|  |  |  |
| --- | --- | --- |
|  | UNIVERZITET U NIŠU  ELEKTRONSKI FAKULTET |  |

Miljan Denić

**REPLIKACIJA KOD POSTGRESQL BAZE PODATAKA**

Seminarski rad

Studijski program: Računarstvo i informatika

Modul: Softversko inženjerstvo

|  |
| --- |
| Kandidat: |
|  |
| Miljan Denić, br. ind. 1517 |

Niš, jun 2023. godina

SADRžaj

[1. uvod 3](#_Toc137535557)

[2. log-shipping Replikacija 3](#_Toc137535558)

[2.1. Pripremanje primarnog servera za standby servere 3](#_Toc137535559)

[2.2. Postavljanje Standby servera 4](#_Toc137535560)

[3. Streaming Replikacija 5](#_Toc137535561)

[3.1. Autentikacija 6](#_Toc137535562)

[3.2. Monitoring 7](#_Toc137535563)

[3.3. Slotovi za replikaciju 7](#_Toc137535564)

[4. Kaskadna replikacija 8](#_Toc137535565)

[5. Sinhrona replikacija 9](#_Toc137535566)

[5.1. Više sinhronih standby servera 11](#_Toc137535567)

[5.2. Planiranje performansi 12](#_Toc137535568)

[5.3. Planiranje visoke dostupnosti 12](#_Toc137535569)

[5.4. Kontinuirano arhiviranje na standby serveru 13](#_Toc137535570)

[6. prelazak na rezervni režim rada 14](#_Toc137535571)

[7. Hot standby 15](#_Toc137535572)

# uvod

Replikacija podataka je ključni koncept u upravljanju bazama podataka koji omogućava dupliranje podataka na više servera radi obezbeđivanja visoke dostupnosti i pouzdanosti sistema. U kontekstu PostgreSQL baze podataka, replikacija se odnosi na proces kopiranja podataka sa primarnog servera na jedan ili više standby servera. Ovi standby serveri su u stanju da preuzmu ulogu primarnog servera u slučaju neuspeha ili prekida rada primarnog servera, čime se osigurava kontinuitet poslovanja i zaštita podataka.

U ovom radu će se detaljnije istražiti različite metode replikacije koje su dostupne u PostgreSQL bazi podataka. Pored klasične metode log-shipping replikacije, biće obrađena i streaming replikacija, kaskadna replikacija, sinhrona replikacija i hot standby. Svaka od ovih metoda ima svoje specifičnosti i prednosti, ali im je zajedničko obezbeđivanje visoke dostupnosti i otpornost na kvarove.

Takođe, biće obrađeni i tehnički aspekti replikacije kao što su autentikacija, monitoring, slotovi za replikaciju i kontinuirano arhiviranje na standby serveru. Ovi elementi su ključni za uspešno implementiranje i upravljanje replikacijom u PostgreSQL okruženju.

Kroz ovaj rad, ideja je približiti način rada replikacije u PostgreSQL bazi podataka i prikazati načine za implementaciju i upravljanje ovim konceptom. Za demonstraciju primera je korišćen PostgreSQL server verzije 15.2[1].

# log-shipping Replikacija

Kontinuirano arhiviranje može se koristiti za stvaranje konfiguracije klastera visoke dostupnosti sa jednim ili više standby servera koji su spremni da preuzmu operacije ako primarni server zakaže. Ova mogućnost se često naziva *topli(warm) standby* ili *log šipovanje*.

Primarni i standby server zajedno rade kako bi pružili ovu mogućnost, iako su serveri slabo povezani. Primarni server radi u režimu kontinuiranog arhiviranja, dok svaki standby server radi u režimu kontinuirane obnove, čitajući WAL (Write-Ahead Log) datoteke sa primarnog servera. Nije potrebno vršiti nikakve promene na tabelama baze podataka da bi se omogućila ova mogućnost, pa pruža nisku administrativnu cenu nadogradnju u poređenju sa nekim drugim rešenjima za replikaciju. Ova konfiguracija takođe ima relativno nizak uticaj na performanse primarnog servera.

Direktno prenošenje zapisa iz WAL-a sa jednog serverskog sistema na drugi se obično naziva log šipovanje. PostgreSQL implementira log šipovanje zasnovano na datotekama tako što prenosi zapise iz WAL-a, jednu datoteku (segment) odjednom. WAL datoteke (16MB) se mogu lako i jeftino prenositi preko bilo koje distance, bilo da se radi o susednom sistemu, drugom sistemu na istoj lokaciji ili sistemu na drugom kraju sveta. Propusna moć potrebna za ovu tehniku varira u zavisnosti od količine transakcija na primarnom serveru. Log šipovanje ima veliku granularnost i inkrementalno prenosi promene iz WAL-a preko mrežne veze.

Važno je napomenuti da je log šipovanje asinhrono, tj. zapisi iz WAL-a se šalju nakon potvrde transakcije. Kao rezultat toga, postoji prozor za gubitak podataka ukoliko primarni server doživi katastrofalni kvar. U tom slučaju transakcije koje još nisu poslate biće izgubljene. Veličina prozora gubitka podataka u log šipovanju zasnovanom na datotekama može se ograničiti korišćenjem parametra *archive\_timeout*, koji može biti podešen na nekoliko sekundi. Međutim, tako nisko podešavanje će značajno povećati propusnu moć potrebnu za šipovanje datoteka.

Performanse obnove su dovoljno dobre da će standby serveru biti potrebno par trenutaka od potpune dostupnosti nakon što bude aktiviran. Kao rezultat toga, ovo se naziva konfiguracija *topli(warm) standby* koja pruža visoku dostupnost. Obnavljanje(*restoring*) servera sa arhiviranom baznom sigurnosnom kopijom i primenom promena će trajati znatno duže, tako da ta tehnika nudi rešenje samo za obnovu nakon katastrofe, a ne visoku dostupnost. Standby server takođe se može koristiti za read-only upite, u tom slučaju naziva se *vrući(hot) standby* server.

