout 18
Ograničenja GTID-bazirane replikacije 19
Multi-Source Replikacija 21
Konfiguracija Multi-Source replikacije 21
Obezbeđivanje replike sa više izvora za replikaciju zasnovanu na GTID 22
Dodavanje GTID-baziranih izvora u Multi-Source repliku 23
Dodavanje izvora baziranih na binarnim logovima u Multi-Source repliku 23
Upravljanje Multi-Source replikacijom 24
Implementacija Replikacije 26
Formati replikacije 26
Prednosti i mane replikacije zasnovane na izjavama i redovima 28
Korišćenje replikacije zasnovane na redovima (RBR) 29
Određivanje sigurnih i nesigurnih izjava u binarnom logovanju 30
Praktične primene replikacije 32
Zaključak 33
Literatura 34

Uvod

U savremenom svetu, gde su podaci ključni resurs, obezbeđivanje dostupnosti i pouzdanosti baza podataka postaje od suštinskog značaja. Replikacija podataka je jedna od tehnika koja se koristi za postizanje ovih ciljeva. Ona omogućava kopiranje i održavanje podataka između više baza podataka, čime se povećava otpornost sistema na greške i omogućava bolje balansiranje opterećenja.

Replikacija podataka u MySQL bazi podataka predstavlja proces prenosa i sinhronizacije podataka sa jednog servera na drugi. Ovo se postiže korišćenjem specijalizovanih mehanizama i algoritama koji obezbeđuju da sve promene na glavnom serveru (source) budu replicirane na jedan ili više odredišnih servera (replike). Na taj način, svaki odredišni server poseduje kopiju baze podataka glavnog servera, što omogućava visok nivo dostupnosti podataka, bolju distribuciju opterećenja i poboljšane performanse aplikacija koje koriste bazu podataka.

U ovom seminarskom radu, biće obrađena replikacija podataka u MySQL bazi podataka, kombinujući teorijske osnove sa praktičnim primerima implementacije. Biće prikazani ključni koncepti, vrste replikacije, proces replikacije, kao i napredne funkcionalnosti koje MySQL nudi za optimizaciju i upravljanje replikacijom.

Teorijska osnova replikacije

Replikacija podataka je proces kopiranja i održavanja baze podataka na više servera. Ovaj proces obezbeđuje nekoliko ključnih prednosti:

Dostupnost	Jedna od primarnih prednosti replikacije je povećanje dostupnosti podataka. U slučaju kvara glavnog servera, podaci su i dalje dostupni na drugim replikacionim serverima. To znači da korisnici mogu nastaviti da pristupaju podacima i aplikacijama bez prekida, što je ključno za poslovne procese koji zahtevaju visoku dostupnost. Ovaj mehanizam smanjuje rizik od gubitka podataka i omogućava kontinuitet poslovanja.
Performanse	Replikacija omogućava raspodelu opterećenja na više servera, što može značajno poboljšati performanse sistema. Umesto da jedan server odgovara na sve upite, više replikacionih servera može deliti ovo opterećenje. To rezultuje bržim odgovorima na korisničke zahteve i efikasnijim korišćenjem resursa. Na primer, u sistemima sa velikim brojem čitanja, slave serveri mogu biti korišćeni za čitanje podataka, dok master server rukuje upisima.
Oporavak od katastrofa	Replikacija podataka je ključna za strategije oporavka od katastrofa. U slučaju gubitka podataka na glavnom serveru zbog kvara hardvera, softverske greške ili bilo kog drugog razloga, podaci se mogu brzo oporaviti sa replikacionih servera. Ovaj proces smanjuje vreme oporavka i minimizira gubitak podataka, što je posebno važno za organizacije koje ne mogu priuštiti duge periode neaktivnosti ili gubitak kritičnih informacija.

Skalabilnost	Replikacija omogućava sistemima da skaliraju horizontalno dodavanjem novih replikacionih servera prema potrebi. Ova sposobnost skalabilnosti je korisna za aplikacije koje rastu i zahtevaju povećanu obradu podataka i kapacitet skladištenja. Dodavanjem novih servera u replikacioni sistem, organizacije mogu efikasno upravljati povećanim opterećenjem bez značajnih promena u postojećoj infrastrukturi.
Redundantnost	Redundantnost je još jedna ključna prednost replikacije. Čuvanje više kopija baze podataka na različitim serverima stvara sigurnosnu mrežu koja štiti od gubitka podataka. U slučaju kvara jednog servera, druge replike obezbeđuju kontinuitet i integritet podataka. Ovaj nivo redundantnosti je posebno važan za sisteme koji zahtevaju visok stepen pouzdanosti i zaštite podataka.
Geografska distribucija	Replikacija omogućava geografski raspored servera, što može smanjiti latenciju i poboljšati brzinu pristupa podacima za korisnike širom sveta. Na primer, replike baza podataka mogu biti postavljene u različitim data centrima širom sveta, čime se obezbeđuje brži pristup podacima za korisnike iz različitih geografskih regiona.

U MySQL-u, replikacija se najčešće koristi za povećanje skalabilnosti i dostupnosti aplikacija. Osnovni model replikacije podataka u MySQL-u je model glavnog i podređenog servera. U ovom modelu, glavni server vodi evidenciju svih promena u bazi podataka, koje zatim šalje podređenim serverima. Podređeni serveri primaju ove promene i primenjuju ih na svoje lokalne baze podataka, čime se obezbeđuje sinhronizacija sa glavnim serverom.

Praktična primena replikacije

Da bismo bolje razumeli kako replikacija funkcioniše u praksi, proći ćemo kroz korake postavljanja replikacije u MySQL-u. Ovi koraci uključuju podešavanje glavnog servera, kreiranje korisnika za replikaciju, podešavanje podređenih servera i praćenje statusa replikacije.

Na kraju, razmotrićemo napredne funkcionalnosti kao što su Global Transaction Identifiers (GTID) i multi-source replikacija, koje omogućavaju veću fleksibilnost i otpornost sistema.

Replikacija kod MySQL

Replikacija [1] omogućava podacima sa jednog MySQL servera (izvor) da budu kopirani na jedan ili više MySQL servera (replike). Replikacija je asinhrona po default-u; replike ne moraju stalno biti povezane da bi primale ažuriranja sa izvora. Prema konfiguraciji, moguće je replikovati sve baze podataka, odabrane baze podataka ili čak odabrane tabele unutar baze.

Prednosti replikacije u MySQL-u uključuju:

Raspodelu opterećenja	Rasprostiranje opterećenja među više replika radi poboljšanja performansi. U ovom okruženju, svi upisi i ažuriranja moraju se izvršiti na izvornom serveru. Čitanja, međutim, mogu se obavljati na jednoj ili više replika. Ovaj model može poboljšati performanse upisa (jer je izvor posvećen ažuriranjima), dok dramatično povećava brzinu čitanja preko rastućeg broja replika.
Bezbednost podataka	Zbog toga što replika može zaustaviti proces replikacije, moguće je pokrenuti usluge za rezervne kopije na replici bez oštećenja odgovarajućih podataka na izvoru.
Analitiku	Živi podaci mogu biti kreirani na izvoru, dok se analiza informacija može vršiti na replici bez uticaja na performanse izvora.
Distribuciju podataka na daljinu	Može se koristiti replikacija za kreiranje lokalne kopije podataka za udaljeno mesto, bez stalnog pristupa izvoru.

MySQL podržava različite metode replikacije. Tradicionalna metoda se zasniva na repliciranju događaja iz binarnog zapisa izvora, i zahteva sinhronizaciju log fajlova i njihovih pozicija između izvora i replike. Novija metoda zasnovana na globalnim identifikatorima transakcija (GTID) je transakciona i stoga ne zahteva rad sa log fajlovima ili pozicijama unutar njih, što značajno pojednostavljuje mnoge uobičajene zadatke replikacije. Replikacija pomoću GTID-a garantuje doslednost između izvora i replike sve dok su sve transakcije koje su potvrđene na izvoru takođe primenjene na replici.

Replikacija u MySQL-u podržava različite vrste sinhronizacije. Osnovni tip sinhronizacije je jednosmerna, asinhrona replikacija, u kojoj jedan server deluje kao izvor, dok jedan ili više drugih servera deluju kao replike. Ovo je u suprotnosti sa sinhronom replikacijom koja je karakteristična za NDB Cluster. U MySQL 8.4, podržana je i polusinhrona replikacija pored ugrađene asinhrone replikacije. Sa polusinhronom replikacijom, potvrda izvršenja na izvoru blokira se pre nego što se vrati sesiji koja je izvršila transakciju, sve dok barem jedna replika ne potvrdi da je primila i zabeležila događaje za transakciju.

Konfiguracija replikacije

Postoji nekoliko načina za konfiguraciju replikacije u MySQL-u, svaki prilagođen različitim potrebama i scenarijima. Ovaj rad se fokusira na tri glavne metode: replikacija zasnovana na binarnom logu, replikacija sa globalnim identifikatorima transakcija (GTID) i multi-source replikacija. Svaka od ovih metoda pruža specifične prednosti i prilagođena je za određene zahteve u performansama, skaliranju i pouzdanosti sistema.

- **Replikacija zasnovana na binarnom logu** omogućava da MySQL serveri razmenjuju podatke beležeći promene u binarni zapis. Ova metoda je fleksibilna i omogućava detaljno konfigurisanje počevši od izbora događaja za replikaciju do kontrole nad binarnim logovima.
- **GTID replikacija** nudi transakcioni pristup, eliminišući potrebu za sinhronizacijom log fajlova i pozicija unutar njih. Ovo pojednostavljuje upravljanje replikacijom i osigurava doslednost između izvora i replika.
- **Multi-source replikacija** omogućava replikaciju podataka sa više izvora na jednu repliku, što je korisno u složenim okruženjima sa više podataka i potrebom za distribuiranjem opterećenja.

Replikacija zasnovana na binarnom log-u

Ovaj deo opisuje replikaciju između MySQL servera zasnovanu na metodi binarnog loga [2], gde MySQL instanca koja funkcioniše kao izvor (gde se vrše promene u bazi podataka) zapisuje ažuriranja i promene kao "događaje" u binarni zapis. Informacije u binarnom zapisu se čuvaju u različitim formatima logovanja u skladu sa promenama baza podataka koje se beleže. Replike su konfigurisane da čitaju binarni zapis sa izvora i izvršavaju događaje iz binarnog zapisa na lokalnoj bazi podataka replike.

