Elektronski fakultet u Nišu

Seminarski rad

PostgreSQL - Replikacija podataka

Predmet: Sistemi za upravljanje bazama podataka

|  |  |
| --- | --- |
| Mentor: Aleksandar Stanimirović | Student: Marko Đokić 1022 |

*April, 2020.*

[**Visoka dostupnost, uravnoteženo opterećenje i replikacija**](#_s2zv5zjxhy34) **3**

[Najvažnija korišćenje replikacije](#_2g3jxm7xi5d3) 5

[Fizička i logička](#_7yn102icge3y) 5

[Modeli replikacije (Single-Master i Multi-Master)](#_rhmepmdrwx4l) 6

[**Streaming replikacija**](#_8bwvbmf1sgmz) **7**

[Razumevanje streaming replikacije](#_f0ypn6k876f2) 7

[Streaming replikacija u PostgreSQL između master i slave servera](#_pq3rmk5mhh5w) 9

Replikacija podataka

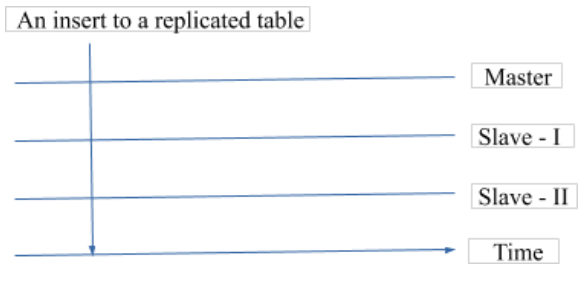
# Visoka dostupnost, uravnoteženo opterećenje i replikacija

Serveri baze podataka mogu raditi zajedno da omoguće brzo preuzimanje podataka sa drugog servera, ako se primarni server pokvari (visoka dostupnost - engl. high availability) ili da dozvole da više servera opslužuju korisnike sa istim podacima (balansiranje opterećenja - engl. load balancind). Najbolje bi bilo da serveri baza podataka rade zajedno neprimetno. Web serveri koji imaju statičke Web stranice mogu ih lako kombinovati jednostavnim balansiranjem Web zahteva na više mašina. U suštini, serveri baza podataka koji služe samo za čitanje mogu se vrlo jednostavno kombinovati. Nažalost većina servera baze podataka ima zahteve i za čitanje i za upis, a te servere je mnogo teže kombinovati. To je zato što podatke koje želimo da koristimo samo za čitanje možemo da upišemo na sve servere samo jednom, a pisanje na bilo koji server mora se proširiti na sve servere kako bi budući zahtevi za čitanje na tim serverima vratili konzistentne podatke.

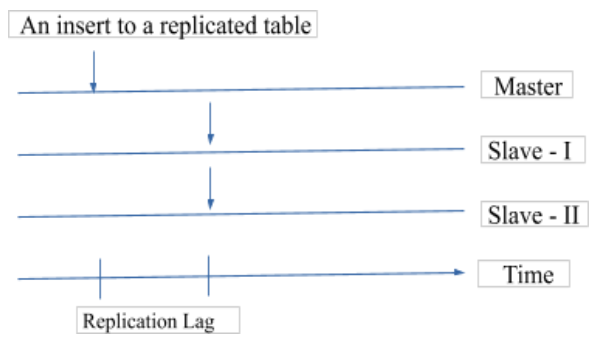
Ovaj problem sinhronizacije je osnovna poteškoća za servere koji rade zajedno. Kako ne postoji jedinstveno rešenje koje eliminiše uticaj problema sa sinhronizacijom za sve slučajeve upotrebe, postoji više rešenja. Svako rešenje rešava ovaj problem na drugačiji način i minimizira njegov uticaj na određeno radno opterećenje.

Neka rešenja se bave sinhronizacijom tako što dozvoljavaju samo jednom serveru da izmeni podatke. Serveri koji mogu modifikovati podatke zovu se read/write, master ili primarni serveri. Serveri koji prate promene u master-u nazivaju se standby(rezervni) ili slave serveri. Server u stanju standby koji nije moguće povezati dok nije promovisan u master server naziva se warm standby server, a onaj koji može prihvatiti veze i poslužiti upite samo za čitanje naziva se hot standby server.

