UNIVERZITET U NIŠU ELEKTRONSKI FAKULTET

Milan Stanković

**SEMINARSKI RAD**

**Replikacija u PostgreSQL-u**

Sistemi za upravljanje bazama podataka

|  |  |
| --- | --- |
| Student: | Mentor: |
| Milan Stanković, br. ind. 1276 | Prof. dr Aleksandar Stanimirović |

Niš, 2022. god.

Sadržaj

[1. Uvod 3](#_Toc106919969)

[2. Replikacija u PostgreSQL-u 4](#_Toc106919970)

[2.1. Modeli replikacija kod PostgreSQL 6](#_Toc106919971)

[2.2. Režimi replikacije kod PostgreSQL 7](#_Toc106919975)

[2.2.1. Asinhrona replikacija 7](#_Toc106919976)

[2.2.2. Sinhrona replikacija pisanja 8](#_Toc106919977)

[2.3. Tipovi replikacija kod PostgreSQL-a 9](#_Toc106919978)

[2.4. Write-Ahead Logging (WAL) fajlovi 9](#_Toc106919984)

[3. Log shipping 11](#_Toc106919985)

[4. Streaming replication 12](#_Toc106919986)

[4.1. Kaskadna replikacija 14](#_Toc106919991)

[4.2. Failover kod PostgreSQL striming replikacije 14](#_Toc106919992)

[4.3. Oporavak servera kod PostgreSQL-a korišćenjem streaming replikacije 16](#_Toc106919999)

[5. Logička replikacija kod PostgreSQL-a 17](#_Toc106920000)

[5.1. Publikacija (Publication) 18](#_Toc106920006)

[5.2. Pretplata (Subscription) 18](#_Toc106920007)

[5.3. Upravljanje slotovima za replikaciju (Replication Slot Management) 19](#_Toc106920008)

[6. Zaključak 26](#_Toc106920009)

[7. Literatura 27](#_Toc106920010)

# Uvod

Replikacija podataka kod DBMS predstavlja jedan od ključnih koncepata, kojim se obezbeđuje dostupnost podataka. Glavna ideja iza bilo koje vrste replikacije baze podataka je da predstavlja proces čuvanja podataka na više od jednog čvora (baze), kao i da se izmene u određenom glavnom čvoru preslikaju na ostale čvorove. Prosto se može predstaviti kao kopiranje podataka sa jednog servera na kome se čuvaju podaci, na drugi, tako da svi korisnici mogu da dele iste podatke bez ikakvih nedoslednosti.

Postgres podržava replikaciju i nudi veliki broj mehanizame kojima se može ostvariti replikacija podataka na različitim nivoima (kao što su File System Replication, Logical Replication, SQL-Based Replication Middleware, Log shipping…). Korišćenje replikacije kod PostgreSQL baze podataka može biti korisno ne samo da bi se obezbedilo okruženje otporno na greške već omogućava poboljšanje performanse sistema balansiranjem saobraćaja između pripravnih čvorova.

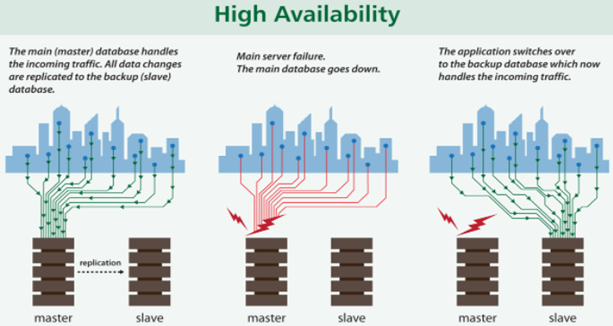
U okviru ovog rada biće detaljnije opisan koncept replikacije kod PostgreSQL-a. Navešćemo neke od osnovnih mehanizama koji se koriste za postizanje replikacije podataka, način na koji funkcioniše replikaciju kod ovih mehanizama kao i njihovi slučajevi korišćenja. Takođe biće prikazan način na koji se obavlja konfiguracija uz konkretan primer korišćenja za određeni mehanizam.

# Replikacija u PostgreSQL-u

Za održavanje visoke dostupnosti baze podataka i balansiranja opterećenja kada veliki broj korisnika pristupa PostgreSQL bazi podataka, potrebno je implementirati različite strategije i može se koristiti više servera baze podataka koji rade zajedno sa serverom korisnika. U ovom slučaju, vrlo je lako obezbediti ovu mogućnost ako su sve operacije samo za čitanje, što znači da se nikakva operacija ne vrši nad podacima prisutnim u serverima baze podataka; preuzimaju se samo dostupni podaci. Ali u slučaju čitanja/pisanja, serveri koji uključuju sve vrste manipulacija podacima baze podataka postaju teški. To je zato što jedna promena u serveru baze podataka treba da se preslika na svim serverima baze podataka sa kojima radi da bi se obezbedila visoka dostupnost krajnjim korisnicima. Proces kopiranja podataka kao i preslikavanje eventualnih izmena sa PostgreSQL servera baze podataka na drugi server naziva se PostgreSQL replikacija. Replikacija se uglavnom implementira u PostgreSQL koristeći konfiguraciju *master-slave*. Izvorni server baze podataka se obično naziva master ili primarni server, dok se server baze podataka koji prima podatke i sve modifikacije napravljene iz primarne baze podataka naziva replika ili standby i ona predstavlja identičnu kopiju master servera. Master je server za čitanje/pisanje, dok je replika uglavnom server samo za čitanje.

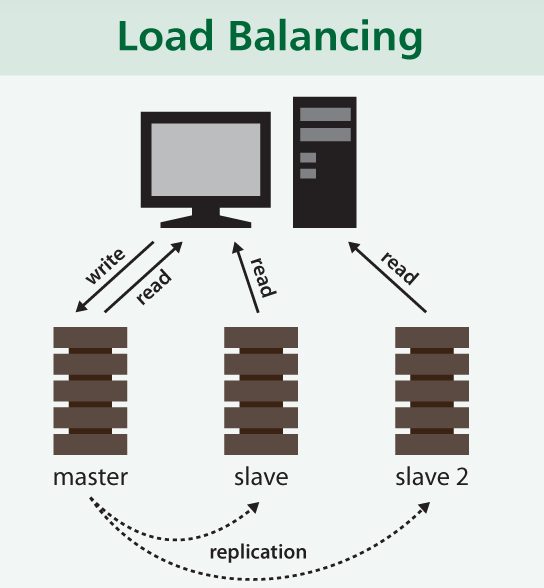
Replikacija PostgreSQL baze podataka ima nekoliko svrha. Kao što smo rekli može pomoći u visokoj dostupnosti i balansiranju opterećenja ali je omogućavanje analitike još jedna sve popularnija svrha replikacije baze podataka. Zamislite da imate Django aplikaciju koja koristi PostgreSQL kao operativnu bazu podataka. Ako želite da omogućite izveštavanje i analitiku na osnovu te baze podataka, najbolje je da je replicirate na drugi server da ne ometa proizvedenu aplikaciju.

Koncept visoka dostupnost (High availability) kod replikacije se odnosi na mogućnost da postoji ažurirana kopija glavne baze podataka u svakom trenutku. U slučaju kvara glavnog servera, uvek postoji mogućnost da se pređe na rezervnu kopiju, za koju se zna da je ažurna. Procedura promovisanja replike u glavnu bazu podataka je uobičajeno poznata kao *failover*, ili procedura prelaska na grešku, i postoji mnogo načina da se to postigne. Odluka o tome koji mehanizam koristiti uvek će zavisiti od korisnikovih specifičnih potreba. Procedura prekomerne greške obično se sastoji od dva osnovna koraka. Prvo, da se ponovo uspostavi fizička veza između klijenta i baze podataka rezervnih kopija, i drugo, da se rezervna baza podataka unapredi u novu glavnu. Specifične procedure za postizanje ovoga zavisiće od konačne metode replikacije koja se koristi. Na slici ispod prikazan je primer High availability-a postignut korišćenjem replikacije.