## Pripremanje primarnog servera za standby servere

Potrebno je podesite kontinuirano arhiviranje na primarnom serveru ka direktorijumu arhive koji je dostupan sa standby servera. Da bi se omogućilo arhiviranje WAL zapisa, potrebno je postaviti:

* konfiguracioni parametar *wal\_level* na *replica* ili viši nivo
* *archive\_mode* na *on*
* odrediti shell komandu za korišćenje u *archive\_command* konfiguracionom parametru ili je potrebno navesti biblioteku koja će se koristiti u *archive\_library* konfiguracionom parametru. U praksi, ova podešavanja se smeštaju u *postgresql.conf* fajl.

U archive\_command, %p se zamenjuje putanjom do fajla koji se arhivira, dok se %f zamenjuje samo imenom fajla. Primer komanda izgleda ovako:

*# Unix*

*archive\_command = 'test ! -f /mnt/server/archivedir/%f && cp %p /mnt/server/archivedir/%f'*

*# Windows*

*archive\_command = 'copy "%p" "C:\server\archivedir\%f"'*

*Ova komanda će kopirati arhivirane segmente WAL zapisa u direktorijum /mnt/server/archivedir. Ako se zamene %p i %f stvarnim vrednostima, stvarna izvršena komanda izgleda ovako:*

*test ! -f /mnt/server/archivedir/00000001000000A900000065 && cp pg\_wal/00000001000000A900000065 /mnt/server/archivedir/00000001000000A900000065*

Slična komanda će biti generisana za svaku novu datoteku koju treba arhivirati.

Arhiva treba da bude dostupna sa standby servera čak i kada je primarni server nedostupan, odnosno treba da se nalazi na samom standby serveru ili na drugom pouzdanom serveru, a ne na primarnom serveru. U slučaju korišćenja streaming replikacije, postaviti autentifikaciju na primarnom serveru kako bi dozvolili replikacione konekcije sa standby serverom. To podrazumeva kreiranje uloge i pružanje odgovarajućeg unosa ili unosa u *pg\_hba.conf* fajlu sa poljem "*database*" postavljenim na "*replication*". Takođe uveriti se da je parametar *max\_wal\_senders* postavljen na dovoljno veliku vrednost u konfiguracionom fajlu primarnog servera. Ako će se koristiti replikacioni slotovi, takođe se uveriti da je parametar *max\_replication\_slots* postavljen na dovoljno visoku vrednost. Napraviti baznu rezervnu kopiju da bi inicijalizovali standby server.

## Postavljanje standby servera

Ukoliko se prilikom pokretanja servera u direktorijumu sa podacima nalazi fajl *standby.signal*, server će ući u režim pripravnosti (standby mode).

U standby režimu, server kontinuirano primenjuje WAL zapise koje prima od primarnog servera. Standby server može čitati WAL zapise iz arhive (*restore\_command*) ili direktno sa primarnog servera putem TCP veze (streaming replikacija). Standby server će takođe pokušati da obnovi sve WAL zapise koji se nalaze u direktorijumu *pg\_wal* u standby klasteru. To se obično dešava nakon restartovanja servera, kada standby ponovo reprodukuje WAL zapise koji su strimovani sa primarnog servera pre restarta, ali takođe je moguće ručno kopirati fajlove u *pg\_wal* u bilo kom trenutku kako bi se reprodukovali.

Prilikom pokretanja, standby server započinje obnavljanje svih dostupnih WAL zapisa u lokaciji arhive, pozivajući *restore\_command*. Kada dostigne kraj dostupnih WAL zapisa u arhivi i *restore\_command* ne uspe, pokušava da obnovi bilo koje WAL zapise koji se nalaze u direktorijumu *pg\_wal*. Ako i to ne uspe, a streaming replikacija je konfigurisana, standby pokušava da se poveže sa primarnim serverom i započne strimovanje WAL zapisa od poslednjeg validnog zapisa pronađenog u arhivi ili pg\_wal. Ako ni to ne uspe ili streaming replikacija nije konfigurisana, ili ako veza kasnije bude prekinuta, standby se vraća na korak 1 i pokušava ponovo da obnovi fajl iz arhive. Ovaj ciklus ponovnih pokušaja iz arhive, *pg\_wal* i putem streaming replikacije se nastavlja sve dok se server ne zaustavi ili ne bude aktiviran failover putem trigger fajla.

Režim standby se završava i server prelazi u normalan rad kada se izvrši *pg\_ctl promote,* pozove *pg\_promote()* ili se pronađe trigger fajl (*promote\_trigger\_file*). Pre izvršavanja failover-a, svi WAL zapisi koji su odmah dostupni u arhivi ili u *pg\_wal* će biti obnovljeni, ali neće se pokušati uspostaviti veza sa primarnim serverom.

Da bi se postavio standby server, potrebno je odraditi restore bekapa preuzetog sa primarnog servera. Kreirajti fajl *standby.signal* u direktorijumu podataka klastera standby servera. Postaviti *restore\_command* na jednostavnu komandu za kopiranje fajlova iz WAL arhive. Ako je planirano da postoje više standby servera radi visoke dostupnosti, treba se pobrinuti da je *recovery\_target\_timeline* postavljen na *latest* (podrazumevano), kako bi standby server pratio promene hronološki koje se javljaju prilikom prelaska na drugi standby server.

Ako se koristi WAL arhiviranje, veličina se može minimizirati koristeći parametar *archive\_cleanup\_command* za uklanjanje fajlova koji više nisu potrebni standby serveru. *pg\_archivecleanup* alat je posebno dizajniran za upotrebu sa *archive\_cleanup\_command* u tipičnim konfiguracijama sa jednim standby serverom. Međutim, treba imati na umu da ako se koristi arhiva u svrhu bekapa, treba zadržati fajlove potrebne za oporavak barem od najnovijeg baznog bekapa, čak i ako više nisu potrebni standby serveru.