Neka rešenja su sinhrona, što znači da se transakcija za promenu podataka ne smatra izvršenom dok svi serveri nisu izvršili transakciju. Ovo garantuje da prebacivanje neće izgubiti nikakve podatke i da će svi serveri vratiti iste rezultate bez obzira na koji server se traži. Suprotno tome, asinhrona rešenja omogućavaju određeno odlaganje između vremena izvršenja i njegovog prosleđivanja na ostale servere, otvarajući mogućnost da se neke transakcije izgube u prelasku na rezervni server i da serveri mogu vratiti malo zastarele rezultate. Asinhrona komunikacija se nekad koristi u slučaju da je sinhrona prespora.



Slika 1. Skica sinhrone replikacije



Slika 2. Skica asinhrone replikacije

Rešenja se takođe mogu kategorisati prema njihovoj granularnosti. Neka rešenja mogu da se bave isključivo sa celim serverom baza podataka, dok druga omogućavaju kontrolu na nivou tabele ili na bazi baze podataka.

Performanse se moraju uzeti u obzir u bilo kojem izboru. Obično postoji kompromis između funkcionalnosti i performansi. Na primer, potpuno sinhrono rešenje preko spore mreže može da smanji performanse za više od polovine, dok bi asinhrono moglo imati minimalan uticaj na performanse.

## Najvažnija korišćenje replikacije

* Kod OLTP (Online Transaction Processing), odnosno onlajn obrade transakcija, replikacija smanjuje dodatna procesiranja izveštaja na mreži, poboljšava vreme trajanja upita i performanse obrade upita.
* Fault tolerance, odnosno tolerancija na greške: U slučaju kvara glavnog servera, određeni server može preuzeti posao, zato što već sadrži podatke iz matičnog servera. U ovo konfiguraciji podređeni server se naziva i rezervni server. Ova konfiguracija se takođe može koristiti za redovno održavanje primarnog servera.
* Migracija podataka: Kada je potrebna nadogradnja hardvera servera baze podataka ili za upotrebu istog sistema za drugog klijenta, moguće je upotrebiti replikaciju.
* Paralelno testiranje sistema: Kada se prebacuje aplikacija iz jednog DBMS u drugi, rezultati na istim podacima i starog i novog sistema moraju se upoređivati kako i se osiguralo da novi sistem funkcioniše kako se očekuje.

## Fizička i logička

Fizička replikacija se bavi datotekama i direktorijumima. Nema znanja o tome šta ove datoteke i direktorijumi predstavljaju. Ona se vrši na nivou fajl sistema ili nivou disku.

S druge strane, logička replikacija se bavi bazama podataka, tabelama i DML operacijama. Stoga je moguće u logičkoj replikaciji kopirati samo određeni skup tabela. Logička replikacija se vrši na nivou klastera baze podataka.

## Modeli replikacije (Single-Master i Multi-Master)

Kod Single-Master replikacije (SMR) promene u redovima tabele u master bazi podataka repliciraju se na jedan ili vise slave servera. Replicirane tabele u slave bazi podataka nemaju dozvolu za prihvatanje bilo kakvih promena osim promena koje se dese na master serveru. Ali čak i ako se to dogodi promene se ne dodaju i na master server.

Kod Multi-Master replikacije (MMR), promene u redovima tabele u više od jedne master baze podataka repliciraju se i u ostale master baze podataka. U ovom se modelu često koriste šeme za rešavanje sukoba kako bi se izbegli problemi poput duplikata primarnih ključeva. Moguće je korišćenje široke mreže (WAN) master baza podataka koje mogu biti izdvojene geografski, da bi bile bliže svim klijentima, a istovremeno održavaju konzistentnost podataka širom mreže.

Klase replikacije mogu biti jednosmerne (unidirectional) kod Single-Master baza i dvosmerne (bidirectional) kod Multi-Master replikacije podataka, jer podaci teku u oba smera.

# Streaming replikacija

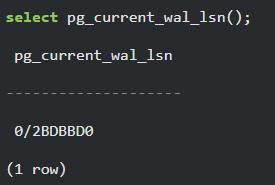
Konfigurisanje replikacije između dve baze podataka smatra se najboljom strategijom za postizanje visoke dostupnosti tokom katastrofa i pruža toleranciju grešaka od neočekivanih kvarova. PostgreSQL omogućava replikaciju putem striming (streaming) replikacije. Pored striming replikacije postoji i logička replikacije i logičko dekodiranje.