Slika 1. Postizanje High Availability-a replikacijom

Termin balansiranje opterećenja (Load Balancing) priliom kroišćena replikacija se odnosi na mogućnost distribucije upita opterećenja između glavne baze podataka i replika. Imati jednu glavnu bazu podataka za čitanje i pisanje, sa više replika samo za čitanje, uobičajeno je podešavanje i dobra praksa. Sa ovom konfiguracijom mogu se pokrenuti svi upite koji obavljaju samo čitanje prema replikama umesto prema glavnoj bazi podataka, efektivno smanjujući njeno radno opterećenje. Na slici ispod prikazan je primer Load Balancing-a postignut korišćenjem replikacije.



Slika 2. Postizanje Load Balancing-a replikacijom

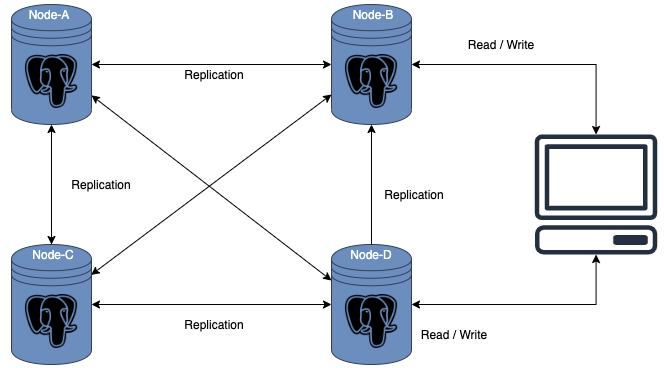
## Modeli replikacija kod PostgreSQL

PosetgreSQL baza podataka podržava dva modela replikacije, a to su:

* Single-Master Replication (SMR)
* Multi-Master Replication (MMR)

Kod Single-Master replikacije, dozvoljava modifikovanje podataka samo na jednom čvoru, a ove modifikacije se repliciraju na jedan ili više čvorova (replika). Ažuriranje i umetanje podataka mogući su samo na glavnom čvoru. U tom slučaju, aplikacije zahtevaju rutiranje saobraćaja do mastera, što dodaje određenu složenost aplikaciji. Zbog pojedinačnih glavnih čvorova nema šanse za sukobe. Većinu vremena jedna glavna replikacija je dovoljna za aplikaciju jer je manje komplikovana za konfigurisanje i upravljanje. Ali u nekim slučajevima, jedna glavna replikacija nije dovoljna i potrebna vam je multi-master replikacija. U slučaju da replika i izvrše neke promene, te promene se ne repliciraju nazad na glavni server.

Multi-master replikacije znače da postoji više od jednog čvora koji deluje kao glavni čvor. Podaci se repliciraju između čvorova, a ažuriranja i umetanje je moguće na grupu glavnih čvorova. U tom slučaju postoji više kopija podataka. Sistem je takođe odgovoran za rešavanje svih sukoba koji se javljaju između istovremenih promena, kako bi se izbegli problemi kao što su duplirani primarni ključevi.



Slika 3. Primer šeme Multi-master replikacije

Prednosti korišćenja multi master replikacija:

* U slučaju da jedan master ne uspe, drugi master je tu da izvrši ažuriranje i umetanje.
* Glavni čvorovi se nalaze na nekoliko različitih lokacija tako da su šanse za kvar svih mastera vrlo minimalne.
* Ažuriranje podataka je moguće na više servera.
* Aplikacija ne zahteva usmeravanje saobraćaja samo do jednog mastera.

Mane prilikom korišćenja multi master replikacija:

* Glavni nedostatak multi-master replikacije je njena složenost.
* Rešavanje sukoba je veoma teško jer je moguće istovremeno upisivanje na više čvorova.
* Ponekad je potrebna ručna intervencija u slučaju sukoba.
* Mogućnost nedoslednosti podataka.

PostgreSQL ima ugrađenu replikaciju sa jednim masterom, ali nažalost, nema višestruke replikacije u glavnom PostgreSQL-u. Dostupna su neka rešenja za multimaster replikaciju, neka od njih su u formi aplikacija, a neka su u sklopu odredjenih podverzija PostgreSQL (PostgreSQL-XL, Rubyrep, Bucardo).



## Režimi replikacije kod PostgreSQL

PostgreSQL podržava dva osnovna moda replikacije, od kojih svaki diktira količinu replikacije podataka koja će se desiti pre nego što klijent smatra da je upisivanje završeno. Ta dva moda su:

* Asinhrona replikacija (Asynchronous replication)
* Sinhrona replikacija pisanja (Synchronous write replication)

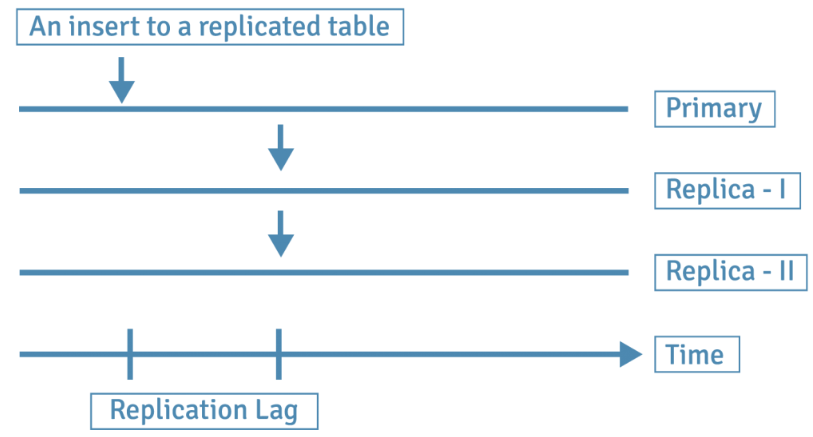
U okviru svakog moda replikacije možete izvršiti dodatna prilagođavanja u vezi sa doslednošću, postojanošću i kada se transakcija izvršava. Ovim ponašanjem upravlja se preko sinchronous\_commit i sinchronous\_standbi\_names podešavanja podešavanja.

### Asinhrona replikacija

U asinhronom režimu, transakcije na glavnom serveru se mogu proglasiti završenim kada su promene izvršene samo na glavnom serveru. Ove promene se kasnije repliciraju na replike. Primarni server upisuje podatke u WAL, odakle se preko WAL sender-a podaci šalju na ostale replike. Serveri replika mogu ostati nesinhronizovani određeno vreme, što se naziva kašnjenje replikacije. Asinhrona replikacija je najučinkovitiji način replikacije za dodavanje podataka. Klijenti moraju samo da sačekaju da primarni server upiše podatke u WAL.

Kako ne postoje garancije da su podaci strimovani u replike nakon uspešnog pisanja, ovaj režim može dovesti do privremene nedoslednosti podataka između mastera i replike.

Gubitak podataka je moguć sa asinhronom replikacijom ako se primarni server nepopravljivo sruši pre nego što je WAL strimovan u replike. U slučaju oporavka primarnog servera, nastaviće se prenos podataka iz WAL-a u replike od mesta gde je prekinut prenos u trenutku prekida.



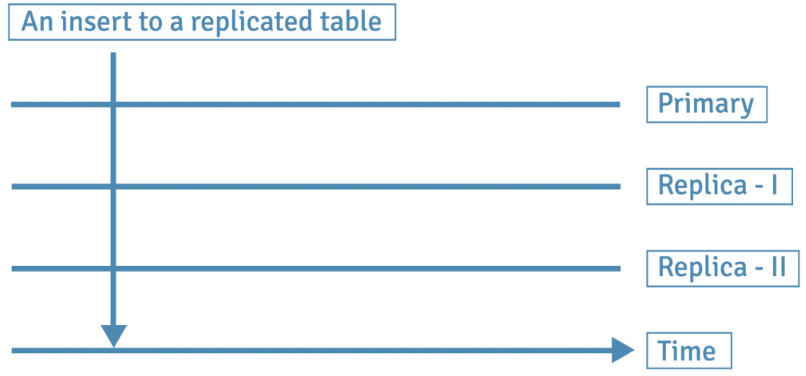
Slika 4. Dodavanja podataka u režimu asinhrone replikacije

### Sinhrona replikacija pisanja

U sinhronom režimu replikacije, transakcije na glavnoj bazi podataka se proglašavaju završenim samo kada su te promene replicirane na sve replike. Sve replika moraju biti dostupni sve vreme da bi se transakcije dovršile na masteru. Performanse sinhronog režima replikacije su znatno sporije u odnosu na asinhroni režim jer je upis gotov tek kada WAL izvrši prenos podataka do svih replikacija.