Jednostavan primer konfiguracije je:

*primary\_conninfo = 'host=192.168.1.50 port=5432 user=foo password=foopass options=''-c*

*wal\_sender\_timeout=5000'''*

*restore\_command = 'cp /path/to/archive/%f %p'*

*archive\_cleanup\_command = 'pg\_archivecleanup /path/to/archive %r'*

# Streaming Replikacija

Streaming replikacija omogućava rezervnom serveru da bude ažurniji nego što je moguće kod replikacije bazirane na slanju log fajlova. Rezervni server se povezuje sa primarnim serverom, koji prenosi zapise o transakcijama (WAL zapisi) rezervnom serveru kako se generišu, bez čekanja da se popuni WAL fajl.

Streaming replikacija je podrazumevano asinhrona, što znači da postoji malo vremensko kašnjenje između potvrde transakcije na primarnom serveru i promena koje postaju vidljive na rezervnom serveru. Međutim, to kašnjenje je mnogo manje nego kod replikacije bazirane na slanju log fajlova i obično iznosi manje od jedne sekunde, pod uslovom da je rezervni server dovoljno moćan da se nosi sa opterećenjem. Sa streaming replikacijom, nije potrebno postavljati *archive\_timeout* kako bi se smanjio prozor gubitka podataka.

Ako se koristi streaming replikacija bez stalnog arhiviranja log fajlova, server može reciklirati stare segmente WAL zapisa pre nego što ih rezervni server primi. U tom slučaju, rezervni server će morati da bude ponovo inicijalizovan na osnovu novog početnog bekapa. Ovo se može izbeći postavljanjem vrednosti wal\_keep\_size na dovoljno visoku vrednost kako bi se osiguralo da se segmenti WAL zapisa ne recikliraju prerano, ili konfigurisanjem replikacionog slota za rezervni server. Ako se postavi arhiva WAL zapisa koja je dostupna sa rezervnog servera, ova rešenja nisu potrebna, jer rezervni server uvek može koristiti arhivu kako bi došao do trenutnog stanja, pod uslovom da zadržava dovoljan broj segmenata.

Da bi se koristila streaming replikacija, rezervni server mora biti postavljen za slanje log fajlova na bazi datoteka. Korak koji pretvara rezervni server za slanje log fajlova u rezervni server za streaming replikaciju je podešavanje *primary\_conninfo* da pokazuje ka primarnom serveru. Treba postaviti *listen\_addresses* i opcije za autentifikaciju (*pg\_hba.conf*) na primarnom serveru tako da se rezervni server može povezati sa pseudo-bazom replikacije na primarnom serveru.

Na sistemima koji podržavaju opciju soketa *keepalive*, postavljanje vrednosti *tcp\_keepalives\_idle*, *tcp\_keepalives\_interval* i *tcp\_keepalives\_count* pomaže primarnom serveru da brzo primeti prekinutu vezu. Potrebno je postaviti maksimalni broj istovremenih veza sa rezervnim serverima (*max\_wal\_senders*).

Kada se rezervni server pokrene i *primary\_conninfo* je pravilno postavljen, rezervni server će se povezati sa primarnim serverom nakon reprodukcije svih dostupnih WAL fajlova iz arhive. Ako se veza uspostavi uspešno, videće se *walreceiver* na rezervnom serveru i odgovarajući proces *walsender* na primarnom serveru.

## Autentikacija

Veoma je važno da privilegije pristupa za replikaciju budu postavljene tako da samo pouzdani korisnici mogu čitati stream WAL-a, jer je lako izvući bitne informacije iz njega. Rezervni serveri moraju da se autentifikuju na primarnom serveru kao korisnik koji ima privilegiju *REPLICATION* ili kao *superkorisnik*.

Preporučuje se kreiranje posebnog korisničkog naloga sa privilegijom *REPLICATION* i *LOGIN* za replikaciju. Iako privilegija *REPLICATION* omogućuje veoma visok nivo dozvola, ne dozvoljava korisniku da menja podatke na primarnom sistemu, za razliku od *SUPERUSER* privilegije.

Autentifikacija klijenta za replikaciju se kontroliše putem naredbi u *pg\_hba.conf* datoteci koji specificira replikaciju u polju baze podataka. Na primer, ako se rezervni server pokreće na hostu sa IP adresom 192.168.1.100 i ime naloga za replikaciju je foo, administrator može dodati sledeću liniju u pg\_hba.conf datoteku na primarnom serveru:

*# Dozvoli korisniku "foo" sa hostom 192.168.1.100 da se poveže na primarni*

*# server kao rezervni server za replikaciju ako je korisnikova lozinka ispravno uneta.*

*#*

*# TYPE DATABASE USER ADDRESS METHOD*

*host replication foo 192.168.1.100/32 md5*

Ime hosta i broj porta primarnog servera, korisničko ime za povezivanje i lozinka se navode u *primary\_conninfo*. Lozinka se takođe može postaviti u ~/.pgpass datoteci na rezervnom serveru (navodeći replikaciju u polju baze podataka). Na primer, ako se primarni server pokreće na hostu sa IP adresom 192.168.1.50, port 5432, ime naloga za replikaciju je foo, a lozinka je foopass, administrator može dodati sledeću liniju u postgresql.conf datoteku na rezervnom serveru:

*# Rezervni server se povezuje na primarni koji je pokrenut na adresi 192.168.1.50*

*# i portu 5432 kao korisnik „foo“ čija je lozinka „foopass“*

*primary\_conninfo = 'host=192.168.1.50 port=5432 user=foo password=foopass'*

## Monitoring

Važan pokazatelj stanja streaming replikacije je količina generisanih WAL zapisa na primarnom serveru, ali koji još nisu primenjeni na rezervnom serveru. Ovo kašnjenje se može izračunati upoređivanjem trenutne lokacije upisa WAL-a na primarnom serveru sa poslednjom primljenom lokacijom WAL-a na rezervnom serveru. Ove lokacije se mogu dobiti pomoću funkcije *pg\_current\_wal\_lsn* na primarnom serveru *i pg\_last\_wal\_receive\_lsn* na rezervnom serveru. Poslednja primljena lokacija WAL-a na rezervnom serveru takođe se prikazuje u statusu WAL procesa za prijem, koji se može dobiti pomoću naredbe "ps".