Svaka replika prima kopiju celokupnog sadržaja binarnog zapisa. Odgovornost replike je da odluči koje izjave iz binarnog zapisa treba da izvrši. Ako nije drugačije specificirano, sve događaje u binarnom zapisu izvora izvršava replika. Po potrebi, može se konfigurisati replika da obrađuje samo događaje koji se odnose na određene baze podataka ili tabele.

Svaka replika čuva zapis o koordinatama binarnog zapisa: ime datoteke i pozicija unutar datoteke koje je pročitala i obradila sa izvora. To znači da se više replika može povezati sa izvorom i izvršavati različite delove istog binarnog zapisa. Zbog toga što replike kontrolišu ovaj proces, pojedinačne replike se mogu povezivati i isključivati sa servera bez uticaja na rad izvora. Takođe, pošto svaka replika beleži trenutnu poziciju unutar binarnog zapisa, moguće je da se replike odvežu, ponovo povežu i nastave sa obradom.

Izvor i svaka replika moraju biti konfigurisani sa jedinstvenim ID-om (koristeći sistemsku varijablu server_id). Pored toga, svaka replika mora biti konfigurisana sa informacijama o imenu hosta izvora, imenu log fajla i poziciji unutar tog fajla. Ove detalje se kontrolišu unutar MySQL sesije koristeći izjavu CHANGE REPLICATION SOURCE TO na replici.

Konfiguracija izvorišnog servera

Za konfiguraciju MySQL servera [3] kao izvora replikacije koristeći binarni zapis zasnovan na poziciji binarnog loga, potrebno je omogućiti binarno logovanje i postaviti jedinstveni identifikacioni broj servera.

Postavljanje Server ID-a

Svaki server u replikacionom okruženju mora imati jedinstveni identifikacioni broj servera, koji se određuje pomoću sistema promenljive server_id. Ovaj broj razlikuje servere unutar replikacione topologije i treba da bude pozitivan ceo broj između 1 i (2^32)−1. Podrazumevano, server_id je postavljen na 1.

Omogućavanje binarnog logovanja

Binarno logovanje je obavezno na izvoru jer binarni zapis služi kao osnova za repliciranje promena sa izvora na njegove replike. Binarno logovanje je podrazumevano omogućeno (promenljiva *log_bin* je podešena na ON). Opcija --log-bin serveru govori o osnovnom imenu datoteka binarnog loga.

Obe ove promenljive (server-id i log-bin) se menjaju u konfiguracionom fajlu mysql-a.

```
log-bin="baze3-bin"
server-id=1
```

Slika 1. Postavljanje promenljivih log-bin i server-id

Da bismo videli da li je omoguceno binarno logovanje, možemo proveriti promenljivu log-bin:

```
mysql> SHOW VARIABLES LIKE '%log_bin%';
+---------------------------------+-----------------------------------------------------------+
| Variable_name                   | Value                                                     |
+---------------------------------+-----------------------------------------------------------+
| log_bin                         | ON                                                        |
| log_bin_basename                | C:\ProgramData\MySQL\MySQL Server 8.4\Data\baze3-bin       |
| log_bin_index                   | C:\ProgramData\MySQL\MySQL Server 8.4\Data\baze3-bin.index |
| log_bin_trust_function_creators | OFF                                                       |
| sql_log_bin                     | ON                                                        |
+---------------------------------+-----------------------------------------------------------+
```

Slika 2. Promenljiva log-bin

Postavljanje odredišnog servera (replike)

Svaka replika u MySQL replikacionoj topologiji mora imati jedinstveni server ID, koji se podešava pomoću promenljive server_id. Ako se konfigurišu više replika, svaka mora imati svoju jedinstvenu vrednost server_id koja se razlikuje od izvora i ostalih replika. [4]

Podrazumevana vrednost server_id je 1. Može se dinamički promeniti server_id izdavanjem naredbe:

SET GLOBAL server_id = 2;

```
mysql> SET GLOBAL server_id = 2;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW VARIABLES LIKE '%server_id%';
+----------------+-------+
| Variable_name  | Value |
+----------------+-------+
| server_id      | 2     |
| server_id_bits | 32    |
+----------------+-------+
2 rows in set (0.00 sec)
```

Slika 3. Promena server_id-a

Važno je napomenuti da vrednost 0 za server ID sprečava repliku da se poveže sa izvorom. Ako je ta vrednost server_id-a (koja je bila podrazumevana u ranijim verzijama) već bila postavljena, mora se restartovati server da bi se inicijalizovala replika sa novim nenultnim server_id. Inače, restart servera nije potreban kada se menja server_id, osim ako se ne naprave druge konfiguracijske promene koje to zahtevaju. Na primer, ako je binarno logovanje bilo onemogućeno na serveru i ako je potrebno omogućiti ga za repliku, potreban je restart servera da bi se to omogućilo.

Binarno logovanje je podrazumevano omogućeno na svim serverima. Replici nije potrebno da ima omogućeno binarno logovanje da bi se vršila replikacija. Međutim, binarno logovanje na replici znači da se binarni zapis replike može koristiti za backup podataka i oporavak nakon pada sistema. Replike koje imaju omogućeno binarno logovanje mogu se takođe koristiti kao deo složenije replikacione topologije. Na primer, moguće je postaviti replikacione servere u ovakav povezani raspored:

A -> B -> C

Ovde A služi kao izvor za repliku B, a B kao izvor za repliku C. Da bi ovo funkcionisalo, B mora biti i izvor i replika. Ažuriranja primljena od A moraju biti zabeležena od strane B u svoj binarni zapis, kako bi se prosledila C. Osim binarnog logovanja, ova replikaciona topologija zahteva da se sistemski promenljiva log_replica_updates omogući. Sa omogućenim ažuriranjima replike, replika zapisuje ažuriranja koja su primljena od izvora i izvršena od strane SQL niti replike u svoj binarni zapis. Podrazumevano je log_replica_updates omogućen.

```
mysql> SHOW VARIABLES LIKE '%log_replica_updates%';
+---------------------+-------+
| Variable_name       | Value |
+---------------------+-------+
| log_replica_updates | ON    |
+---------------------+-------+
1 row in set (0.01 sec)
```

Slika 4. Default vrednost promenljive log_replica_updates

Ako je potrebno da se onemogući binarno logovanje ili zapisivanje ažuriranja replike na replici, to se može podesiti korišćenjem opcija --skip-log-bin i --log-replica-updates=OFF. Ako je kasnije potrebno ponovo da se omoguće ove funkcije na replici, uklanjaju se odgovarajuće opcije i restartuje se server.

Kreiranje korisnika za replikaciju

Svaka replika se povezuje sa izvorom koristeći MySQL korisničko ime i šifru, pa mora postojati korisnički nalog na izvoru koji će replika koristiti za povezivanje [5]. Ime korisnika se specificira opcijom SOURCE_USER u CHANGE REPLICATION SOURCE TO naredbi prilikom postavljanja replike. Za ovu operaciju može se koristiti bilo koji nalog, pod uslovom da je dodeljena privilegija REPLICATION SLAVE. Mogu se kreirati različiti nalozi za svaku repliku, a moguće je i da se povezuju sa izvorom koristeći isti nalog za sve replike.

Za kreiranje novog naloga koristi se CREATE USER. Da bi se dodelila potrebna privilegija ovom nalogu za replikaciju, koristi se GRANT naredba. Ako se kreira nalog isključivo za potrebe replikacije, taj nalog treba imati samo REPLICATION SLAVE privilegiju. Na primer, za postavljanje novog korisnika repl, koji može da se poveže za replikaciju sa bilo kog hosta unutar *localhost* domena, izvršava se ova naredbe na izvoru:

```
mysql> CREATE USER 'repl'@'localhost' IDENTIFIED BY 'repl';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

Slika 5. Kreiranje naloga i dodela privilegija

Ovde se prvo kreira user *repl*, kome se doda privilegija REPLICATION SLAVE i onda se FLUSH-uju privilegije.

Dobijanje informacija o binarnom log-u sa izvora

Ovaj postupak opisuje kako se dobijaju koordinate binarnog zapisa sa MySQL izvornog servera [6], što je ključno za postavljanje replikacije s replikom. Koordinate binarnog zapisa određuju tačno mesto unutar datoteke binarnog zapisa odakle će replika početi primenjivati promene s izvora.

Flush Tables with Read Lock:

Na izvornom serveru se započinje nova sesija i izvršava se sledeća naredba kako bi se blokirale sve tabele i operacije pisanja:

```
mysql> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.00 sec)
```

Slika 6. Naredba FLUSH TABLES WITH READ LOCK

Dobijanje imena datoteke i pozicije binarnog zapisa:

Na source serveru se koristi naredba SHOW BINARY LOG STATUS kako bi se dobili trenutno ime datoteke binarnog zapisa i pozicija:

```
mysql> SHOW BINARY LOG STATUS\G
*************************** 1. row ***
             File: baze3-bin.000004
         Position: 158
     Binlog_Do_DB:
 Binlog_Ignore_DB:
Executed_Gtid_Set:
1 row in set (0.00 sec)
```

Slika 7. Informacije o bin log fajlu

- File: Prikazuje ime trenutne datoteke binarnog zapisa (baze3-bin.000004 u primeru).
- Position: Pokazuje poziciju unutar datoteke binarnog zapisa (158 u primeru).

Ove vrednosti su bitne jer označavaju početnu tačku u binarnom zapisu odakle će replika započeti replikaciju.

Snimanje podataka

Za kreiranje snimka podataka [7] iz postojeće izvorne baze podataka može se koristiti alat **mysqldump**. Nakon završetka snimanja podataka, potrebno je da se podaci uvezu na repliku pre pokretanja procesa replikacije.

U sledećem primeru se sve baze podataka snimaju u datoteku nazvanu dbdump.db, a --source-data opcija automatski dodaje CHANGE REPLICATION SOURCE TO naredbu koja je potrebna na repliku za pokretanje procesa replikacije:

```
C:\Windows\System32>mysqldump -u root -p --all-databases --source-data > dbdump.db
Enter password: ****
```

Slika 8. Izvoz svih baza u fajl dbdump.db

Moguće je i ručno odabrati za koje baze podataka želimo da napravimo snimak, kao i za koje tabele u okviru baze. Taj primer je dat na sledećoj slici:

```
C:\Windows\System32>mysqldump -u root -p --databases baze3 --tables employee --source-data > employee.sql
Enter password: ****
```

Slika 9. Izvoz tabele employee iz baze baze3

Ovde je napravljen snimak baze *baze3* i tabele *employee* u okviru te baze. Fajl koji se kreira izgleda ovako:

```
--
-- Position to start replication or point-in-time recovery from
--

-- CHANGE REPLICATION SOURCE TO SOURCE_LOG_FILE='baze3-bin.000004', SOURCE_LOG_POS=158;

--
-- Table structure for table `employee`
--

DROP TABLE IF EXISTS `employee`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `employee` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(100) NOT NULL,
  `department` varchar(100) NOT NULL,
  `salary` decimal(10,2) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `employee`
--

LOCK TABLES `employee` WRITE;
/*!40000 ALTER TABLE `employee` DISABLE KEYS */;
INSERT INTO `employee` VALUES (1,'John Doe','IT',5000.00),(2,'Jane Smith','HR',4500.00),
/*!40000 ALTER TABLE `employee` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
```

Slika 10. Fajl employee.sql nakon izvoza

Konfiguracija odredišnog servera

Da bi se pravilno konfigurisala [8] MySQL replikacija koristeći izvorišni (source) i odredišni (destination) server, potrebno je imati dva servera: jedan koji će služiti kao izvor promena i drugi kao odredište tih promena. U ovom konkretnom primeru, oba servera će biti pokrenuta na istom računaru, ali će imati različite konfiguracije i portove.

Postavljanje MySQL odredišnog servera (Destination)

- **Kreiranje direktorijuma za odredišni server** - Prvo je potrebno kreirati direktorijum gde će se nalaziti sve datoteke potrebne za pokretanje MySQL servera za odredište. Na primer, možemo koristiti direktorijum C:\mysql_slave_data.
- **Inicijalizacija direktorijuma za odredišni server** - Nakon kreiranja direktorijuma, inicijalizujemo MySQL server za odredište. Ovo uključuje konfigurisanje servera, definisanje direktorijuma podataka i drugih važnih parametara.
- **Pokretanje MySQL servera na odredištu** - Pokrećemo MySQL server na odredišnom serveru, koristeći odgovarajuće komande za pokretanje sa unapred definisanim parametrima i specifikacijama.

Svi ovi koraci su dati u nastavku:

```
mkdir C:\mysql_slave_data

mysqld --initialize-insecure --datadir=C:\mysql_slave_data

mysqld --port=3307 --datadir=C:\mysql_slave_data --server-id=2 --log-bin=mysql-bin
```

Slika 11. Konfiguracija odredišnog servera

Ovim je pokrenut MySQL server na portu 3307, dodeljen mu je folder gde će da smešta sve informacije, promenjen je server_id i definisan je log fajl.

Sledeći korak je uvoz dump fajla na replika server. Kada je odredišni MySQL server pokrenut i spreman za rad, možemo uvesti dump fajl sa izvornog servera. Ovo se postiže korišćenjem mysql komande za uvoz, na primer:

```
mysql -u root -p --port=3307 < employee.sql
```

Slika 12. Uvoz employee.sql fajla na odredišni server

Kada se uveze fajl, potrebno je otvoriti destination server putem komande:

```
mysql -u root -p --port=3307
```

Slika 13. Otvaranje destination servera

Nakon što je uvoz dump fajla završen, konfigurišemo odredišni server za replikaciju koristeći informacije sa izvornog servera. Ovo uključuje podešavanje binarnog loga i pozicije na kojoj će odredišni server čekati nove promene:

```
mysql> CHANGE REPLICATION SOURCE TO
    -> SOURCE_HOST='127.0.0.1',
    -> SOURCE_USER='repl',
    -> SOURCE_PASSWORD='repl',
    -> SOURCE_LOG_FILE='baze3-bin.000004',
    -> SOURCE_LOG_POS=158;
```

Slika 14. Konfiguracija odredišnog servera

Nakon konfiguracije, pokrećemo proces replikacije na odredišnom serveru:

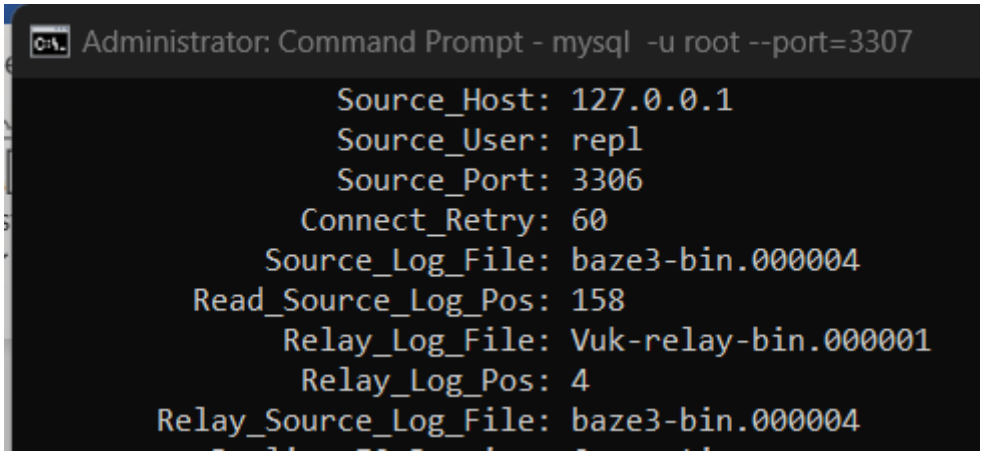
```
mysql> START REPLICA;
Query OK, 0 rows affected (0.01 sec)
```

Slika 15. Pokretanje replikacije

Konačno, proveravamo da li je replikacija uspešno pokrenuta na odredišnom serveru:

```
mysql> SHOW REPLICA STATUS\G;
```

Slika 16. Naredba za prikaz informacija o replikaciji



Slika 17. Informacije o source serveru

Replikacija sa globalnim transakcionim identifikatorima

Ovaj deo objašnjava replikaciju baziranu na transakcijama koristeći globalne identifikatore transakcija (GTID) [9]. GTID omogućava jedinstveno identifikovanje i praćenje svake transakcije kako se komituje na izvornom serveru i primenjuje na replikama. Ovo eliminiše potrebu za referenciranjem specifičnih log fajlova ili pozicija prilikom postavljanja nove replike ili prelaska na novi izvor, što značajno pojednostavljuje ove zadatke. Replikacija bazirana na GTID-ovima osigurava konzistentnost između izvora i replika praćenjem transakcija na nivou transakcije. Dokle god su sve transakcije koje su komitovane na izvoru takođe komitovane na replici, konzistentnost je zagarantovana.

Ovaj deo uključuje sledeće teme:

- Kako su definisani i kreirani GTID-ovi, kao i njihovo skladištenje u MySQL serveru.
- Životni ciklus GTID-a.
- Automatsko pozicioniranje za sinhronizaciju između replike i izvora koji koriste GTID-ove.
- Opšti postupak za podešavanje i pokretanje replikacije bazirane na GTID-ovima.
- Preporučeni metodi za postavljanje novih servera za replikaciju kada se koriste GTID-ovi.
- Ograničenja i limitacije koje treba imati na umu prilikom korišćenja replikacije bazirane na GTID-ovima.

Ovo osigurava sveobuhvatan pregled GTID-based replikacije u MySQL-u, pružajući stabilan mehanizam za upravljanje podacima između servera bez potrebe za složenim konfiguracijama i praćenjem specifičnih log fajlova.

GTID format i skladištenje

Globalni identifikator transakcije (GTID) je jedinstveni identifikator koji se dodeljuje svakoj transakciji koja se komituje na izvornom serveru u MySQL-u. Ovaj identifikator je jedinstven ne samo na serveru na kojem je transakcija nastala, već i u celoj replikacionoj topologiji.

Klijentske transakcije dobijaju novi GTID prilikom komitovanja na izvoru i upisivanja u binarni zapis, garantujući monoton rast bez praznina među generisanim brojevima.

Auto-skip funkcija za GTID-ove omogućava da se svaka transakcija sa određenim GTID-om primeni samo jednom na replici, čime se garantuje doslednost. Ako transakcija sa određenim GTID-om već počne da se izvršava na serveru, bilo koji pokušaj pokretanja konkurentne transakcije sa istim GTID-om blokira se sve dok se prva transakcija ne komituje ili vrati unazad.

GTID u MySQL-u je jedinstveni identifikator koji se sastoji od dva ili tri dela odvojenih dvotačkom, u zavisnosti od toga da li je običan ili označen:

Običan GTID: Sastoji se od source_id i transaction_id:

- source_id: Identifikuje server sa kojeg potiče transakcija, obično korišćenjem server_uuid-a.
- transaction_id: Redni broj koji određuje redosled komitovanja transakcija na izvornom serveru.

Na primer, transakcija koja je 23. po redu komitovana na serveru sa UUID-om

3E11FA47-71CA-11E1-9E33-C80AA9429562

ima GTID

3E11FA47-71CA-11E1-9E33-C80AA9429562:23.

Označen GTID: Sastoji se od source_id, tag i transaction_id:

- source_id: Identifikuje server sa kojeg potiče transakcija, obično korišćenjem server_uuid-a.
- tag: Korisnički definisan string koji identifikuje specifičnu grupu transakcija.
- transaction_id: Redni broj koji određuje redosled komitovanja transakcija na izvornom serveru.

Na primer, transakcija koja je 117. po redu komitovana na serveru sa UUID-om

ed102faf-eb00-11eb-8f20-0c5415bfaa1d

i sa tagom

Domain_1

ima GTID

ed102faf-eb00-11eb-8f20-0c5415bfaa1d:Domain_1:117.

GTID Set

GTID set u MySQL-u predstavlja skup koji se sastoji od jednog ili više pojedinačnih GTID-ova ili opsega GTID-ova. GTID setovi se koriste na razne načine u MySQL serveru:

- **Skladištenje GTID setova**: Vrednosti koje čuva sistemskim promenljivama gtid_executed i gtid_purged su GTID setovi.
- **Korišćenje u REPLICA komandi**: Opcije START REPLICA kao što su UNTIL SQL_BEFORE_GTIDS i UNTIL SQL_AFTER_GTIDS omogućavaju procesu replikacije da obradi transakcije samo do prvog GTID-a u skupu ili da se zaustavi nakon poslednjeg GTID-a u skupu.
- **Funkcije GTID_SUBSET() i GTID_SUBTRACT()**: Ove ugrađene funkcije zahtevaju GTID setove kao ulaz.

GTID setovi mogu da obuhvataju jedan ili više pojedinačnih GTID-ova ili opsega GTID-ova. Na primer:

Opseg GTID-ova sa istog servera može se sažeti u jedan izraz:

3E11FA47-71CA-11E1-9E33-C80AA9429562:1-5

Više pojedinačnih GTID-ova ili opsega sa istog servera može biti uključeno u jedan izraz, razdvojeni dvotačkom:

3E11FA47-71CA-11E1-9E33-C80AA9429562:1-3:11:47-49

Tabela *mysql.gtid_executed* se kreira (ako već nije kreirana) prilikom instalacije ili nadogradnje MySQL Server-a, koristeći CREATE TABLE izjavu sličnu sledećoj:

```
CREATE TABLE gtid_executed (
  source_uuid CHAR(36) NOT NULL,
  interval_start BIGINT NOT NULL,
  interval_end BIGINT NOT NULL,
  gtid_tag CHAR(32) NOT NULL,
  PRIMARY KEY (source_uuid, gtid_tag, interval_start)
);
```

Slika 18. Tabela gtid_executed

Tabela mysql.gtid_executed je namenjena internoj upotrebi od strane MySQL servera. Omogućava replici da koristi GTID-ove kada je binarno zapisivanje isključeno na replici, i omogućava čuvanje stanja GTID-a kada su binarni zapisi izgubljeni.

GTID-ovi se čuvaju u tabeli mysql.gtid_executed samo kada je gtid_mode podešen na ON ili ON_PERMISSIVE. Ako je binarno zapisivanje isključeno (log_bin je OFF) ili log_replica_updates isključen, server čuva GTID-ove za svaku transakciju u baferu prilikom

commitovanja. Periodično, sadržaj bafera se dodaje u tabelu mysql.gtid_executed kao jedan ili više unosa. Tabela se takođe periodično kompresuje prema korisnički podešenom intervalu.

Kada je binarno zapisivanje uključeno (log_bin je ON), samo za InnoDB Storage Engine, server ažurira tabelu mysql.gtid_executed prilikom commitovanja svake transakcije. Za druge Storage Engine, ažuriranje se dešava kada se binarni zapis rotira ili server isključi, upisujući GTID-ove za transakcije iz prethodnog binarnog zapisa.

Ako mysql.gtid_executed tabela nije dostupna za pisanje i binarni zapis se rotira iz bilo kog razloga osim dosezanja maksimalne veličine fajla (max_binlog_size), trenutni binarni zapis nastavlja da se koristi. Klijent koji zatraži rotaciju dobiće grešku, a na serveru će biti zabeleženo upozorenje. Ako mysql.gtid_executed tabela nije dostupna za pisanje i max_binlog_size je dostignut, server će reagovati prema postavci binlog_error_action. IGNORE_ERROR će zaustaviti binarno zapisivanje i zabeležiti grešku, dok će ABORT_SERVER isključiti server.

MySQL server može periodično kompresovati tabelu mysql.gtid_executed radi uštede prostora. To se postiže zamenom svakog takvog skupa redova sa jednim redom koji obuhvata čitav interval identifikatora transakcija, kao što je prikazano u sledećem primeru:

source_uuid	interval_start	interval_end	gtid_tag
3E11FA47-71CA-11E1-9E33-C80AA9429562	31	31	Domain_1
3E11FA47-71CA-11E1-9E33-C80AA9429562	32	32	Domain_1
3E11FA47-71CA-11E1-9E33-C80AA9429562	33	33	Domain_1
3E11FA47-71CA-11E1-9E33-C80AA9429562	34	34	Domain_1
3E11FA47-71CA-11E1-9E33-C80AA9429562	35	35	Domain_1
3E11FA47-71CA-11E1-9E33-C80AA9429562	36	36	Domain_2
3E11FA47-71CA-11E1-9E33-C80AA9429562	37	37	Domain_2
3E11FA47-71CA-11E1-9E33-C80AA9429562	38	38	Domain_2
3E11FA47-71CA-11E1-9E33-C80AA9429562	39	39	Domain_2
3E11FA47-71CA-11E1-9E33-C80AA9429562	40	40	Domain_1
3E11FA47-71CA-11E1-9E33-C80AA9429562	41	41	Domain_1
3E11FA47-71CA-11E1-9E33-C80AA9429562	42	42	Domain_1
3E11FA47-71CA-11E1-9E33-C80AA9429562	43	43	Domain_1
3E11FA47-71CA-11E1-9E33-C80AA9429562	44	44	Domain_2
3E11FA47-71CA-11E1-9E33-C80AA9429562	45	45	Domain_2
3E11FA47-71CA-11E1-9E33-C80AA9429562	46	46	Domain_2
3E11FA47-71CA-11E1-9E33-C80AA9429562	47	47	Domain_1
3E11FA47-71CA-11E1-9E33-C80AA9429562	48	48	Domain_1

Slika 19. Nekompresovana tabela

Prethodni skup GTID-ova se zamenjuje sledećim skupom:

source_uuid	interval_start	interval_end	gtid_tag
3E11FA47-71CA-11E1-9E33-C80AA9429562	31	35	Domain_1
3E11FA47-71CA-11E1-9E33-C80AA9429562	36	39	Domain_2
3E11FA47-71CA-11E1-9E33-C80AA9429562	40	43	Domain_1
3E11FA47-71CA-11E1-9E33-C80AA9429562	44	46	Domain_2
3E11FA47-71CA-11E1-9E33-C80AA9429562	47	48	Domain_1

Slika 20. Kompresovana tabela

GTID životni ciklus

GTID [10] se dodjeljuje transakciji na izvoru i zapisuje u binarni zapis. Nakon što se prenese na repliku, GTID se koristi za identifikaciju i proveru statusa transakcija. Replika postavlja *gtid_next* na primljeni GTID i proverava da niko drugi nije započeo tu transakciju. Ako je GTID nov, transakcija se primenjuje na repliku i zapisuje u *mysql.gtid_executed*. Filtrirane transakcije takođe se zapisuju, osiguravajući integritet GTID-a. Na višenitnim replikama, transakcije se mogu primenjivati paralelno, što može stvarati praznine u skupu GTID-ova.

GTID automatsko pozicioniranje

GTID automatsko pozicioniranje [11] zamenjuje file-offset parove koji su ranije bili potrebni za određivanje tačaka za početak, zaustavljanje ili nastavak protoka podataka između izvora i replike. Kada se koriste GTID-ovi, sve potrebne informacije za sinhronizaciju replike sa izvorom dobijaju se direktno iz toka replikacije.

Da bi se pokrenula replika sa GTID baziranom replikacijom, potrebno je omogućiti opciju SOURCE_AUTO_POSITION u CHANGE REPLICATION SOURCE TO naredbi. Opcije SOURCE_LOG_FILE i SOURCE_LOG_POS specificiraju ime log fajla i početnu poziciju, ali sa GTID-ovima replika ne zahteva ove podatke.

SOURCE_AUTO_POSITION opcija je isključena po default-u. Ako je omogućena replikacija sa više izvora na replici, treba postaviti ovu opciju za svaki primenjivi kanal replikacije. Ponovno isključivanje ove opcije vraća repliku na fajl-baziranu replikaciju, što može rezultovati nevalidnim položajima kada je GTID_ONLY=ON.

Podešavanje replikacije preko GTID

U nastavku su dati koraci za podešavanje replikacije preko GTID-ova: [12]

- **Sinhronizacija servera**: Ako već postoji replikacija bez GTID-ova, oba servera se postavljaju kao read-only i čekaju da se sve transakcije završe pre nego što se nastavi proces. Ovo je važno jer transakcije bez GTID-ova ne mogu biti korišćene kada su GTID-ovi omogućeni.

```
mysql> SET @@GLOBAL.read_only = ON;
```

Slika 21. Postavljanje servera na read_only

- **Zaustavljanje servera**: Koristi se mysqladmin za zaustavljanje oba servera. Ovo osigurava da se serveri zaustave kontrolisano pre nego što se promeni konfiguracija.

```
mysqladmin -uusername -p shutdown
```

Slika 22. Zaustavljanje servera

- **Pokretanje servera sa omogućenim GTID-ovima**: Servere je potrebno pokreniti sa gtid_mode=ON i enforce-gtid-consistency=ON opcijama kako bi se omogućila GTID bazirana replikacija. Replika se pokreće sa --skip-replica-start opcijom da bi se privremeno onemogućilo automatsko pokretanje replikacije

```
gtid_mode=ON
enforce-gtid-consistency=ON
```

Slika 23. Podešavanje servera

- **Konfiguracija replike za GTID-Based Auto-Positioning**: Koristi se CHANGE REPLICATION SOURCE TO za konfigurisanje replike tako da koristi izvor sa GTID baziranim transakcijama i omogućava se auto-positioning. Ovo osigurava da se replika automatski sinhronizuje sa izvorom bez potrebe za ručnim postavljanjem pozicija log fajlova.

```
mysql> CHANGE REPLICATION SOURCE TO
    >     SOURCE_HOST = host,
    >     SOURCE_PORT = port,
    >     SOURCE_USER = user,
    >     SOURCE_PASSWORD = password,
    >     SOURCE_AUTO_POSITION = 1;
```

Slika 24. Konfiguracija replike

- **Kreiranje novog Backupa**: Budući da stari backup-ovi bez GTID-ova ne mogu biti korišćeni, potrebno je napraviti novi backup. Ovo osigurava postojanje sveže tačke oporavka koja se može koristiti u slučaju potrebe za obnavljanjem podataka.
- **Pokretanje replike i isključivanje read-only moda**: Potrebno je pokrenuti repliku i isključiti read-only mod kako bi serveri mogli prihvatati nove upite i nastaviti sa normalnim radom.