Konzistentnos podataka je veća u odnosu na ashinroni režim, ali još uvek može doći do nekonzistentnosti i do pojave kašnjenja.

Gubitak podataka je i dalje moguć kod sinhronog režima, ali je mnogo manje verovatan nego kod asinhronog. U većini slučajeva sinhrone replikacije, i primarni server kao i sve replike bi morale da se nepopravljivo sruše. U svakom drugom slučaju, podaci se mogu oporaviti.

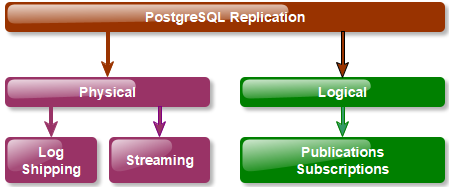


Slika 5. Dodavanje podatka kod sinhronog režima replikacije

## Tipovi replikacija kod PostgreSQL-a

Kod PostgreSQL-a možemo izvršiti osnovnu podelu replikacije na dva tipa, a to su:

* + Fizička replikacija (Physical Replication)
  + Logička replikacija (Logical Replication)



Slika 6. Tipovi replikacije kod PostgreSQL-a

Fizička replikacija se bavi datotekama i direktorijumima. Ne zna šta ovi fajlovi i direktorijumi predstavljaju. Vrši se na nivou sistema datoteka ili na nivou diska. Fizička replikacija šalje promene replici u obliku binarnog formata. Može biti ostvarena pomoću

* File Based Log Shipping – replikacija zasnovana na arhivama
* Streaming Replication - Prenosi WAL fajlove replici (standby-u)

Logička replikacija se, s druge strane, bavi bazama podataka, tabelama i DML operacijama. Stoga je moguće u logičkoj replikaciji replicirati samo određeni skup tabela. Logička replikacija se vrši na nivou klastera baze podataka. Koristi model publisher/subscriber.

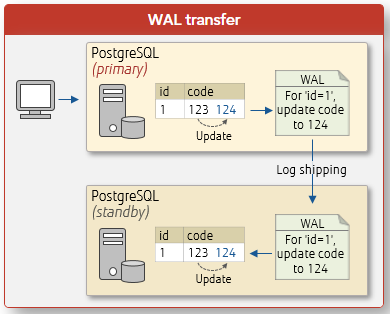
Ovi tipovi replikacije biće detaljnije opisani u nastavku.



## Write-Ahead Logging (WAL) fajlovi

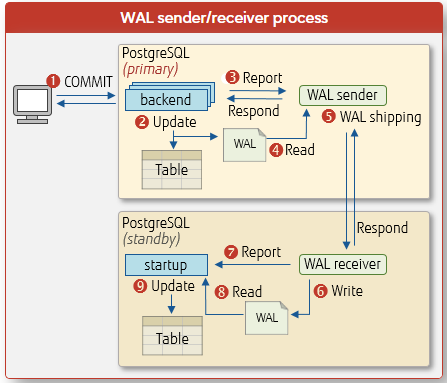
Kako bi razumeli način na koji funkcionišu replikacije, prvo će biti potrebno objasniti način kako se obavljaju transakcije.

U PostgreSQL-u, sve promene izvršene transakcijom se prvo čuvaju u datoteci evidencije (log file), a zatim se rezultat transakcije šalje klijentu koji je inicirao transakciju. Same datoteke sa podacima se ne menjaju pri svakoj transakciji. Ovo je standardni mehanizam za sprečavanje gubitka podataka u slučajevima kao što su pad operativnog sistema, kvar hardvera ili pad PostgreSQL-a. Ovaj mehanizam se zove Write Ahead Logging (WAL), a datoteka evidencije se zove Write Ahead Log. Svaka promena koju transakcija izvrši (INSERT, UPDATE, DELETE, COMMIT) upisuje se u log kao WAL zapis. WAL zapisi se prvo upisuju u WAL bafer u memoriji. Kada se transakcija izvrši, zapisi se upisuju u datoteku WAL segmenta na disku. Ako dođe do pada sistema, baza podataka može da povrati izvršene transakcije iz WAL-a.



Slika 7. Wal transakcije

WAL otpremanje između primarnog servera i standby servera vrši se korišćenjem procesa WAL sender na primarnom serveru do procesa WAL receiver na standby serveru. Ovi procesi se pokreću postavljanjem odgovarajućih parametara u fajlovima postgresql.conf i pg\_hba.conf. WAL receiver, radi na standby serveru, koristi detalje veze date u parametru primary\_conninfo datoteke recoveri.conf i povezuje se sa primarnim serverom pomoću TCP/IP veze. WAL sender zadužen je za slanje WAL zapisa na standby server kako se generišu.



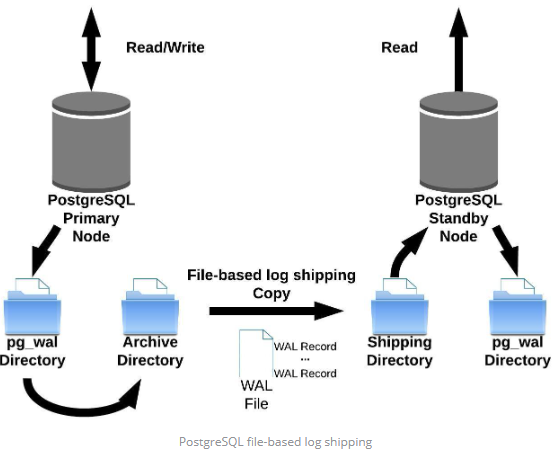
Slika 8. Procesi WAL sender/reciver

# Log shipping

Jedan od načina da se podesi fizička replikacija je da se stalno šalju nove WAL datoteke direktno sa glavnog servera (mastera) baze podataka na drugi server (standby server) i tamo ih primenjivati da bi se dobila sinhronizovana kopiju baze podataka. Ovaj scenario se naziva isporuka evidencije odnosno log shipping.

PostgreSQL implementira isporuku logova zasnovanu na datotekama (file based log shipping) tako što prenosi sve WAL zapise kao jednan fajl (WAL segment). WAL datoteke (16MB) se mogu lako i jeftino slati na bilo koju udaljenost, bilo da se radi o susednom sistemu, drugom sistemu na istoj lokaciji ili drugom sistemu na drugoj strani sveta. Propusni opseg potreban za ovu tehniku varira u zavisnosti od brzine transakcije primarnog servera. Treba napomenuti da je log shipping podrazumevano asinhron, tj. WAL zapisi se šalju nakon izvršavanja transakcije. Kao rezultat toga, postoji prostor za gubitak podataka ako primarni server doživi katastrofalan kvar, transakcije koje još nisu isporučene biće izgubljene. Veličina prostora za gubitak podataka u isporuci evidencije zasnovanoj na fajlovima može biti ograničena korišćenjem parametra archive\_timeout, koji se može podesiti na samo nekoliko sekundi. Međutim, takva podešavanja će značajno povećati propusni opseg potreban za slanje datoteka. Streaming dozvoljava mnogo manji prostor za gubitak podataka.