Može se dobiti lista procesa WAL slanja putem pregleda *pg\_stat\_replication*. Velike razlike između *pg\_current\_wal\_lsn* i polja *sent\_lsn* u pregledu mogu ukazivati da je primarni server pod velikim opterećenjem, dok razlike između *sent\_lsn* i *pg\_last\_wal\_receive\_lsn* na rezervnom serveru mogu ukazivati na kašnjenje u mreži ili da je rezervni server pod velikim opterećenjem.

Na aktivnom rezervnom serveru, status WAL procesa za prijem može se dobiti putem pregleda *pg\_stat\_wal\_receiver*. Velika razlika između *pg\_last\_wal\_replay\_lsn* i polja *flushed\_lsn* u pregledu ukazuje da se WAL zapisi primaju brže nego što se mogu reprodukovati.

## Slotovi za replikaciju

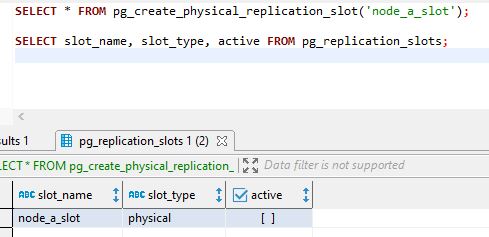
Replikacijski slotovi pružaju automatizovan način da se osigura da primarni server ne uklanja WAL segmente dok ih nisu primili svi rezervni serveri i da primarni server ne uklanja redove koji bi mogli izazvati sukob u oporavku čak i kada je rezervni server isključen.

Umesto korišćenja replikacijskih slotova, moguće je sprečiti uklanjanje starih WAL segmenata pomoću *wal\_keep\_size*, ili čuvanjem segmenata u arhivi korišćenjem *archive\_command* ili *archive\_library*. Međutim, ovi metodi često rezultuju zadržavanjem više WAL segmenata nego što je potrebno, dok replikacijski slotovi zadržavaju samo broj segmenata koji su potrebni. S druge strane, replikacijski slotovi mogu zadržati toliko mnogo WAL segmenata da popune prostor alociran za *pg\_wal*. *max\_slot\_wal\_keep\_size* ograničava veličinu zadržanih WAL fajlova od strane replikacijskih slotova.

Slično tome, *hot\_standby\_feedback* i *vacuum\_defer\_cleanup\_age* pružaju zaštitu od uklanjanja relevantnih redova od strane vakuuma, ali prvi ne pruža zaštitu tokom perioda kada rezervni server nije povezan, a drugi često mora biti postavljen na visoku vrednost kako bi pružio adekvatnu zaštitu. Replikacijski slotovi prevazilaze ove nedostatke.

Svaki replikacijski slot ima ime koje može sadržavati mala slova, brojeve i donju crtu. Postojeći replikacijski slotovi i njihovo stanje mogu se videti u pregledu *pg\_replication\_slots*. Slotovi se mogu kreirati i obrisati putem protokola za streaming replikaciju ili putem SQL funkcija.

Slot za replikaciju se može kreirati na sledeći način:



Slika 1. Primer kreiranja slota za replikaciju

Da bi se konfigurisao standby da koristi ovaj slot, treba konfigurisati *primary\_slot\_name* na standby-u.

*Primer:*

*primary\_conninfo = 'host=192.168.1.50 port=5432 user=foo password=foopass' primary\_slot\_name = 'node\_a\_slot'*

# Kaskadna replikacija

Mogućnost kaskadne replikacije omogućava standby serveru da prihvati konekcije za replikaciju i da strimuje WAL zapise drugim standby serverima, ponašajući se kao relej. Ovo se može koristiti kako bi se smanjio broj direktnih konekcija sa primarnim serverom i smanjila na minimum opterećenja mrežnog saobraćaja između lokacija.

Standby server koji istovremeno deluje kao primalac i pošiljalac naziva se kaskadni standby. Standby serveri koji su direktnije povezani sa primarnim serverom nazivaju se *upstream* serveri, dok su standby serveri koji su udaljeniji *downstream* serveri. Kaskadna replikacija ne postavlja ograničenja na broj ili raspored *downstream* servera, i svaki standby server se povezuje samo sa jednim *upstream* serverom koji na kraju vodi ka jednom primarnom serveru.

Kaskadni standby šalje ne samo WAL zapise koje je primio od primarnog servera, već i one koji su obnovljeni iz arhive. Tako da čak i ako se replikaciona konekcija na nekom *upstream* serveru prekine, streaming replikacija se nastavlja *downstream* sve dok postoje novi WAL zapisi.

Trenutno je kaskadna replikacija asinhrona i postavke sinhronizovane replikacije nemaju uticaj na kaskadnu replikaciju.

Informacije o stanju standby servera se propagiraju na više, bez obzira na kaskadni raspored. Ako se *upstream* standby server promoviše da postane novi primarni server, *downstream* serveri će nastaviti da strimuju sa novog primarnog servera ukoliko je *recovery\_target\_timeline* postavljen na *latest* (podrazumevana vrednost).