```
mysql> START REPLICA;
```

```
mysql> SET @@GLOBAL.read_only = OFF;
```

Slika 25. Pokretanje servera i postavljanje read_only na OFF

Ovaj proces omogućava efikasno postavljanje GTID bazirane replikacije u MySQL, obezbeđujući visok nivo pouzdanosti i lakšu administraciju replikacije između više servera.

Korišćenje GTID-ova za Failover i Scaleout

Korišćenje Global Transaction Identifiers (GTID-ova) u MySQL replikaciji omogućava efikasno upravljanje podacima i aktivnostima failovera [13]. GTID-ovi jedinstveno identifikuju skup događaja binarnih logova koji zajedno čine jednu transakciju. Ovi identifikatori su ključni za primenu promena u bazi podataka, jer server automatski preskače bilo koju transakciju koja ima identifikator koji je već obradio. Ovaj mehanizam je ključan za automatsko pozicioniranje replikacije i ispravan failover.

Postoji veliki broj tehnika za skaliranje i failover, a to su:

Jednostavna replikacija

Najjednostavniji način za uspostavljanje novog servera je da se konfiguriše kao replika postojećeg izvora koji već ima kompletnu istoriju izvršavanja, uz omogućene GTID-ove na oba servera. Ovaj pristup, iako jednostavan, može potrajati dok nova replika ne sustigne izvor, što ga čini manje pogodnim za brzi failover ili obnavljanje iz rezervne kopije.

Kopiranje podataka i transakcija na repliku

Ovaj metod omogućava uvoz snimka skupa podataka, binarnih logova i globalnih informacija o transakcijama sa izvora na novu repliku. Opcije uključuju korišćenje mysqldump za kreiranje dump fajla sa podešenom opcijom --source-data za uključivanje informacija o binarnim logovima, ili korišćenje alata poput mysqlbackup za kreiranje konzistentnog snimka, što je korisno za brže uspostavljanje replika.

Ubacivanje praznih transakcija

Umesto kopiranja kompletnih binarnih logova, ovaj metod zavisi od zapisivanja gtid_executed sa izvora i primene praznih transakcija na novom serveru za svaki identifikator transakcije. Ovo stvara server koji je u suštini snimak, ali koji vremenom može postati izvor kako se njegova istorija binarnih logova usklađuje sa izvorom i ostalim replikama.

Isključivanje transakcija sa gtid_purged

Ovaj pristup omogućava kreiranje servera koji je suštinski snimak, ali koji može postati izvor kako se njegova istorija binarnih logova usklađuje sa izvorom. Razlika u odnosu na prethodni metod je u tome što se umesto primene praznih transakcija direktno postavlja gtid_purged na repliku, bazirano na vrednosti gtid_executed sa izvora.

Obnavljanje replika sa GTID modom

Kada se obnavlja replika u GTID baziranom sistemu replikacije koji je naišao na grešku, koristi se mysqlbinlog za pronalaženje sledeće transakcije i kopiranje svega do COMMIT za tu transakciju. Ovo obezbeđuje da replika bude ponovo sinhronizovana i spremna za nastavak replikacije sa minimalnim prekidom.

Kombinacija ovih tehnika omogućava efikasno korišćenje GTID-ova za skaliranje i pouzdan failover u MySQL replikaciji, pružajući visok nivo pouzdanosti i skalabilnosti sistema.

Replikacija sa izvora bez GTID-ova na repliku sa GTID-ovima

Mogu se konfigurisati kanali replikacije da dodeljuju GTID-ove replikovanim transakcijama koje ih nemaju [14]. Ova funkcionalnost omogućava replikaciju sa servera izvora koji nije konfigurisan za korišćenje GTID-ova, na repliku koja koristi GTID-ove. Ukoliko je moguće, preporučuje se omogućavanje GTID-ova na serveru izvora.

Korišćenjem opcije ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS u CHANGE REPLICATION SOURCE TO izjavi, može se konfigurisati da se GTID-ovi dodeljuju na kanalu replikacije. Opcija LOCAL dodeljuje GTID koji uključuje UUID replike, dok opcija uuid dodeljuje GTID sa specifičnim UUID-om izvornog servera.

```
mysql> CHANGE REPLICATION SOURCE TO
    ->   ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS = 'uuid';
```

```
mysql> CHANGE REPLICATION SOURCE TO
    ->   ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS = LOCAL;
```

Slika 26. Opcije LOCAL i uuid u okviru ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTION

Da bi replika ispravno funkcionisala sa dodeljenim GTID-ovima, mora imati podešen gtid_mode=ON, što se ne može promeniti nakon što se ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS aktivira. Važno je napomenuti da replika koja koristi ovu opciju ne može biti promovisana u izvor tokom operacija poput failover-a, niti se backup napravljen sa replike može koristiti za vraćanje izvornog servera.

Ova opcija omogućava upotrebu mešovitih kanala replikacije na multi-source replikama, gde neki kanali koriste ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS, dok drugi ne.

Korišćenje ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS na kanalu replikacije ne podrazumeva potpunu implementaciju GTID-bazirane replikacije na tom kanalu. GTID-ovi koji su dodeljeni anonimnim transakcijama imaju značenje samo unutar te replike i nisu prenosivi ili poredljivi sa GTID-ovima drugih servera.

Ukoliko koristite više replika, samo replika koja direktno prima transakcije sa servera izvora bez GTID-ova treba da koristi ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS. Ostale replike u lancu mogu koristiti standardnu GTID-baziranu replikaciju međusobno, kao i za operacije kao što su failover i kreiranje novih replika iz backupa.

Ograničenja GTID-bazirane replikacije

GTID-bazirana replikacija zavisi od transakcija, što dovodi do nekih ograničenja u korišćenju određenih funkcionalnosti MySQL-a [15]. Ova sekcija pruža informacije o ograničenjima i limitacijama replikacije sa GTID-ovima.

Ažuriranja uključujući ne-transakcijski storage engine:

Prilikom korišćenja GTID-ova, ažuriranja tabela koje koriste ne-transakcijski storage enginove poput MyISAM ne mogu se izvršiti u istoj SQL naredbi ili transakciji sa ažuriranjima tabela koje koriste transactional storage enginove poput InnoDB. Ovo ograničenje nastaje jer ažuriranja na tabelama koje koriste nontransactional storage engine, kombinovana sa ažuriranjima na tabelama koje koriste transactional storage engine u istoj transakciji, mogu rezultirati dodjeljivanjem više GTID-ova istoj transakciji.

CREATE TABLE ... SELECT naredbe:

Za storage engine koji podržavaju atomic DDL, CREATE TABLE ... SELECT je zapisan u binarnom logu kao jedna transakcija.

Privremene tabele:

Ako je binlog_format postavljen na STATEMENT, CREATE TEMPORARY TABLE i DROP TEMPORARY TABLE naredbe ne mogu se koristiti unutar transakcija, procedura, funkcija i okidača kada su GTID-ovi u upotrebi na serveru (tj. kada je enforce_gtid_consistency sistemski varijabla postavljena na ON). Kada je binlog_format postavljen na ROW ili MIXED, ove naredbe su dozvoljene unutar transakcije, procedure, funkcije ili okidača, ali se ne pišu u binarni zapis i stoga se ne replikuju na replike.

Sprečavanje izvršenja nepodržanih naredbi:

Da bi se sprečilo izvršenje naredbi koje bi mogle uzrokovati probleme sa GTID-baziranom replikacijom, svi serveri moraju biti pokrenuti sa opcijom --enforce-gtid-consistency pri omogućavanju GTID-ova.

Preskakanje transakcija:

sql_replica_skip_counter nije dostupan kada se koristi GTID-bazirana replikacija. Umesto toga, koristi se vrednost gtid_executed varijable sa izvora. Ako se omogući dodeljivanje GTID-ova na kanalu replikacije korišćenjem opcije

ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS,

sql_replica_skip_counter je dostupan.

Ignorisanje servera:

IGNORE_SERVER_IDS ne može se koristiti sa CHANGE REPLICATION SOURCE TO kada su u upotrebi GTID-ovi, jer su transakcije koje su već primenjene automatski ignorisane. Pre pokretanja GTID-bazirane replikacije, proverava se i brišu se sve liste ignorisanih server ID-ova koji su prethodno postavljeni na uključenim serverima.

Ova ograničenja važe za korišćenje GTID-ova u MySQL-u i važno ih je uzeti u obzir prilikom konfiguracije replikacije.

Multi-Source Replikacija

MySQL 8.4 omogućava višestruku replikaciju izvora, gde replika može primati transakcije iz više izvora istovremeno. U ovakvoj topologiji, svaki izvor ima svoj kanal replikacije na replici. Ova funkcionalnost je korisna za:

- Backup više servera na jedan server.
- Spajanje delova tabela.
- Konsolidaciju podataka sa više servera na jedan.

Višestruka replikacija ne rešava konflikte pri primeni transakcija, što ostavlja aplikaciji da rešava ako je potrebno.

Svaki kanal na višestrukoj replici mora replicirati iz različitog izvora radi jedinstvenosti server ID-ova. Može se konfigurisati i kao višenitna replika, gde svaki kanal ima određeni broj applier niti plus koordinatora. MySQL 8.4 podržava i filtere replikacije na specifičnim kanalima, korisne kada isti podaci postoje na više izvora.

Konfiguracija Multi-Source replikacije

Za multi-source replikaciju potrebna su najmanje dva izvora i jedna replika. Pretpostavimo da imamo dva izvora, *source1* i *source2*, i jednu repliku *replicahost*. Replika replicira bazu podataka *db1* sa *source1* i bazu *db2* sa *source2*.

Koraci za Konfiguraciju

- Izbor tipa replikacije:
  - GTID-bazirana replikacija
  - Replikacija bazirana na poziciji binarnog loga
- Podešavanje replike:
  - Replike u multi-source replikaciji zahtevaju TABLE repozitorijume za konekcione i applier metapodatke, što je podrazumevano u MySQL 8.4.
- Kreiranje korisničkog naloga na izvorima:
  - Kreiranje korisničkog naloga na svakom izvoru koji replika koristi za povezivanje.
  - Moguće je koristiti isti nalog na svim izvorima, a moguće je koristiti i različite naloge.
  - Nalog koji je samo za potrebe replikacije treba da ima REPLICATION SLAVE privilegiju.

Primer za kreiranje korisnika *ted* koji se povezuje sa replike *replicahost*:

```
mysql> CREATE USER 'ted'@'replicahost' IDENTIFIED BY 'password';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'ted'@'replicahost';
```

Slika 28. Kreiranje korisnika na izvoru

Obezbeđivanje replike sa više izvora za replikaciju zasnovanu na GTID

Kada izvorni serveri u topologiji multi-source replikacije već imaju podatke, moguće je uštedeti vreme tako što će se unapred pripremiti replika sa odgovarajućim podacima. Najbolji način za to je korišćenje mysqldump za kreiranje dump fajlova na svakom od izvora, a zatim korišćenje mysql klijenta za uvoz dump fajlova na repliku.

Koraci:

- Kreiranje dump fajlova za baze podataka na izvorima:

```
mysqldump -u<user> -p<password> --single-transaction --triggers --routines --set-gtid-purged=ON --databases db1 > dumpM1.sql
mysqldump -u<user> -p<password> --single-transaction --triggers --routines --set-gtid-purged=ON --databases db2 > dumpM2.sql
```

Slika 29. Izvoz podataka sa oba izvora

- Izvlačenje vrednosti gtid_purged iz dump fajlova:

```
cat dumpM1.sql | grep GTID_PURGED | perl -p0 -e 's#/\*.*?\*/##sg' | cut -f2 -d'=' | cut -f2 -d$'\''
cat dumpM2.sql | grep GTID_PURGED | perl -p0 -e 's#/\*.*?\*/##sg' | cut -f2 -d'=' | cut -f2 -d$'\''
```

Slika 30. Izvlačenje vrednosti gtid_purged iz dump fajlova

  - Rezultat u svakom slučaju treba da bude GTID skup, na primer:

```
source1:   2174B383-5441-11E8-B90A-C80AA9429562:1-1029
source2:   224DA167-0C0C-11E8-8442-00059A3C7B00:1-2695
```

Slika 31. Rezultal kao skup GTID-ova

- Brisanje linija koje sadrže SET @@GLOBAL.gtid_purged iz dump fajlova:

```
sed '/GTID_PURGED/d' dumpM1.sql > dumpM1_nopurge.sql
sed '/GTID_PURGED/d' dumpM2.sql > dumpM2_nopurge.sql
```

Slika 32. Brisanje linija koje sadrže SET @@GLOBAL.gtid_purged iz dump fajlova

- Uvoženje dump fajlova na repliku:

```
mysql -u<user> -p<password> < dumpM1_nopurge.sql
mysql -u<user> -p<password> < dumpM2_nopurge.sql
```

Slika 33. Uvoz dump fajlova na repliku

- Resetovanje binarnih logova i GTID-ova na replici i postavljanje gtid_purged na objedinjene vrednosti:

```
mysql> RESET BINARY LOGS AND GTIDS;
mysql> SET @@GLOBAL.gtid_purged = "2174B383-5441-11E8-B90A-C80AA9429562:1-1029, 224DA167-0C0C-11E8-8442-00059A3C7B00:1-2695";
```

Slika 34. Resetovanje binarnih logova i GTID-ova

Dodavanje GTID-baziranih izvora u Multi-Source repliku

Ovi koraci pretpostavljaju da su omogućeni GTID-ovi za transakcije na izvorima sa gtid_mode=ON, kreiran korisnik za replikaciju, obezbeđeno da replika koristi TABLE metapodatke za replikaciju, i pripremljena replika sa podacima sa izvora ukoliko je to potrebno.

Koraci:

- Konfigurisanje kanala za svaki izvor na replici koristeći CHANGE REPLICATION SOURCE TO:
  - Koristi se FOR CHANNEL klauzula za specificiranje kanala. Za GTID-baziranu replikaciju koristi se GTID automatsko pozicioniranje (SOURCE_AUTO_POSITION).
  - Na primer, da se doda *source1* i *source2* kao izvore na repliku, koristi se mysql klijent za izdavanje sledeće izjave:

```
mysql> CHANGE REPLICATION SOURCE TO SOURCE_HOST="source1", SOURCE_USER="ted",
SOURCE_PASSWORD="password", SOURCE_AUTO_POSITION=1 FOR CHANNEL "source_1";
mysql> CHANGE REPLICATION SOURCE TO SOURCE_HOST="source2", SOURCE_USER="ted",
SOURCE_PASSWORD="password", SOURCE_AUTO_POSITION=1 FOR CHANNEL "source_2";
```

Slika 35. Konfiguracija kanala za svaki izvor

- Postavljanje replika da replicira samo bazu podataka db1 sa source1, i samo bazu db2 sa source2:
  - Koristi se mysql klijent da izda CHANGE REPLICATION FILTER izjavu za svaki kanal:

```
mysql> CHANGE REPLICATION FILTER REPLICATE_WILD_DO_TABLE = ('db1.%') FOR CHANNEL "source_1";
mysql> CHANGE REPLICATION FILTER REPLICATE_WILD_DO_TABLE = ('db2.%') FOR CHANNEL "source_2";
```

Slika 36. Repliciranje tacno određenih baza po kanalu

Ovi koraci omogućavaju postavljanje replikacije iz više izvora na jednu repliku, pri čemu svaki izvor ima svoj kanal i replicira samo specificirane baze podataka. GTID automatsko pozicioniranje olakšava sinhronizaciju sa izvorima, dok filteri za replikaciju osiguravaju da se repliciraju samo željeni podaci.

Dodavanje izvora baziranih na binarnim logovima u Multi-Source repliku

Ovi koraci pretpostavljaju da je binarno logovanje omogućeno na izvoru (što je podrazumevano), replika koristi TABLE metapodatke za replikaciju (što je podrazumevano u MySQL 8.4), i da je kreiran korisnik za replikaciju i da je zabeleženo trenutno ime i pozicija binarnog log fajla.

Koraci:

- Konfigurisanje kanala za svaki izvor na replici koristeći CHANGE REPLICATION SOURCE TO:
  - Koristi se FOR CHANNEL klauzula za specificiranje kanala. Na primer, da se dodaju *source1* i *source2* kao izvori na repliku, koristi se mysql klijent sa sledećim izjavama:

```
mysql> CHANGE REPLICATION SOURCE TO SOURCE_HOST="source1", SOURCE_USER="ted", SOURCE_PASSWORD="password",
SOURCE_LOG_FILE='source1-bin.000006', SOURCE_LOG_POS=628 FOR CHANNEL "source_1";

mysql> CHANGE REPLICATION SOURCE TO SOURCE_HOST="source2", SOURCE_USER="ted", SOURCE_PASSWORD="password",
SOURCE_LOG_FILE='source2-bin.000018', SOURCE_LOG_POS=104 FOR CHANNEL "source_2";
```

Slika 37. Konfiguracija kanala za svaki izvor

- Postavljanje replike da replicira samo bazu podataka db1 sa source1, i samo bazu db2 sa source2:
  - Koristi se mysql klijent da se izda CHANGE REPLICATION FILTER izjava za svaki kanal:

```
mysql> CHANGE REPLICATION FILTER REPLICATE_WILD_DO_TABLE = ('db1.%') FOR CHANNEL "source_1";
mysql> CHANGE REPLICATION FILTER REPLICATE_WILD_DO_TABLE = ('db2.%') FOR CHANNEL "source_2";
```

Slika 38. Repliciranje tacno određenih baza po kanalu

Ovi koraci omogućavaju postavljanje replikacije iz više izvora baziranih na binarnim logovima na jednu repliku, pri čemu svaki izvor ima svoj kanal i replicira samo specificirane baze podataka. Filteri za replikaciju osiguravaju da se repliciraju samo željeni podaci.

Upravljanje Multi-Source replikacijom

Multi-source replikacija omogućava replici da prima transakcije iz više izvora paralelno. Ova funkcionalnost se koristi za konsolidaciju podataka iz više servera i za bekapovanje.

Pokretanje Multi-Source replika:

Nakon što se dodaju kanali za sve izvore replikacije, izdaje se START REPLICA izjava za pokretanje replikacije. Kada je omogućeno više kanala na replici, može se odabrati da se pokrenu svi kanali ili samo određeni kanal. Na primer, da bi se zasebno pokrenula dva kanala, koristi se mysql klijent i izdaju se sledeće izjave:

```
mysql> START REPLICA FOR CHANNEL "source_1";
mysql> START REPLICA FOR CHANNEL "source_2";
```

Slika 39. Pokretanje replikacije

Proveravanje statusa svakog kanala pomoću:

```
mysql> SHOW REPLICA STATUS FOR CHANNEL "source_1"\G
mysql> SHOW REPLICA STATUS FOR CHANNEL "source_2"\G
```

Slika 40. Provera statusa replikacije

Zaustavljanje Multi-Source replika

Izjava STOP REPLICA može se koristiti za zaustavljanje multi-source replikacije. Podrazumevano, ako se koristi izjava STOP REPLICA na multi-source replici, svi kanali će biti zaustavljeni. Opciono, može se koristiti klauzula FOR CHANNEL da bi se zaustavio samo određeni kanal.

Za zaustavljanje svih trenutno konfigurisanih kanala replikacije:

```
mysql> STOP REPLICA;
```

Slika 41. Zaustavljanje svih kanala

Za zaustavljanje samo imenovanih kanala, koristi se klauzula FOR CHANNEL:

```
mysql> STOP REPLICA FOR CHANNEL "source_1";
```

Slika 42. Zaustavljanje određenog kanala

Resetovanje Multi-Source replika

Izjava RESET REPLICA može se koristiti za resetovanje multi-source replikacije. Podrazumevano, ako se koristi izjava RESET REPLICA na multi-source replici, svi kanali će biti resetovani. Opciono, može se koristiti klauzula FOR CHANNEL da se resetuje samo određeni kanal.

Za resetovanje svih trenutno konfigurisanih kanala replikacije:

```
mysql> RESET REPLICA;
```

Slika 43. Resetovanje svih kanala

Za resetovanje samo imenovanih kanala, koristi se klauzula FOR CHANNEL:

```
mysql> RESET REPLICA FOR CHANNEL "source_1";
```

Slika 44. Resetovanje određenog kanala

Implementacija Replikacije

Replikacija [16] u MySQL-u zasniva se na tome da server izvora prati sve promene u svojim bazama podataka (ažuriranja, brisanja itd.) u svom binarnom logu. Binarni log je zapisnik svih događaja koji menjaju strukturu ili sadržaj baze podataka od trenutka kada je server pokrenut. Obično, SELECT izjave nisu zabeležene jer ne menjaju ni strukturu ni sadržaj baze podataka.

Svaka replika koja se povezuje na izvor zahteva kopiju binarnog loga. Dakle, replika povlači podatke sa izvora, umesto da izvor šalje podatke replici. Replika takođe izvršava događaje iz binarnog loga koje primi, čime se originalne promene ponavljaju upravo onako kako su napravljene na izvoru. Tabele se kreiraju ili se njihova struktura menja, a podaci se ubacuju, brišu i ažuriraju prema promenama koje su prvobitno napravljene na izvoru.

Svaka replika je nezavisna, pa se ponavljanje promena iz binarnog loga izvora dešava nezavisno na svakoj replici koja je povezana na izvor. Pored toga, pošto svaka replika prima kopiju binarnog loga samo na zahtev iz izvora, replika je u mogućnosti da čita i ažurira kopiju baze podataka sopstvenim tempom i može započeti i zaustaviti proces replikacije po želji bez uticaja na mogućnost ažuriranja na najnoviji status baze podataka na bilo kojoj strani, izvornoj ili replici.

Formati replikacije

Replikacija u MySQL-u se bazira na beleženju svih promena u bazi podataka izvornog servera u binarnom logu. Binarni log služi kao zapis svih događaja koji menjaju strukturu ili sadržaj baze podataka. Replikacija se zatim odvija tako što replika preuzima ove zapise iz binarnog loga i izvršava ih, ponavljajući originalne promene napravljene na izvornom serveru.

Postoje različiti formati replikacije u MySQL-u, koji odgovaraju formatu binarnog logovanja korišćenog prilikom beleženja događaja:

- **Replikacija zasnovana na izjavama (SBR)**: Izvorni server beleži SQL izjave u binarnom logu. Replika zatim izvršava ove SQL izjave, replicirajući promene napravljene na izvoru.
- **Replikacija zasnovana na redovima (RBR)**: Izvorni server beleži događaje koji ukazuju na to kako su pojedinačni redovi tabele promenjeni. Replika zatim kopira ove događaje, replicirajući promene na nivou redova.
- **Mešoviti format replikacije**: MySQL može koristiti kombinaciju oba formata, u zavisnosti od toga šta je prikladnije za određenu promenu. Ovaj format se naziva mešovito logovanje.

Podrazumevani metod logovanja u MySQL-u je zasnovan na redovima. Međutim, moguće je konfigurirati MySQL da koristi bilo koji od tri formata logovanja.

```
mysql> show variables like '%binlog_format%';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| binlog_format | ROW   |
+---------------+-------+
1 row in set (0.00 sec)
```

Slika 45. Default format je ROW

Promena formata binarnog logovanja se kontroliše pomoću promenljive binlog_format, koja može biti postavljena na sesijski ili globalni nivo. Ova promena može imati različite efekte, u zavisnosti od trenutnog stanja servera i sesija koje su povezane.

```
mysql> set session binlog_format = statement;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> show variables like '%binlog_format%';
+---------------+-----------+
| Variable_name | Value     |
+---------------+-----------+
| binlog_format | STATEMENT |
+---------------+-----------+
1 row in set (0.00 sec)
```

Slika 46. Postavljanje formata na STATEMENT