Log shipping može raditi čak i u slučaju kada master i replika ne mogu komunicirati direktno preko TCP/IP protokola, koristeći deljenu memoriju. Na ovaj način može se nastaviti sa radom čak i u slučaju da replika nije dostupna neko vreme. Master će odbaciti WAL nakon što ga arhivira, ali je WAL idalje arhiviran i može biti korišćen on strane replike, iako ga master ne bi mogao poslati korišćenjem streaminga, jer streaming replikacija zahteva da su i master i replika online.  
Log shipping nudi veoma dobre performanse oporavka. Koristeći warm standby konfiguraciju moguće je obezbediti visoku dostupnost. Log shipping između dva servera koji koriste različita izdanja PostgreSQL-a nije moguć, takođe nije moguć u slučaju korisišćenja dve različite arhitekture sistema ili hardvera.

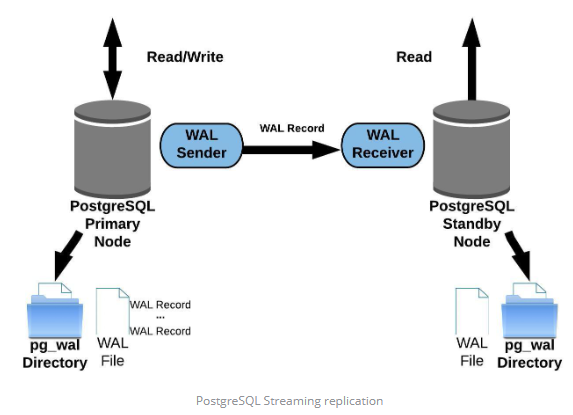


Slika 9. Dijagram Log shipping-a kod PostgreSQL-a

# Streaming replication

Striming replikacija je najčešća korišćena metoda replikacija kod PostgreSQL-a. To je fizička replikacija koja replicira promene na nivou bajt po bajt, stvarajući identičnu kopiju baze podataka na drugom serveru. Zasnovan je na metodi log shipping-a. WAL zapisi se direktno premeštaju sa jednog servera (master) baze podataka na drugi server (standby) da bi se primenili.

Prenos WAL se obavlja na dva različita načina, prenosom WAL zapisa jedne datoteke (WAL segment) u isto vreme (file based log shipping) i drugi način je prenosom WAL zapisa (WAL datoteka se sastoji od WAL zapisa) u hodu (record based log shipping), direktno između primarnog servera i jednog ili više servera u stanju pripravnosti, bez čekanja da se popuni WAL datoteka. Streaming replikacija koristi drugi način. U praksi, proces pod nazivom WAL reciver, koji je pokrenut na standby serveru, povezaće se sa primarnim serverom koristeći TCP/IP vezu. Na primarnom serveru postoji još jedan proces pod nazivom WAL sender, on je zadužen za slanje WAL registara na stanby server.



Slika 10. Dijagram Striming replikacije

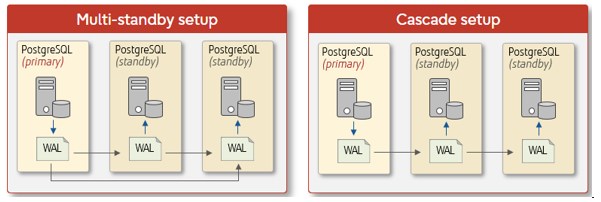
Kada konfigurišete replikaciju striminga, imate opciju da omogućite WAL arhiviranje. Ovo nije obavezno, ali je izuzetno važno za robusno podešavanje replikacije, jer je neophodno izbeći da glavni server reciklira stare WAL datoteke koje još nisu primenjene na standby server. Ako se to dogodi, moraćete da ponovo napravite repliku od nule.

Kad god se desi da striming replikacije ne uspe, vratiće se na prvi korak, odnosno ponovo će vratiti zapise iz arhive. Ova petlja ponovnih pokušaja vraćanja iz arhive, pg\_val, nastaviće se sve dok se server ne zaustavi, ili dok se ne pokrene failover iz trigger fajla.

Striming replikacija je podrazumevano asinhrona, tako da u svakom trenutku možete imati neke transakcije koje se mogu izvršiti na primarnom serveru, a koje još nisu replicirane na standby server. Ovo implicira potencijalni gubitak podataka. Međutim, ovo kašnjenje između izvršenja i uticaja na promene u replici bi trebalo da bude zaista malo (neke milisekunde), pod pretpostavkom da je server replike dovoljno moćan da održi korak sa opterećenjem.

Za slučajeve kada čak i rizik od blagog gubitka podataka nije prihvatljiv, može se koristiti funkcija sinhrone replikacije koja je uvedena u PostgreSQL-u u verziji 9.1. Korišćenjem ove konfiguracije smanjuje se gubitak podataka ali očigledna mana je da se vreme odgovorno za svaku transakciju pisanja povećava.

Striming replikacija se može izgraditi u 1:N konfiguraciji, gde se može konfigurisati samo jedan primarni server, ali se može podesiti više standby servera. Konfiguracija koja povezuje (isporučuje WAL) primarni server sa svim serverima u stanju pripravnosti naziva se multi-standby podešavanje. Takođe pored multi-standby podešavanja moguće je napraviti kaskadno podešavanje gde se standby server povezuje (šalje WAL) sa drugim standby serverom.



Slika 11. Konfiguracija Striming replikacije

Striming replikacija kao što smo već rekli zasnovana je na prosleđivanju i primenjivanju WAL zapisa na standby server. Kao rezultat dobijamo da je standby server kopija primarnog servera bit po bit.

Prednost koja se postiže korišćenjem striming replikacije kod PostgreSQL-a su:

* Rešavanje failover-a (u slučaju kada primarni server otkaže, standby server može preuzeti izvršavanje operacija)
* Balansiranje opterećenja servera samo za čitanje (Obrada operacija čitanja može biti distribuirana među više servera, čime bi se postiglo balansiranje opterećenja servera)

Ograničenja koja se javljaju korišćenjem striming replikacije su:

* Nije moguće vršiti repliciranje na različitim verzijama ili arhitekturama
* Nije moguće vršiti bilo kakve izmene na standby serveru
* Nema previše granularnosti u onome što se replicira

Kako bi se prevazišla ova ograničenja PostgreSQL 10 dodao je podršku za logičku replikaciju.



## Kaskadna replikacija

Funkcija kaskadne replikacije omogućava standby serveru da prihvati veze za replikaciju i strimuje WAL zapise u druge rezervne servere, delujući kao relej. Ovo se može koristiti za smanjenje broja direktnih veza sa primarnim serverom i takođe za smanji troškove propusnog opsega među lokacijama.

Standby server koje deluje i kao primalac i kao pošiljalac je poznato kao kaskadni standby. Standby serveri koji su bliže povezani sa primarnim su poznati kao upstream serveri, dok su oni dalje povezani poznati kao downstream serveri. Kaskadna replikacija ne postavlja ograničenja u broju ili rasporedu downstream servera, ali svaki od njih može biti povezan samo sa jednim upstream serverom koji se na kraju povezuje sa jednim primarnim serverom. Kaskadni standby ne šalje samo WAL zapise primljene iz primarnog servera, već i one vraćene iz arhive. Dakle, čak i ako je veza replikacije u nekoj uzvodnoj vezi prekinuta, replikacija striminga se nastavlja nizvodno sve dok su novi WAL zapisi dostupni.

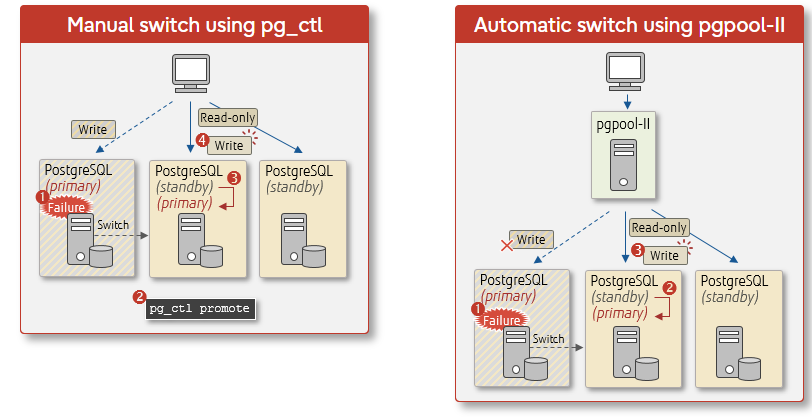
Ako je neki od upstream standby servera promovisan u primarni server, nastavlja se prenos podataka na downstream servere. Kako bi koristili kaskadna replikacija, potrebno je na standby serverima omogućiti prihvatanje veze za replikaciju, treba podesiti parametre max\_wal\_senders, hot\_standby I host-based\_authentication.