Da bi se koristila kaskadna replikacija, treba konfigurisati kaskadni standby da prihvati konekcije za replikaciju (postaviti *max\_wal\_senders* i *hot\_standby*, i konfigurisati autentifikaciju na osnovu hosta). Takođe je potrebno da se konfiguriše *primary\_conninfo* na *downstream* standby serveru tako da pokazuje na kaskadni standby server.

# Sinhrona replikacija

PostgreSQL *streaming* replikacija je podrazumevano asinhrona. Ako primarni server doživi pad, neke transakcije koje su bile potvrđene možda nisu replikovane na standby server, što može dovesti do gubitka podataka. Količina gubitka podataka je proporcionalna kašnjenju replikacije u trenutku prelaska na standby server.

Sinhrona replikacija pruža mogućnost potvrde da su sve promene izvršene u okviru transakcije prenete na jedan ili više sinhronih standby servera. Ovo proširuje standardni nivo trajnosti koji se dobija potvrdom transakcije. Ovaj nivo zaštite se naziva *2-safe* replikacija u teoriji računarstva, a *group-1-safe* (group-safe i 1-safe) kada *je synchronous\_commit* podešen na *remote\_write*.

Kada se zahteva sinhrona replikacija, svaka potvrda transakcije sa upisom će čekati da se primi potvrda da je potvrda zabeležena u zapisniku upisa na disku kako primarnog tako i standby servera. Jedina mogućnost gubitka podataka je ako i primarni i standby server dožive pad istovremeno. Ovo može pružiti mnogo veći nivo trajnosti, ali samo ako sistem administrator bude pažljiv u postavljanju i upravljanju ova dva servera. Čekanje na potvrdu povećava sigurnost korisnika da promene neće biti izgubljene u slučaju pada servera, ali takođe povećava vreme odziva za zahtevanu transakciju. Minimalno vreme čekanja je vreme kašnjenja između primarnog i standby servera.

Read-only transakcije i rollback transakcija ne moraju čekati odgovore od standby servera. Potvrde podtransakcija ne čekaju odgovore od standby servera, već samo potvrde najviših nivoa. Dugotrajni procesi kao što su učitavanje podataka ili kreiranje indeksa ne čekaju do poslednje potvrde transakcije.

Sinhroni standby može biti fizički standby ili logički pretplatnik replikacije. Takođe može biti bilo koji drugi potrošač strimova fizičke ili logičke WAL replikacije koji zna kako da šalje odgovarajuće povratne informacije. Osim ugrađenih sistema za fizičku i logičku replikaciju, ovo uključuje posebne programe kao što su *pg\_receivewal* i *pg\_recvlogical*, kao i neke *3rd party* sisteme za replikaciju i custom rešenja.

Nakon što je strimovna replikacija konfigurisana, konfiguracija sinhronne replikacije zahteva samo još jedan dodatni korak: postavljanje vrednosti *synchronous\_standby\_names* na određenu vrednost. *synchronous\_commit* takođe mora biti podešen na *on*, ali budući da je ovo podrazumevana vrednost, obično nije potrebna promena. Ova konfiguracija će uzrokovati da svaki komit čeka potvrdu da je standby server zabeležio potvrdu transakcije na trajnom skladištu. *synchronous\_commit* može biti podešen od strane pojedinačnih korisnika, pa se može konfigurisati u konfiguracionom fajlu, za određene korisnike ili baze podataka, ili dinamički od strane aplikacija, kako bi se kontrolisala garancija trajnosti na nivou pojedinačne transakcije.

Nakon što je potvrda transakcije zabeležena na disku na primarnom serveru, WAL zapis se šalje na standby server. Standby server šalje poruke svaki put kada se nova grupa WAL podataka zabeleži na disku, osim ako je *wal\_receiver\_status\_interval* postavljen na nulu na standby serveru. U slučaju kada je *synchronous\_commit* podešen na *remote\_apply*, standby server šalje odgovore kada se potvrda transakcije reprodukuje, čime se transakcija čini vidljivom. Ako je standby server odabran kao sinhroni standby server prema postavci *synchronous\_standby\_names* na primarnom serveru, odgovori od tog standby servera će se uzeti u obzir zajedno sa odgovorima od drugih sinhronih standby servera kako bi se odlučilo kada osloboditi transakcije koje čekaju na potvrdu da je zapis o potvrdi primljen. Ovi parametri omogućavaju administratoru da odredi koji standby serveri trebaju biti sinhroni standby serveri.

Treba napomenuti da se konfiguracija sinhronne replikacije uglavnom vrši na primarnom serveru. Imenovani standby serveri moraju biti direktno povezani sa primarnim serverom. Primarni server ne zna ništa o standby serverima koji koriste kaskadnu replikaciju.

Podešavanje *synchronous\_commit* na *remote\_write* će uzrokovati da svaka potvrda čeka potvrdu da je standby server primio zapis o potvrdi i zabeležio ga na svom operativnom sistemu, ali ne i da se podaci isprazne na disk na standby serveru. Ovo podešavanje pruža slabiju garanciju trajnosti u odnosu na podešavanje *on*: standby server može izgubiti podatke u slučaju pada operativnog sistema, iako ne i pada PostgreSQL-a. Međutim, ovo je korisno podešavanje u praksi jer može smanjiti vreme odziva transakcije. Gubitak podataka može se dogoditi samo ako i primarni i standby server dožive pad i baza podataka primarnog servera istovremeno bude oštećena.

Podešavanje *synchronous\_commit* na *remote\_apply* će uzrokovati da svaka potvrda čeka dok trenutni sinhroni standby serveri ne prijave da su reprodukovali transakciju, čime je ona vidljiva za upite korisnika. U jednostavnim slučajevima, ovo omogućava ravnotežu opterećenja uz kauzalnu doslednost.

Korisnici će prestati čekati ako se zatraži brzo gašenje. Međutim, kao i kod asinhronne replikacije, server se neće potpuno isključiti dok se svi izuzetni WAL zapisi ne prenesu na trenutno povezane standby servere.