```
mysql> set session binlog_format = mixed;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> show variables like '%binlog_format%';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| binlog_format | MIXED |
+---------------+-------+
1 row in set (0.00 sec)
```

Slika 47. Postavljanje formata na MIXED

Svaki format replikacije ima svoje prednosti i nedostatke. Na primer, replikacija zasnovana na izjavama može imati problema sa replikacijom složenih operacija poput procedura ili okidača, dok replikacija zasnovana na redovima bolje podržava ove operacije.

Prednosti i mane replikacije zasnovane na izjavama i redovima

Prednosti replikacije zasnovane na izjavama (SBR)

- Dokazana tehnologija: Dugogodišnje iskustvo i pouzdanost.
- Manje podataka u log datotekama: Kada ažuriranja ili brisanja utiču na mnogo redova, zapisivanje SQL izjava zahteva manje prostora u log datotekama, što ubrzava proces pravljenja i vraćanja bekapa.
- Mogućnost audita (nadzora): Log datoteke sadrže sve izjave koje su promenile podatke, što omogućava audit baze podataka.

Mane replikacije zasnovane na izjavama (SBR)

- Nedeterminističke izjave: Neke SQL izjave, poput onih koje zavise od funkcija koje nisu determinističke ili složenih izraza, teško je replicirati.
- Veći broj zaključavanja redova: Izjave poput INSERT ... SELECT i složene UPDATE operacije zahtevanju više zaključavanja redova nego kod replikacije zasnovane na redovima.
- Blokade kod AUTO_INCREMENT: INSERT izjave sa AUTO_INCREMENT mogu blokirati druge nekolizirajuće INSERT izjave.
- Problemi sa složenim izjavama: Replike moraju izvršiti celu izjavu, što može povećati greške u složenim scenarijima.

Prednosti replikacije zasnovane na redovima (RBR)

- Sigurnost: Sve promene mogu biti replicirane, što čini ovaj metod najsigurnijim.
- Manje zaključavanja redova: Zahteva manje zaključavanja redova na izvoru i replici, što povećava konkurentnost.
- Bolja podrška za složene operacije: Smanjuje probleme sa složenim DML izjavama kao što su INSERT sa AUTO_INCREMENT, UPDATE ili DELETE sa WHERE klauzulama koje ne koriste ključeve.

Mane replikacije zasnovane na redovima (RBR)

- Više podataka u logu: Svaka promena reda se beleži, što može generisati mnogo više podataka u binarnom logu. Ovo može produžiti vreme potrebno za pravljenje i vraćanje bekapa, kao i izazvati probleme sa konkurentnošću.
- Nemogućnost uvida u izvršene izjave: Na replici nije moguće videti koje izjave su primljene i izvršene, ali je moguće videti promene podataka korišćenjem mysqlbinlog alata.

- Problemi sa velikim BLOB vrednostima: Replikacija velikih BLOB vrednosti može trajati duže zbog logovanja celih kolona.

Za većinu korisnika, mešoviti format replikacije pruža najbolji balans između integriteta podataka i performansi. Međutim, odabir između replikacije zasnovane na izjavama ili redovima zavisi od specifičnih potreba i scenarija upotrebe.

Korišćenje replikacije zasnovane na redovima (RBR)

MySQL podržava tri formata zapisivanja u binarni log: zapisivanje zasnovano na izjavama (SBL), zapisivanje zasnovano na redovima (RBL) i mešoviti format zapisivanja. Izbor između RBR i SBR zavisi od aplikacije i okruženja jer tip binarnog loga utiče na veličinu i efikasnost zapisivanja.

Poznati problemi pri korišćenju RBR:

Privremene tabele	Privremene tabele se ne repliciraju kada se koristi RBR ili mešoviti format jer nema potrebe za tim. Takođe, privremene tabele mogu biti pročitane samo iz niti koja ih je kreirala.
Promena formata zapisivanja	Može se promeniti format zapisivanja iz SBR u RBR čak i kada su kreirane privremene tabele, ali se ne može promeniti iz RBR ili mešovitog formata u SBR zbog izostavljanja CREATE TEMPORARY TABLE izjava u binarnom logu.
Sinhronizacija netransakcionih tabela	Kada mnogo redova bude izmenjeno, promene se dele na nekoliko događaja koji se zapisuju u binarni log kada se izjava izvrši. Na replici se zaključava tabela i primenjuju se promene u serijama, što može biti efektivno zavisno od korišćenog skladišnog mehanizma.
Latencija i veličina binarnog loga	RBR zapisuje promene za svaki red u binarni log, što može brzo povećati njegovu veličinu i produžiti vreme potrebno za primenu promena na replici.
Čitanje binarnog loga	Alat mysqlbinlog prikazuje događaje zapisane u RBR formatu kao binarne kodirane nizove. Upotrebom opcija --base64-output=DECODE-ROWS i --verbose moguće je formatirati sadržaj binarnog loga da bude čitljiv.
Greške u izvršavanju binarnog loga	Podešavanje opcije replica_exec_mode na IDEMPOTENT može biti korisno za NDB Cluster replikaciju, ali za druge scenarije, vrednost STRICT je obično dovoljna.
Filtriranje zasnovano na server ID-u nije podržano	Filtriranje na osnovu server ID-a pomoću IGNORE_SERVER_IDS opcije ne funkcioniše kada je gtid_mode=ON. Alternativno, filtriranje se može postići korišćenjem WHERE klauzule sa server_id sistemskom promenljivom u izjavama UPDATE i DELETE, ali ovo ne funkcioniše pravilno sa RBR.
Netransakcione tabele i	Ako je replikacioni server zaustavljen dok replika nit ažurira netransakcionu tabelu, baza podataka može postati nekonzistentna.

zaustavljene replike	Preporučuje se korišćenje transakcionog skladišnog mehanizma kao što je InnoDB za sve tabele koje se repliciraju koristeći RBR.

Određivanje sigurnih i nesigurnih izjava u binarnom logovanju

Sigurnost izjava u MySQL replikaciji odnosi se na to da li se izjava i njeni efekti mogu ispravno replikovati koristeći format baziran na izjavama (SBR). Ako je izjava sigurna, nazivamo je "sigurnom", u suprotnom, nazivamo je "nesigurnom".

Rukovanje sigurnim i nesigurnim izjavama

Kada se koristi zapisivanje zasnovano na redovima (RBR), ne pravi se razlika između sigurnih i nesigurnih izjava.

Kada se koristi mešoviti format zapisivanja (MIXED), nesigurne izjave se zapisuju koristeći format zasnovan na redovima, dok se sigurne izjave zapisuju koristeći format zasnovan na izjavama.

Kada se koristi zapisivanje zasnovano na izjavama (SBR), nesigurne izjave generišu upozorenje, dok se sigurne izjave zapisuju normalno.

Svaka nesigurna izjava generiše upozorenje. Da bi se sprečilo prekomerno povećanje fajlova sa greškama, MySQL ima mehanizam za suzbijanje upozorenja. Ako se 50 najnovijih upozorenja ER_BINLOG_UNSAFE_STATEMENT generiše više od 50 puta u bilo kom periodu od 50 sekundi, aktivira se suzbijanje upozorenja. Ovo uzrokuje da se takva upozorenja ne zapisuju u fajl sa greškama, već se umesto toga zapisuje beleška poput: "The last warning was repeated N times in last S seconds".

Izjave koje se smatraju nesigurnim

- Izjave koje sadrže sistemske funkcije koje mogu vratiti različitu vrednost na replici: Ove funkcije uključuju FOUND_ROWS(), GET_LOCK(), IS_FREE_LOCK(), IS_USED_LOCK(), LOAD_FILE(), RAND(), RELEASE_LOCK(), ROW_COUNT(), SESSION_USER(), SLEEP(), SOURCE_POS_WAIT(), SYSDATE(), SYSTEM_USER(), USER(), UUID(), i UUID_SHORT().
- Nedeterminističke funkcije koje se ne smatraju nesigurnim: Iako ove funkcije nisu determinističke, one se tretiraju kao sigurne za potrebe zapisivanja i replikacije: CONNECTION_ID(), CURDATE(), CURRENT_DATE(), CURRENT_TIME(), CURRENT_TIMESTAMP(), CURTIME(), LAST_INSERT_ID(), LOCALTIME(), LOCALTIMESTAMP(), NOW(), UNIX_TIMESTAMP(), UTC_DATE(), UTC_TIME(), i UTC_TIMESTAMP().
- Reference na sistemske promenljive: Većina sistemskih promenljivih se ne replicira ispravno koristeći format baziran na izjavama.
- Plugin za pun tekst: Ovaj plugin može raditi različito na različitim MySQL serverima, pa su sve izjave koje zavise od njega tretirane kao nesigurne.
- Trigeri ili programi koji ažuriraju tabelu sa AUTO_INCREMENT kolonom: Ovo je nesigurno jer redosled ažuriranja može biti različit na izvornom serveru i replici.

- INSERT ... ON DUPLICATE KEY UPDATE izjave na tabelama sa više primarnih ili jedinstvenih ključeva: Ove izjave su osetljive na redosled provere ključeva, što nije determinističko.
- Ažuriranja koristeći LIMIT: Redosled dohvatanja redova nije specificiran i stoga se smatra nesigurnim.
- Pristupi ili reference na log tabele: Sadržaj sistemske log tabele može se razlikovati između izvornog servera i replike.
- Netransakcione operacije nakon transakcionih operacija: Unutar transakcije, omogućavanje netransakcionih čitanja ili pisanja nakon transakcionih čitanja ili pisanja se smatra nesigurnim.
- Pristupi ili reference na samologujuće tabele: Sva čitanja i pisanja na samologujuće tabele se smatraju nesigurnim.
- LOAD DATA izjave: Ove izjave se tretiraju kao nesigurne i kada je binlog_format=MIXED, izjava se loguje koristeći format zasnovan na redovima.
- XA transakcije: Ako se dve XA transakcije izvrše paralelno na izvornom serveru, a pripremaju se na replici u obrnutom redosledu, mogu se pojaviti problemi sa zaključavanjem.

Ova pravila i preporuke pomažu da se obezbedi ispravna replikacija i minimalizuju potencijalne greške i nekonzistentnosti u MySQL replikaciji.

Praktične primene replikacije

Replikacija u MySQL-u se često koristi u različitim industrijama za postizanje visoke dostupnosti, balansiranje opterećenja i disaster recovery. Evo nekoliko primera:

- **Implementacija Visoke Dostupnosti (HA) korišćenjem replikacije**: Ovaj primer fokusira se na implementaciju visoko dostupnog sistema (HA) pomoću MySQL-ove replikacije. Glavni server konfigurisan je kao izvor (source) koji emituje transakcije ka jednom ili više odredišnih (replica) servera. Korišćenjem GTID-ova za automatsko pozicioniranje, može se postići brzo prebacivanje na repliku u slučaju pada glavnog servera.
- **Multi-Source replikacija za distribuirane sisteme**: Ovaj primer istražuje konfiguraciju multi-source replikacije za skaliranje i distribuiranje baza podataka. Svaki izvor (source) predstavlja različitu aplikaciju ili servis koji generiše transakcije. Korišćenjem kanalskih filtera, ovaj primer pokazuje kako se može kontrolisati koji izvori doprinose kojim podacima na svakoj replici, čime se optimizuje resursna potrošnja i performanse sistema.
- **Replikacija za sigurnost i bezbednost podataka**: Ovaj primer slučaja istražuje kako se MySQL-ova replikacija može koristiti za osiguranje podataka i njihovu zaštitu od gubitaka. Fokus je na konfiguraciji periodičnih sigurnosnih tačaka (checkpoints) na odredišnim serverima kako bi se minimizovali gubici podataka u slučaju neočekivanog kvara sistema. Dodatno, istražuje se upotreba replikacije za geografsko razdvajanje podataka radi bolje otpornosti na katastrofe.
- **Optimizacija performansi sa paralelnim apliciranjem**: U ovom primeru, istražuju se tehnike optimizacije performansi korišćenjem paralelnog apliciranja u multi-source replikaciji. Podešavanjem broja applier thread-ova za svaki kanal, ovaj primer analizira kako se mogu ubrzati procesi replikacije i smanjiti ukupna latencija u distribuiranim sistemima.
- **Skaliranje i rasterećenje opterećenja korišćenjem GTID-ova**: Ovaj primer istražuje upotrebu GTID-ova za skaliranje i rasterećenje opterećenja u sistemima sa visokom dostupnošću. Analizira se kako se GTID-ovi mogu koristiti za lako dodavanje novih izvora podataka u postojeću replikaciju bez potrebe za kompleksnim ručnim intervencijama.

Svaki od ovih primera može biti detaljno razrađen sa koracima implementacije, konfiguracijom servera, primenom odgovarajućih sigurnosnih mera i strategija upravljanja podacima. Ovi scenariji ne samo da ilustruju teorijske koncepte replikacije, već i pružaju praktične smernice za implementaciju u stvarnim okruženjima.

Zaključak

Ovaj rad pokriva različite aspekte MySQL replikacije, uključujući osnovnu konfiguraciju, implementaciju i napredne tehnike kao što su replikacija zasnovana na poziciji binarnog log-a, GTID replikacija i multi-source replikacija.

Replikacija zasnovana na poziciji binarnog log-a omogućava osnovno sinhronizovanje podataka između izvorišnog i odredišnog servera. Ovaj metod je jednostavan za implementaciju, ali zahteva pažljivo praćenje binarnih logova i pozicija. Sa druge strane, GTID replikacija pruža napredniji način praćenja transakcija kroz jedinstvene identifikatore, olakšavajući failover i scaleout operacije. GTID omogućava automatsko pozicioniranje i smanjuje rizik od nekonzistentnosti podataka između izvora i replike.

Multi-source replikacija dodatno povećava fleksibilnost sistema omogućavajući replikama da primaju podatke od više izvora. Ovo je posebno korisno u scenarijima gde je potrebno objediniti podatke iz više različitih baza. Konfiguracija multi-source replikacije zahteva pažljivo upravljanje kanalima replikacije, ali pruža značajne prednosti u smislu raspodele opterećenja i konsolidacije podataka.

Razmatrane su i prednosti i mane različitih formata replikacije, kao što su replikacija zasnovana na izjavama (SBR) i replikacija zasnovana na redovima (RBR). SBR je lakši za implementaciju, ali može biti nesiguran za određene vrste izjava. RBR je pouzdaniji u smislu konzistentnosti podataka, ali generiše veće količine logova. Određivanje sigurnih i nesigurnih izjava u binarnom logovanju je ključno za održavanje integriteta sistema.

Sve ove tehnike i koncepti doprinose izgradnji pouzdanih, skalabilnih i visoko dostupnih MySQL sistema. Implementacija MySQL replikacije zahteva pažljivu konfiguraciju i praćenje, ali koristi koje donosi u pogledu performansi, sigurnosti podataka i fleksibilnosti sistema čine ovaj napor vrednim. Replikacija omogućava neprekidno pružanje usluga, efikasno upravljanje podacima i brzu reakciju na eventualne kvarove, što je od suštinskog značaja za moderne poslovne aplikacije.

Literatura

[1] MySQL™, Chapter 19 Replication,
Datum pristupa: 04.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication.html

[2] MySQL™, Binary Log File Position Based Replication Configuration Overview,
Datum pristupa: 04.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/binlog-replication-configuration-overview.html

[3] MySQL™, Setting the Replication Source Configuration,
Datum pristupa: 04.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-howto-masterbaseconfig.html

[4] MySQL™, Setting the Replica Configuration,
Datum pristupa: 06.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-howto-slavebaseconfig.html

[5] MySQL™, Creating a User for Replication,
Datum pristupa: 06.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-howto-repuser.html

[6] MySQL™, Obtaining the Replication Source Binary Log Coordinates,
Datum pristupa: 07.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-howto-masterstatus.html

[7] MySQL™, Choosing a Method for Data Snapshots,
Datum pristupa: 07.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-snapshot-method.html

[8] MySQL™, Setting Up Replicas,
Datum pristupa: 10.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-setup-replicas.html

[9] MySQL™, Replication with Global Transaction Identifiers,
Datum pristupa: 10.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-gtids.html

[10] MySQL™, GTID Life Cycle,
Datum pristupa: 13.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-gtids-lifecycle.html

[11] MySQL™, GTID Auto-Positioning,
Datum pristupa: 13.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-gtids-auto-positioning.html

[12] MySQL™, Setting Up Replication Using GTIDs,
Datum pristupa: 15.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-gtids-howto.html

[13] MySQL™, Using GTIDs for Failover and Scaleout,
Datum pristupa: 16.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-gtids-failover.html

[14] MySQL™, Replication From a Source Without GTIDs to a Replica With GTIDs,
Datum pristupa: 20.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-gtids-assign-anon.html

[15] MySQL™, Restrictions on Replication with GTIDs,
Datum pristupa: 21.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-gtids-restrictions.html

[16] MySQL™, Replication Implementation,
Datum pristupa: 21.06.2024.
Dostupno na: https://dev.mysql.com/doc/refman/8.4/en/replication-implementation.html