Kaskadna replikacija je asinhrona. Sinhrona replikacija nema nikakav efekat na kaskadnu replikaciju.

## Failover kod PostgreSQL striming replikacije

Korišćenjem replikacije strimovanja, kao što smo rekli možemo obezbediti rezervni plan u slučaju da dođe do prekida rada nekog od servera, bilo da je reč o primarnom ili rezervnom serveru. Npr. ukoliko primarni server otkaže sa radom moguće je preći na rezervni (standby) server i nastaviti sa radom nesmetano. Ovaj deo biće fokusira na način za prevazilaženje greške (failover) i oporavak korišćenjem striming replikacije, kako bi se omogućio stabilan i kontinuiran rad sistema.

PostgreSQL ne sadrži funkcije za otkrivanje grešaka i automatsko prebacivanje na drugi server. Potrebno je obezbediti mehanizam za otkrivanje abnormalnosti i skriptu za promovisanje rezervnog servera ('pg\_ctl promote'). Alternativno, može se koristiti softver klastera kao što je pgpool-II, sa svojom detekcijom grešaka i funkcijama automatskog prebacivanja.



Slika 12. Neki od načini rešavanaja Failover-a korišćenjem Striming replikacije

U nastavku biće demonstiran način kako se rešava failover kod PostgreSQL striming replikacije, odnosno simuliraćemo prestanak (otkazivanje) rada master klastera i izvršićemo promovisanje rezervnog klastera u novi master. Potrebno je napomenuti da je u ovom primeru smatrano kao da su prethodno izvršitene odgovarajuće konfiguracije i da je omogućena replikacija između master (u ovom slučaju main) klastera i rezervnog (replica1) klastera.

Da bi simulirali prestanka rada main klastera izvršićemo sledeću naredbu:

sudo systemctl stop postgresql@10-main

Sada kada smo simulirali otkazivanje master čvora, vršimo ručno promovisanje rezervnog replica1 klastera u novi primarni klaster, koristeći naredbu *pg\_ctl promote*:

sudo pg\_ctlcluster 10 replica1 promote

Nakon ovoga klaster replica1 promovisan je u novi master čvor, i može se nastaviti sa nesmetanim radom. Ukoliko postoji mogućnost, stari master čvor (main) se može opet promovisati u master čvor što se naziva failback. To se izvršava tako što će se obaviti sinhornizacija izmenjenih podataka sa novim master čvorom replica1 (tako da će main čvor postati prvo standby čvor), a zatim je moguće izvršiti ponovno promovisanje main čvora da postane novi master.



## Oporavak servera kod PostgreSQL-a korišćenjem streaming replikacije

Radnje oporavka koje treba preduzeti se razlikuju u zavisnosti od toga koji server, primarni ili rezervni (standby), ne radi.

Ukoliko primarni server otkaže sa radom, potrebno je ponovno izgraditi ga kao standby server. Postoje različite metode da se obavi oporavak servera koji je otkazao, ali tipične metode:

* Korišćenjem rezervne kopije baze podataka
* Korišćenjem klaster fajla

Prva metoda koristi rezervnu kopiju baze (pg\_basebackup naredbu). Ovaj metod se realizuje na sledeći način: Prvo se indetifikuje uzrok otkazivanja i preduzimaju se određene akcije kako bi se oporavio server koji je otkazao. Na serveru koji je otkazao uklanjaju se klasteri baze podataka. Izvršava se komanda za vraćanje rezerne kopije pg\_basebackup. Potrebno je modifikovatio recovery.conf ifajl na serveru koji je otkazao i podesiti opciju standby\_mode=on. Nakon toga može se pokrenuti server koji je otkazao koji će se ponašati kao novi rezervni server.

Druga metoda koristi klaster fajlove za oporavak i koristi se samo u slučaju kada server nije otkazao usled problema sa hardverom, tako da možemo resetovati bazu podataka. Oporavak servera koji je otkazao se obavlja tako što se vrši sinhronizacija razlike između tog servera i novog primarnog servera, korišćenjem naredbe pg\_rewind. Zatim se vrši potrebno konfigurisanje otkazanog servera kako bi radio kao rezervni server, I nakon završetka konfigurisanja server je spreman za rad.

# Logička replikacija kod PostgreSQL-a

Logička replikacija je uvedena kod PostgreSQL-a kao dodatna funkcionalnost nakon verzije 10. Ona predstavlja fleksibilnije i lakše sredstvo za repliciranje tabela od redovnog mehanizma replikacije metodom strimovanja. Međutim, logička replikacija sama po sebima ima i neka ograničenja koja mogu ili ne moraju biti razlog za korišćenje neki drugih metoda replikacije.

Logička replikacija je metod replikacije objekata podataka i njihovih promena, na osnovu njihovog identiteta replikacije (obično primarnog ključa). Koristimo termin logička za razliku od fizičke replikacije, koja koristi tačne adrese blokova i replikaciju bajt po bajt. PostgreSQL kao što smo već rekli podržava oba mehanizma istovremeno. Logička replikacija omogućava preciznu kontrolu nad replikacijom podataka i bezbednošću.

Logička replikacija koristi model objavljivanja (publish) i pretplate (subscribe) sa jednim ili više pretplatnika (subscribers) koji se pretplate na jednu ili više publikacija (publications) na čvoru izdavača (publisher node). Pretplatnici preuzimaju podatke iz publikacija na koje su pretplaćeni i mogu naknadno ponovo objaviti podatke kako bi omogućili kaskadno replikaciju ili složenije konfiguracije.

Logička replikacija tabele obično počinje pravljenjem snimka (snapshot) podataka u bazi podataka izdavača i njihovim kopiranjem pretplatniku. Kada se to uradi, promene na izdavaču se šalju pretplatniku kako se dešavaju u realnom vremenu. Pretplatnik primenjuje podatke istim redosledom kao i izdavač, tako da je zagarantovana konzistentnost transakcija za publikacije u okviru jedne pretplate. Ovaj metod replikacije podataka se ponekad naziva transakciona replikacija.

Tipični slučajevi upotrebe za logičku replikaciju su:

* Slanje inkrementalnih promena u jednoj bazi podataka ili podskupu baze podataka pretplatnicima kako do njih dođe.
* Aktiviranje pokretača za pojedinačne promene kako stignu na pretplatnika.
* Konsolidovanje više baza podataka u jednu (na primer za analitičke svrhe).
* Repliciranje između različitih verzija PostgreSQL-a.
* Replikacija između PostgreSQL instanci na različitim platformama (na primer sa Linux-a na Windows)
* Omogućavanje pristupa repliciranim podacima različitim grupama korisnika.
* Deljenje podskupa baze podataka između više baza podataka.

Baza podataka pretplatnika se ponaša na isti način kao i svaka druga PostgreSQL instanca i može se koristiti kao izdavač za druge baze podataka definisanjem sopstvenih publikacija. Kada se pretplatnik (subscriber) tretira samo kao da se obavlja čitanje (read-only) od strane aplikacije, neće doći do sukoba iz jedne pretplate. S druge strane, ako postoje drugi zapisi koje vrši aplikacija ili drugi pretplatnici na isti skup tabela, može doći do sukoba.



## Publikacija (Publication)

Publikacija se može definisati na bilo kojoj primarnoj fizičkoj replikaciji. Čvor u kome je definisana publikacija naziva se izdavač . Publikacija je skup promena generisanih iz tabele ili grupe tabela, a može se opisati i kao skup promena ili skup replikacije. Svaka publikacija postoji samo u jednoj bazi podataka. Takođe svaka publikacija može imati više pretplatnika.