## Više sinhronih standby servera

Sinhrona replikacija podržava jedan ili više sinhronih standby servera. Transakcije će čekati dok svi sinhroni standby serveri koji su označeni kao sinhroni ne potvrde prijem svojih podataka. Broj sinhronih standby servera od kojih transakcije moraju čekati odgovor je naveden u parametru *synchronous\_standby\_names*. Ovaj parametar takođe navodi listu imena standby servera i metodu (*FIRST* i *ANY*) za odabir sinhronih standby servera iz liste.

Metoda *FIRST* određuje sinhronu replikaciju na osnovu prioriteta i čini da se transakcije čekaju dok se njihovi WAL zapisi ne replikiraju na traženi broj sinhronih standby servera koji su odabrani na osnovu prioriteta. Standby serveri čija se imena nalaze ranije na listi imaju veći prioritet i smatraju se sinhronim. Ostali standby serveri koji se pojavljuju kasnije na ovoj listi predstavljaju potencijalne sinhronizovane standby servere. Ako se bilo koji od trenutnih sinhronih standby servera isključi iz bilo kojeg razloga, odmah će biti zamenjen sledećim standby serverom najvišeg prioriteta.

Primer za *synchronous\_standby\_names* za više sinhronih standby servera na osnovu prioriteta je:

*synchronous\_standby\_names = 'FIRST 2 (s1, s2, s3)'*

U ovom primeru, ako su pokrenuta četiri standby servera s1, s2, s3 i s4, dva standby servera s1 i s2 će biti odabrana kao sinhroni standby serveri jer se njihova imena pojavljuju na početku liste imena standby servera. s3 je potencijalni sinhroni standby server i preuzeće ulogu sinhronog standby servera kada bilo koji od s1 ili s2 ne uspe. s4 je asinhroni standby server jer njegovo ime se ne nalazi na listi.

Metoda ANY određuje sinhronu replikaciju zasnovanu na kvorumu i čini da se transakcije čekaju dok se njihovi WAL zapisi ne repliciraju na barem traženi broj sinhronih standby servera sa liste.

Primer za *synchronous\_standby\_names* za sinhronu replikaciju zasnovanu na kvorumu sa više sinhronih standby servera je:

*synchronous\_standby\_names = 'ANY 2 (s1, s2, s3)'*

U ovom primeru, ako su pokrenuta četiri standby servera s1, s2, s3 i s4, transakcije će čekati odgovore od barem bilo koja dva standby servera od s1, s2 i s3. s4 je asinhroni standby server jer njegovo ime se ne nalazi na listi.

Stanja sinhronih standby servera mogu se videti korišćenjem pregleda pg\_stat\_replication.

## Planiranje performansi

Sinhrona replikacija obično zahteva pažljivo planirane i postavljene standby servere kako bi se osigurale zadovoljavajuće performanse aplikacija. Čekanje ne koristi sistemski resurse, ali zaključavanje transakcija se i dalje zadržava sve dok se prenos ne potvrdi. Kao rezultat, neoprezna upotreba sinhrone replikacije će smanjiti performanse aplikacije baze podataka zbog povećanog vremena odziva i veće konkurencije.

PostgreSQL omogućava programeru aplikacija da odredi nivo trajnosti koji je potreban putem replikacije. Ovo se može odrediti za sistem u celini, ali se može odrediti i za određene korisnike ili konekcije, pa čak i pojedinačne transakcije.

Na primer, opterećenje aplikacije može se sastojati od: 10% promena su važni detalji o kupcima, dok je 90% promena manje važnih podataka koje poslovanje može preživeti ako se izgube, kao što su razmena poruka između korisnika.

Sa postavljenim opcijama sinhronizovane replikacije na nivou aplikacije (na primarnom serveru), možemo ponuditi sinhronu replikaciju za najvažnije promene, bez usporavanja većeg dela ukupnog opterećenja. Opcije na nivou aplikacije su važan i praktičan alat koji omogućava prednosti sinhrone replikacije za visoko performansne aplikacije.

Treba imati na umu da protok mreže mora biti veći od brzine generisanja podataka za pisanje u log datoteku (WAL).

## Planiranje visoke dostupnosti

*synchronous\_standby\_names* određuje broj i imena sinhronih standby servera na koje će transakcije čekati odgovor kada je *synchronous\_commit* postavljen na *on*, *remote\_apply* ili *remote\_write*. Takve transakcije možda neće biti završene ukoliko jedan od sinhronih standby servera padne.

Najbolje rešenje za visoku dostupnost je osigurati da postoje koliko god sinhronih standby servera je potrebno. To se može postići navođenjem više potencijalnih sinhronih standby servera koristeći *synchronous\_standby\_names*.

U sinhronoj replikaciji sa prioritetima, standby serveri čija se imena nalaze na početku liste će se koristiti kao sinhroni standby serveri. Standby serveri koji su navedeni nakon njih će preuzeti ulogu sinhronog standby servera ukoliko neki od trenutnih padne.

U sinhronoj replikaciji zasnovanoj na kvorumu, svi standby serveri koji se nalaze na listi će biti kandidati za sinhroni standby. Čak i ako neki od njih padne, drugi standby serveri će nastaviti sa obavljanjem uloge kandidata za sinhroni standby.

Kada se standby server prvi put poveže sa primarnim serverom, još uvek nije potpuno sinhronizovan. To se naziva režim sustizanja. Kada kašnjenje između standby servera i primarnog servera prvi put dostigne nulu, prelazi se u režim strimovanja u stvarnom vremenu. Period sustizanja može biti dug nakon što je standby server kreiran. Ako se standby server isključi, period hvatanja će se povećati u skladu sa vremenom koliko je standby server bio isključen. Standby server može postati sinhroni standby samo kada dostigne stanje strimovanja. Ovo stanje se može videti korišćenjem pogleda *pg\_stat\_replication*.

Ako se primarni server restartuje dok komiti čekaju na potvrdu, te transakcije će biti označene kao potpuno izvršene kada se primarna baza podataka oporavi. Nije moguće sa sigurnošću tvrditi da su svi standby serveri primili sve otvorene WAL podatke u trenutku pada primarnog servera. Neke transakcije se možda neće prikazivati kao izvršene na standby serveru, iako su prikazane kao izvršene na primarnom serveru. Garancija koju nudimo je da aplikacija neće dobiti eksplicitnu potvrdu o uspešnom izvršavanju transakcije sve dok se ne zna da su svi sinhroni standby serveri sigurno primili WAL podatke.

Ako se ne može održavati traženi broj sinhronih standby servera, trebalo bi smanjiti broj sinhronih standby servera na koje transakcije moraju čekati odgovore u *synchronous\_standby\_names* (ili ga onemogućiti) i ponovno učitati konfiguracioni fajl na primarnom serveru.

Ako je primarni server izolovan od preostalih standby servera, trebalo bi izvršiti prelazak na najboljeg kandidata od preostalih standby servera.

Ako je potrebno ponovo kreirati standby server dok transakcije čekaju, treba se pobrinuti da se komande *pg\_backup\_start()* i *pg\_backup\_stop()* izvršavaju u sesiji sa *synchronous\_commit = off*, inače će ti zahteve čekati zauvek dok se standby server ne pojavi.

## Kontinuirano arhiviranje na standby serveru

Kada se koristi kontinuirano arhiviranje WAL-a (Write-Ahead Log) na standby serveru, postoje dva različita scenarija. WAL arhiva može biti deljena između primarnog servera i standby servera ili standby server može imati svoju sopstvenu WAL arhivu. Kada standby server ima svoju sopstvenu WAL arhivu, ako se postavi *archive\_mode* na *always* standby server će pozivati arhivnu komandu za svaki primljeni WAL segment, bilo da se radi o obnavljanju iz arhive ili strimovanoj replikaciji. Deljena arhiva se može obrađivati na sličan način, ali *archive\_command* ili *archive\_library* moraju proveriti da li već postoji fajl koji se arhivira i da li već postoji fajl sa identičnim sadržajem. To zahteva pažnju u *archive\_command* ili *archive\_library*, jer treba biti oprezan da ne prepiše postojeći fajl sa različitim sadržajem, ali da vrati uspešan rezultat ako je isti fajl arhiviran dva puta. Takođe, sve to mora biti urađeno bez *race condition*-a, ukoliko se dva servera pokušavaju arhivirati isti fajl istovremeno.

Ako je *archive\_mode* postavljen na *on*, arhiver nije omogućen tokom režima oporavka ili standby režima. Ako se standby server promoviše, počeće sa arhiviranjem nakon promocije, ali neće arhivirati nijedan WAL ili istorijski fajl timelina koji nije generisao sam. Da bi se dobila potpuna serija WAL fajlova u arhivi, mora se pobrinuti da se sav WAL arhivira pre nego što stigne na standby server. To je inherentno tačno *file-based* slanja logova, jer standby server može samo obnoviti fajlove koji se nalaze u arhivi, ali ne i ako je omogućena strimovana replikacija. Kada server nije u režimu oporavka, nema razlike između *on* i *always* režima.

# prelazak na rezervni režim rada

Ako primarni server prestane da radi, tada standby server treba da započne postupak prelaska na rezervni režim rada (failover). Ako standby server prestane da radi, nije potreban postupak prelaska na rezervni režim rada. Ako se standby server može ponovo pokrenuti, čak i neko vreme kasnije, onda se proces oporavka može odmah ponovo pokrenuti, iskoristivši mogućnost ponovnog pokretanja oporavka. Ako se standby server ne može ponovo pokrenuti, tada treba kreirati potpuno novu instancu standby servera.

Ako primarni server prestane da radi, a standby server postane novi primarni server, a zatim se stari primarni server ponovo pokrene, mora postojati mehanizam za obaveštavanje starog primarnog servera da više nije primarni. Ovo se nekad naziva *STONITH* (Shoot The Other Node In The Head), što je neophodno kako bi se izbegle situacije u kojima oba sistema misle da su primarni, što može dovesti do konfuzije i na kraju do gubitka podataka.

Mnogi sistemi za prelazak na rezervni režim rada koriste samo dva servera, primarni i standby server, povezana nekom vrstom *heartbeat* mehanizma koji kontinuirano proverava povezanost između ova dva sistema i ispravnost primarnog servera. Takođe je moguće koristiti i treći sistem (tzv. "svedok" server) kako bi se sprečili određeni slučajevi nepravilnog prelaska na rezervni režim rada, ali dodatna složenost možda neće biti opravdana osim ako nije pažljivo konfigurisana i testirana.

PostgreSQL ne pruža softver koji je potreban za identifikaciju kvara na primarnom serveru i obaveštavanje standby servera. Postoji mnogo takvih alata koji su dobro integrisani sa operativnim sistemskim funkcionalnostima potrebnim za uspešan prelazak na rezervni režim rada, kao što je migracija IP adrese.

Kada se prelazi na standby server, postoji samo jedan server u radu, što se naziva *degenerisano* stanje. Bivši standby server postaje primarni, ali bivši primarni server je nedostupan i može ostati nedostupan. Da bi se vratili u normalan rad, standby server mora biti ponovo kreiran, ili na bivšem primarnom sistemu kada se on pokrene, ili na trećem, možda novom sistemu. Alat *pg\_rewind* može ubrzati ovaj proces na velikim klasterima. Kada je proces završen, primarni i standby server se mogu smatrati da su zamenili uloge. Neki ljudi biraju da koriste treći server kako bi pružili rezervnu kopiju za novi primarni server dok se standby server ne ponovo kreira, iako to očigledno komplikuje konfiguraciju sistema i operativne procese.