Publikacije se razlikuju od šema i ne utiču na to kako se pristupa tabeli. Svaka tabela se može dodati u više publikacija ako je potrebno. Publikacije trenutno mogu da sadrže samo tabele. Objekti se moraju eksplicitno dodati, osim kada se kreira publikacija za sve tabele (ALL TABLES).

Publikacije mogu izabrati da ograniče promene koje proizvode na bilo koju kombinaciju naredbi INSERT, UPDATE, DELETE, i TRUNCATE, slično načinu na koji se okidači pokreću određenim tipovima događaja. Podrazumevano, svi tipovi operacija se repliciraju.

Podeljena tabela mora imati konfigurisan „ identitet replike “ (replica identity) da bi mogla da replicira UPDATE i DELETE operacije, tako da se odgovarajući redovi za ažuriranje ili brisanje mogu identifikovati na strani pretplatnika. Podrazumevano, identifikator je primarni ključ, ako postoji. Još jedan jedinstveni indeks (sa određenim dodatnim zahtevima) se takođe može podesiti da bude identitet replike. Ako tabela nema odgovarajući ključ, onda se može postaviti na identitet replike „ pun “, što znači da ceo red postaje ključ. Ovo je, međutim, veoma neefikasno i trebalo bi da se koristi samo kao rezerva ako nije moguće drugo rešenje. Ako replika identiteta je postavljen drugačije umesto na „ puna” na strani izdavača, replika identiteta koja se sastoji od istih ili manje kolona takođe mora biti postavljena na strani pretplatnika. Ako se tabela bez identiteta replike doda publikaciji koja replicira UPDATE ili DELETE operacije, onda će naknadne UPDATE ili DELETE operacije izazvati grešku na publisheru. INSERT operacije se mogu nastaviti bez obzira na bilo koji identitet replike.

Publikacija se kreira pomoću CREATE PUBLICATION komande i može kasnije biti izmenjena ili odbačena korišćenjem odgovarajućih komandi. Pojedinačne tabele se mogu dodavati i uklanjati dinamički pomoću ALTER PUBLICATION. I operacije ADD TABLE ili DROP TABLE transakcione; tako da će tabela početi ili prestati da se replicira na tačnom snapshotu jednom kada se transakcija izvrši odnosno bude commitovana.

## Pretplata (Subscription)

Pretplata je nizvodna strana logičke replikacije. Čvor gde je pretplata definisana naziva se pretplatnik odnosno subscriber. Pretplata definiše vezu sa drugom bazom podataka i skupom publikacija (jednom ili više) na koje želi da se pretplati.

Baza podataka pretplatnika se ponaša na isti način kao i svaka druga PostgreSQL instanca i može se koristiti kao izdavač za druge baze podataka definisanjem sopstvenih publikacija.

Pretplatnički čvor može imati više pretplata po želji. Moguće je definisati više pretplata između jednog para izdavač-pretplatnik, i u tom slučaju se mora voditi računa da se objekti publikacije nad kojim je izvršeno pretplaćivanje ne preklapaju. Svaka pretplata će primiti promene preko jednog slota za replikaciju. Dodatni slotovi za replikaciju mogu biti potrebni za početnu sinhronizaciju podataka već postojećih podataka tabele i oni će biti odbačeni na kraju sinhronizacije podataka.

Pretplata kod logičke replikacije može biti stanje pripravnosti (standby) za sinhronu replikaciju. Ime standby-a je podrazumevano ime pretplate. Alternativno ime se može navesti kao application\_name u informacijama o vezi pretplate. Pretplate se smestaju u pg\_dump ako je trenutni korisnik superkorisnik. U suprotnom se piše upozorenje i pretplate se preskaču, jer nesuperkorisnici ne mogu da pročitaju sve informacije o pretplati iz pg\_subscription kataloga.

Pretplata se dodaje pomoću CREATE SUBSCRIPTION i može se zaustaviti / nastaviti u bilo kom trenutku pomoću ALTER SUBSCRIPTION komande i ukloniti pomoću DROP SUBSCRIPTION komande. Kada se pretplata odbaci i ponovo kreira, informacije o sinhronizaciji se gube. To znači da se podaci moraju naknadno ponovo sinhronizovati. Definicije šeme se ne repliciraju, a objavljene tabele moraju postojati na pretplatniku. Samo regularne tabele mogu biti deo replikacije, dok na primer, ne možet se replicirati kreirani pogled (View).

Tabele se podudaraju između publishera i subscribera koristeći kvalifikovano ime tabele. Replikacija tabela sa različitim imenima na pretplatniku nije podržana. Kolone tabele se takođe podudaraju po imenu. Redosled kolona u tabeli pretplatnika ne mora da se podudara sa redosledom kolona publishera. Tipovi podataka kolona ne moraju da se podudaraju, sve dok se tekstualni prikaz podataka može konvertovati u ciljni tip. Na primer, može se izvršiti replickacija iz kolone tipa integer u kolonu tipa bigint. Ciljna tabela takođe može imati dodatne kolone koje nisu obezbeđene u objavljenoj tabeli. Sve takve kolone će biti popunjene podrazumevanom vrednošću kako je navedeno u definiciji ciljne tabele.

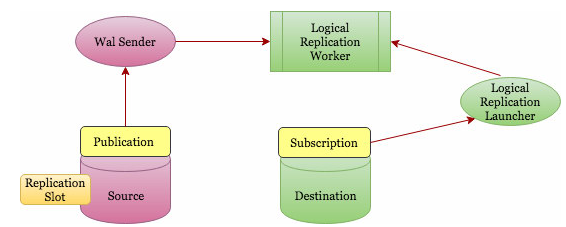
## Upravljanje slotovima za replikaciju (Replication Slot Management)

Kao što je ranije pomenuto, svaka (aktivna) pretplata prima promene iz slota za replikaciju na udaljenoj (publishing) strani.Dodatni slotovi za sinhronizaciju tabele su obično prolazni, kreirani interno da bi se izvršila početna sinhronizacija tabele i automatski se ispuštaju kada više nisu potrebni. Ovi slotovi za sinhronizaciju tabela imaju generisana imena: “ *pg\_%u\_sync\_%u\_%llu*” (parametri koji se koriste su: Subscription *oid*, Table *relid*, system identifier *sysid*)

Obično se slot za udaljenu replikaciju kreira automatski kada se pretplata kreira pomoću CREATE SUBSCRIPTION i automatski se odbacuje kada se pretplata odustane pomoću DROP SUBSCRIPTION. U nekim situacijama, međutim, može biti korisno ili neophodno da se zasebno manipuliše pretplatom i osnovnim slotom za replikaciju. Evo nekoliko scenarija:

* Kada se kreira pretplata, slot za replikaciju već postoji. U tom slučaju, pretplata se može kreirati korišćenjem *create\_slot = false* opcije za povezivanje sa postojećim slot-om.
* Kada kreirate pretplatu, udaljeni host nije dostupan ili je u nejasnom stanju. U tom slučaju, pretplata se može kreirati pomoću connect = false opcije. Udaljeni host tada uopšte neće biti kontaktiran. Ovo je ono što pg\_dump koristi. Slot za udaljenu replikaciju će tada morati da se kreira ručno pre nego što se pretplata može aktivirati.
* Slučaj kada odustanete od pretplate, slot za replikaciju se treba zadržati. Ovo bi moglo biti korisno kada se baza podataka pretplatnika premešta na drugi host i odatle će se aktivirati. U tom slučaju, odvojite slot od pretplate koristeći ALTER SUBSCRIPTION pre nego što pokušate da otkažete pretplatu.
* Slučaj kada odustanete od pretplate, a udaljeni host nije dostupan. U tom slučaju, odvojite slot od pretplate koristeći ALTER SUBSCRIPTION pre nego što pokušate da otkažete pretplatu. Ako instanca udaljene baze podataka više ne postoji, nisu potrebne dalje radnje. Međutim, ako je instanca udaljene baze podataka samo nedostupna, slot za replikaciju (i sve preostale slotove za sinhronizaciju tabele) bi tada trebalo ručno da se odbace; u suprotnom bi on/oni nastavili da rezervišu WAL i mogli bi na kraju dovesti do popunjavanja diska. Takve slučajeve treba pažljivo istražiti.