Prebacivanje sa primarnog na standby server može biti brzo, ali zahteva vreme za ponovno pripremanje klastera za rezervni režim rada. Redovno prebacivanje sa primarnog na standby server je korisno jer omogućava redovno vreme nedostupnosti za održavanje na svakom sistemu. Takođe služi kao test mehanizma prelaska na rezervni režim rada kako bi bilo sigurno da će stvarno raditi kada je potrebno. Preporučuje se pisanje administrativnih procedura.

Da bi se pokrenuo prelazak na rezervni režim rada za standby server koji koristi log-shipping, treba pokrenuti *pg\_ctl* *promote*, pozvati *pg\_promote()*, ili kreirati *trigger* fajl sa nazivom fajla i putanjom navedenom u *promote\_trigger\_file*. Ako je planiran za korišćenje *pg\_ctl promote* ili se poziva *pg\_promote()* za prelazak na rezervni režim rada, *promote\_trigger\_file* nije potreban. Ako se postavljaju serverske instance koje se koriste samo za izvršavanje *read-only* zahteva sa primarnog servera, a ne u svrhu visoke dostupnosti, nije potreban mehanizam za prelazak na rezervni režim rada.

# Hot standby

*Hot standby* je termin koji se koristi da opiše mogućnost povezivanja sa serverom i izvršavanja samo čitanje upita dok je server u režimu obnove arhive ili standby režimu. Ovo je korisno i za svrhu replikacije i za vraćanje sigurnosne kopije na željeno stanje sa velikom preciznošću. Termin hot standby takođe se odnosi na mogućnost servera da pređe iz režima obnove u normalni rad dok korisnici nastavljaju izvršavati upite i/ili održavaju otvorene veze.

Izvršavanje upita u režimu hot standby slično je normalnom izvršavanju upita, iako postoje neke razlike u upotrebi. Tokom rada u režimu "hot standby" na standby serveru, postoje određena ograničenja. Standby server prihvata samo čitanje podataka, ne može vršiti upisivanje i neće mu biti dodeljen ID transakcije. Ograničenja uključuju nedozvoljene operacije manipulacije podacima (DML) i definisanja podataka (DDL), SELECT upite koji generišu DML komande, LOCK komande sa visokim modovima, dvofaznu potvrdu, ažuriranje sekvenci, LISTEN i NOTIFY komande. Tokom rada u režimu "hot standby", parametar *transaction\_read\_only* je uvek postavljen na true. Nakon završetka rada u režimu "hot standby", moguće je započeti transakcije za čitanje i pisanje.

Ako je u postgresql.conf podešena opcija *hot\_standby* (podrazumevana vrednost) i postoji fajl *standby.signal*, server će raditi u režimu "hot standby". Međutim, potrebno je neko vreme da se omoguće konekcije na *hot standby* jer server ne prihvata konekcije dok ne završi dovoljno oporavk da pruži dosledno stanje nad kojim mogu da se izvrše upiti. Tokom ovog perioda, klijenti koji pokušaju da se povežu će biti odbijeni sa porukom o grešci. Da bi se potvrdilo da se server podigao, može se pokušati povezivanje iz aplikacije u petlji ili se mogu potražiti sledeće poruke u logovima servera:

*LOG: entering standby mode ...*

*Zatim neko vreme kasnije ...*

*LOG: consistent recovery state reached*

*LOG: database system is ready to accept read-only connections*

# Zaključak

Replikacija baze podataka predstavlja jedan od najbitnijih mehanizama za obezbeđivanje visoke dostupnosti, pouzdanosti i otpornosti na kvarove kod PostgreSQL baze. U ovom radu su istražene različite metode replikacije, uključujući log-shipping replikaciju, streaming replikaciju, kaskadnu replikaciju, sinhronu replikaciju i hot standby.

Svaka od ovih metoda ima svoje prednosti i nedostatke, i izbor odgovarajuće metode replikacije zavisi od specifičnih zahteva i ciljeva organizacije. Log-shipping replikacija pruža jednostavno i efikasno rešenje za replikaciju podataka, dok streaming replikacija omogućava brže i kontinuirano ažuriranje standby servera. Kaskadna replikacija omogućava prenos podataka između više nivoa replikacije, dok sinhrona replikacija osigurava potpunu doslednost podataka između primarnog i standby servera. Hot standby pruža mogućnost čitanja sa standby servera, čime se povećava performansa i skalabilnost sistema.

Implementacija replikacije zahteva odgovarajuće konfiguracije, kao što su autentikacija, monitoring, slotovi za replikaciju i kontinuirano arhiviranje na standby serveru. Ovi tehnički aspekti su vrlo bitni za uspešno upravljanje replikacijom i održavanje stabilnog i pouzdanog sistema.

U suštini, replikacija kod PostgreSQL baze podataka pruža bitan alat za postizanje visoke dostupnosti i pouzdanosti sistema. Razumevanje različitih metoda replikacije i pravilna implementacija su od velikog značaja za organizacije koje zahtevaju kontinuitet poslovanja i zaštitu podataka. Kroz pravilno planiranje, konfiguraciju i upravljanje replikacijom, može se obezbediti stabilnost, skalabilnost i pouzdanost PostgreSQL baze podataka, čime se zadovoljavaju zahtevi modernog poslovanja i zaštite podataka.

# Literatura

1. https://github.com/catherinedevlin/opensourceshakespeare [Accessed: 5-Maj-2023].
2. https://www.postgresql.org/docs/current/transaction-iso.html [Accessed: 15-Maj -2023].
3. <https://www.postgresql.org/docs/current/warm-standby.html> [Accessed: 25-Maj -2023]
4. <https://www.postgresql.org/docs/current/hot-standby.html> [Accessed: 3-Jun -2023]