Logička replikacija se ponaša slično kao normalne DML operacije u smislu da će podaci biti ažurirani čak i ako su promenjeni lokalno na pretplatničkom čvoru. Ako dolazni podaci krše bilo koja ograničenja, replikacija će se zaustaviti, tako dolazi do konflikta. Kada se repliciraju operacije UPDATE ili DELETE, podaci koji nedostaju neće proizvesti konflikt i takve operacije će jednostavno biti preskočene.Konflikt će proizvesti grešku i zaustaviće replikaciju; korisnik mora da ga reši ručno. Rešavanje se može izvršiti ili promenom podataka o pretplatniku tako da nisu u suprotnosti sa dolaznom promenom ili preskakanjem transakcije koja je u sukobu sa postojećim podacima. Transakcija se može preskočiti pozivanjem funkcije *pg\_replication\_origin\_advance()* sa node\_name koje odgovara imenu pretplate i pozicijom. Trenutna pozicija porekla može se videti u sistemskom prikazu *pg\_replication\_origin\_status.* Na slici ispod prikazan je dijagram arhitekture logičke replikacije:



Slika 13. arhitektura Logičke replikacije

Kao što vidimo na slici iznad, logička replikacija počinje kopiranjem snimka podataka u bazi podataka izdavača. Kada se to uradi, promene na izdavaču se šalju pretplatniku kako se dešavaju u realnom vremenu. Subscriber prihvata podatke u redosledom kojim je obavljeno izvršavanja na strani publishera, tako da je zagarantovana konzistentnost transakcija za publikacije u okviru bilo koje pojedinačne pretplate.

Logička replikacija je izgrađena sa arhitekturom sličnom fizičkoj striming. Implementiran je procesima “ *walsender* “ i „ *application* “ . Walsender proces započinje logičko dekodiranje WAL-a i učitava standardni dodatak za logičko dekodiranje (*pgoutput*). Dodatak transformiše promene pročitane iz WAL-a u protokol logičke replikacije i filtrira podatke prema specifikaciji publikacije. Podaci se zatim kontinuirano prenose korišćenjem protokola replikacije strimovanja do work applay-a, koji mapira podatke u lokalne tabele i primenjuje pojedinačne izmene kako se primaju, u ispravnom transakcionom redosledu. Proces primene na bazu podataka pretplatnika uvek se pokreće sa *session\_replication\_role* postavljenom na repliku, što izaziva određeni efekte na pokretače i ograničenja. Nadgledanje čvora izdavača je identično kao nadgledanje primarnog čvora kod fizičke replikacije. Nadgledanje informacija o čvoru pretplatnika tj. subscriber-a moguće je obaviti korišćenjem *pg\_stat\_subscription*.

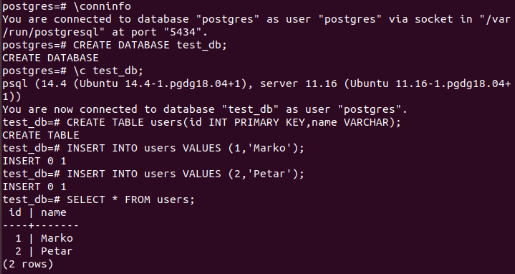
Ograničenja prilikom upotrebe logičke replikacije:

* Tabele moraju imati isto puno kvalifikovano ime između objavljivanja i pretplate.
* Tabele moraju imati primarni ključ ili jedinstveni ključ
* Uzajamna (dvosmerna) replikacija nije podržana
* Ne replicira šemu/DDL
* Ne replicira sekvence
* Ne replicira TRUNCATE
* Ne replicira velike objekte
* Pretplate mogu imati više kolona ili drugačiji redosled kolona, ali tipovi i nazivi kolona moraju da se podudaraju između publikacije i pretplate.
* Privilegije superkorisnika za dodavanje svih tabela
* Ne možete da prenosite na isti host (pretplata će se zaključati).

Primer kofigurisanja i korišćenja Logičke replikacije prikazan je u nastavku.

Prvo kako bi smo obezbedili logičku replikaciju kod PostgreSQL-a potrebno pokrenuti dve istance PostgreSQL sesrvera, koje koriste različite portove i koje će predstavljati Publisher i Subscriber čvorove. Nakon toga potrebno je izvršiti određene izmene u okviru postgresql.conf fajla. Potrebno je zaustaviti PostgreSQL server ukoliko je pokrenut a zatim postaviti parametar wal\_level = logical. Parametar govori serveru da skladišti dodatne informacije u WAL-u za pretvaranje binarnih promena u logičke. Nakon toga potrebno je ponovo pokrenuti PostgreSQL server. Kada je logička replikacija uključena možemo nastaviti sa izvršavanjem.

U okviru Publisher-a (koja u ovom slučaju radi na port 5434) prvo će biti kreirana nova baza podataka (test\_db), a zatim ćemo u okviru te baze kreirati tabelu users i dodati podatke u tabelu. Kao što je prikazano na slici ispod:



Slika 14. Kreiranje baze i tabele na Publisheru

Kada smo kreirali tabelu i popunili je podacima, sada je na redu kreiranje publikacije, na koju će se kasnije moći pretplatiti drugi serveri kako bi se podataci sa publishera replicirali na njih. Kreiranje publikacije prikazano je na slici ispod:



Slika 15. Kreiranje publikacije pod nazivom my\_publication za tabelu users.

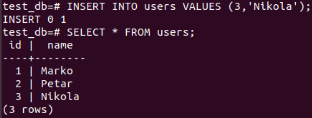
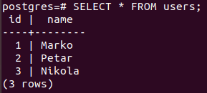
Nakon kreiranja publikacije, na drugoj instanci servera (koji radi na portu 5435) moramo pre pretplaćivanja napraviti tabelu koja po nazivu i parametrima odgovara tabeli iz publishera. Nakon toga ćemo kreirati pretplatu (Subscription) i konekcioni string servera na kome se Publisher nalazi. Ukolko je pretplata uspešno obavljena biće izvršena replikacija podataka iz tabele users.



Slika 16. Kreiranje pretplate my\_subscription na postojeću publikaciju

Kao što vidimo na slici iznad kreirali smo praznu tabelu users, a zatim smo uspešno kreirali pretplatu my\_subscription na publikaciju my\_publication koja se nalazi na instanci servera čiji je port 5434, a naziv baze test\_db.

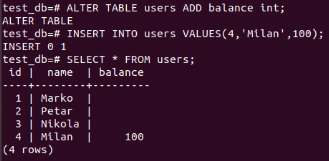
Kako bi testirali da li replikacija radi dodaćemo jedan red u našoj kreiranoj publikaciji, a zatim da li se izvršila replikacija na pretplatnika.

Slika 17. dodavanje novog reda na publikaciji i provera stanja na kreiranoj pretplati

Kao što vidimo na slici sve izmene na Publikaciji uspešno su preslikane na Subscription-u.

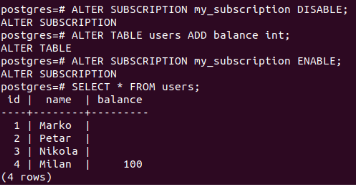
Sledeći slučaju je da u okviru publikacije dođe do izmene šeme tabele replikacija se neće obaviti na pretplatniku ukoliko nije ažurirana i šema na njumu.



Slika 18. modifikacija šeme tabele na publisheru

Na slici iznad, na publisheru smo izmenili izgled tabele users dodavanjem nove kolone balance, a zatim smo dodali i novi red u tabeli. Budući da šema tabele nije ažuirarana na pretplatniku neće doći do replikacije izmena načinjenih u publisheru.

U tom slučaju potrebno je privremeno zaustaviti kreiranu pretplatu, izvršiti migraciju šeme odnosno odgovarajuća ažuriranja kako bi se uskladile šeme i nakon toga nastaviti sa pretplatnom i replikacija će se obaviti. Ovaj slučaj prikazan je na slici ispod:



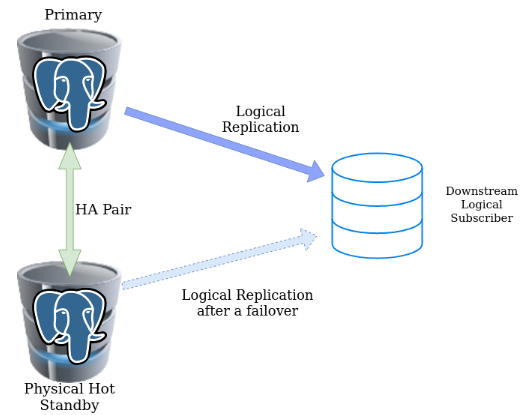
Slika 19. ručno prilagođavanje šeme pretplatnika

Za slučaj ako se dodaju nove tabele u već kreiranoj publikacija, mora se ručno dodati u publikaciji kako bi bila dostupna pretplatnicima za replikaciju, naredbom:

ALTER PUBLICATION my\_publication ADD TABLE newly\_added\_table;

Mogućnost logičkog dekodiranja postoji u PostgreSQL-u u poslednjih nekoliko verzija, a mnogo više funkcionalnosti se dodaje tokom vremena. Međutim, jedna od veoma ključnih funkcionalnosti još uvek nedostaje u PostgreSQL-u, što sprečava da logička replikacija bude deo kritičnih proizvodnih okruženja u kojima se očekuje da se logičke promene pouzdano prenose preko veze sa bazom podataka – čak i nakon prelaska na grešku visoke dostupnosti.

Koren problema je u tome što slot logičke replikacije koji postoji na primarnom serveru (publisher) nije dostupan u fizičkom stanju pripravnosti. U međuvremenu, skoro sva PostgreSQL rešenja visoke dostupnosti zavise od fizičkog stanja pripravnosti. Dakle, ako dođe do failover-a, novi primarni server (odnosno stari fizički standby) neće biti svestan bilo kakvih slotova koji su postojali sa starim primarnim serverom (odnosno neće znati za ostale pretplaćene servere na primarni server). Ovo dovodi do potpuno prekinute logičke replikacije (ne kontinuirane) nizvodno. Možemo da napravimo novi slot na novom primarnom i podesimo novu replikaciju, ali ako je originalna logička replikacija kasnila, to dovodi do gubitka podataka.



U idealnom slučaju, trebalo bi da postoji način da se označi da li je slot „failover slot“ tako da će odgovarajuće informacije biti zapisane u WAL-u. Ali svaki standby koje prima WAL-ove uključujući kaskadne replike će kreirati i održavati slotove. Ovo će rezultirati nepotrebnim troškovima i zadržavanjem WAL-a, održavanjem starih verzija sistemskog kataloga, itd. Ne postoji generički način da se ovo reši za sve.

Jedno od alternativnih rešenja je da se koristi PostgreSQL dodataka *pglogical*, koje pruža podršku za prevazilaženje failover-a i na strani primarnog servera i na strani pretplatnika. Ovaj dodatak koristi fizičku striming replikaciju za rešavanje failover-a.

Uz odgovarajuću konfiguraciju na serveru publisher-a i na rezervnom serveru publisher-a (koji bi predstavljao fizički standby), pretplatnici bi mogli nakon failover-a da se prebace na promovisani rezervni server publishera koji je postao glavni server (koristeći streaming replikaciju).

U slučaju otkazivanje primarnog (publisher) servera rezervni server će ga zameniti, a svi pretplatnici primarnog servera će dosledno pratiti prelazak na rezervni server. Koraci kako bi se izveo ovaj postupak su:

Publisher1 i publisher2 koriste streaming replikaciju.

* Na publisheru2 potrebno je podesiti recovery.conf fajl stavljajuci *primary\_conninfo* da ukazuje na publisher, I *primary\_slot\_name* stavljajuci naziv fizičke replikacije slota na publisher1
* Takođe na publisher2 podešavamo postgresql.conf:
  + *pglogical* stavljamo na *shared\_preload\_libraries*
  + *hot\_standby = on*
  + *hot\_standby\_feedback = on*
  + [*pglogical.synchronize\_failover\_slot\_names*](https://content-www.enterprisedb.com/docs/pglogical/latest/configuration/#pglogical-synchronize-failover-slot-names)  podešavamo na specifičan slot koji treba biti sinhronizovan
* Na publisher1 u okviru postgresql.conf podešavamo *pglogical.stanby\_slot\_names* postavljamo listu slotova korišćenih od strane publisher2. Zamena bi idalje radila i bez postavljanja ove opcije, ali pretplatnici mogu postati nekonzistentni u toku prebacivanja na publisher2
* Publisher2 je imao vremena da se sinhronizuje i napravio je kopiju slota za logičku replikaciju Publisher1. pglogical3 automatski kreira glavne slotove na replikama kada rezervacije resursa replike mogu da zadovolje zahteve glavnog slota.
* Publisher2 preuzima IP adresu ili ime hosta publisher1 ili je postojeća pretplata publisher1 rekonfigurisana da se poveže sa publisher2 koristeći *pglogical.alter\_node\_add\_interface* i *pglogical.alter\_subscription\_interface.*

Nakon ovog procesa publisher2 se ponaša kao primarni server, koji je idalje zadržava logičku replikaciju nad preuzetim pretplatnicima.

# Zaključak

Kao što smo već rekli replikacija podataka predstavlja jedan od ključnih koncepata, kojim se obezbeđuje dostupnost podataka u radu sa sistemima za upravljanje bazama podataka. Korišćenjem različitih metoda i alata PostgreSQL-a može se obezbediti visoka dostupnost i uravnoteženo opterećenje na serveru baze podataka održavanjem više servera baze podataka koji mogu da rade zajedno i implementiraju replikaciju da opslužuju veliki broj korisnika bez mnogo uticaja na performanse. Međutim, potrebno je voditi računa dok se izvodi replikacija, potrebno je uverite se da su sva podešavanja ispravno postavljena za odgovarajuće servere baze podataka u zavisnosti od uloge koju će obavljati u replikaciji i rukovanju zahtevima. Takođe potrebno je voditi računa koja tehnika će se koristiti prilkom obavljanje replikacije, jer kao što smo videli u radu svaka tehnika u zavisnosti od slučaja korišćenja ima svoje prednosti i mane.

# Literatura

[1] PostgreSQL replikacija baze podataka: <https://www.vertabelo.com/blog/postgresql-database-replication/>

[2] Tipovi replikacija kod PostgreSQL i failover: <https://www.enterprisedb.com/postgres-tutorials/postgresql-replication-and-automatic-failover-tutorial>

[3] Fizička replikacija kod PostgreSQL: <https://pgdash.io/blog/postgres-physical-replication.html>

[4] PostgreSQL log shipping (warm standby I streaming replikacija): <https://www.postgresql.org/docs/current/warm-standby.html>

[5] Streaming replikacija kod PostgreSQL : <https://www.scalingpostgres.com/tutorials/postgresql-streaming-replication/>

[6] Logička replikacija kod PostgreSQL: <https://severalnines.com/database-blog/overview-logical-replication-postgresql>

[7] Konfigurisanje logičke replikacije: <https://www.enterprisedb.com/postgres-tutorials/logical-replication-postgresql-explained>

[8] Primeri upotrebe logičke replikacije: <https://pgdash.io/blog/postgres-replication-gotchas.html>

[9] Logička replikacija: <https://www.postgresql.org/docs/14/logical-replication.html>

[10] Failover kod logičke replikacije: <https://content-www.enterprisedb.com/docs/pglogical/latest/failover/>