## Razumevanje streaming replikacije

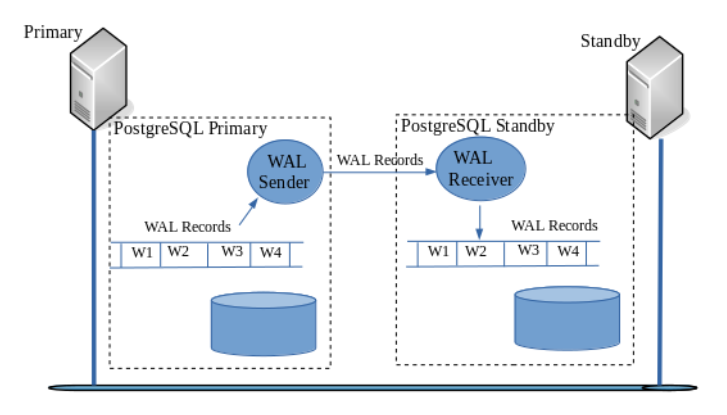
Streaming replikacija kod PostgreSQL radi po principu prosleđivanja logova. Svaka transakcija kod PostgreSQL se zapisuje u transakcione logove koji se zovu WAL (write-ahead log) kako bi se postigla trajnost. Slave server koristi te WAL segmente da kontinuirano replicira promene iz mastera. Sve promene koje se izvrše prvo se čuvaju u fajlu logova. Svaka promena koju transakcija izvrši upisuje se u fajlu logova kao WAL zapis, WAL zapis se prvo upisuje u WAL bafer u memoriji. Kada se transakcija izvrši zapisi se zapisuju u datoteku WAL segmenta na disku.

Postoje tri obavezna proces - *wal sender*, *wal receiver* i *startup proces*, koji igraju glavnu ulogu u postizanju streaming replikacije kod PostgreSQL.

*Wal sender* proces se izvršava na masteru, dok *wal receiver* i *startup proces* se izvršavaju na slave procesu. Kada se pokrene replikacija, *wal receiver* proces šalje LSN (Log Sequence Number), LSN je pokazivač na lokaciju u WAL, sve do trenutka, kada se WAL podaci repliciraju na slave procesu. *Wal sender* proces na masteru šalje WAL podatke do poslednjeg LSN, počevši od LSN-a koji je poslao *wal receiver*, sa slave servera. *Wal receiver* upisuje WAL podatake poslate od strane *wal sendera* u WAL segmente. *Startup proces* na slave serveru replicira podatke zapisane u WAL segmentu. I tada streaming replikacija kreće.



Slika 3. primer LSN



Slika 4. Slikoviti prikaz streaming replikacije

Ako dođe do rušenja sistema, baza podataka može oporaviti počinjene transakcije iz WAL-a. Oporavak započinje od poslednje REDO tačke ili kontrolne tačke. Kontrolna tačka je tačka u dnevniku transakcija na kojoj su sve datoteke podataka ažurirane kako bi se uskladile sa podacima u zapisniku. Proces čuvanja WAL zapisa iz datoteke dnevnika u stvarne datoteke podataka naziva se check-pointing.

## Streaming replikacija u PostgreSQL između master i slave servera

Potrebno je kreirati korisnika u masteru koristeći bilo koji slave koji može da se poveže za streaming WAL-podataka. Ovaj korisnik mora imati REPLIACTION ROLE.

CREATE USER replicator

WITH REPLICATION

ENCRYPTED PASSWORD 'replicator';

Sledeći parametri na masteru su obavezni za podešavanje streaming replikacije:

* archive\_mode: Mora biti postavljeno na ON da bi se omogućilo arhiviranje WAL-ova.
* wal\_level: Mora biti postavljeno na ‘hot\_standby’ do verzije 9.5 ili ‘replica’ u kasnijim verzijama.
* max\_wal\_senders: Mora biti postavljeno na 3 ako se započinje sa jednim slave. Za svaki slave potrebno je dodati 2 wal sendera.
* wal\_keep\_segments: Podesite zadržavanje WAL-a u pg\_xlog i pg\_wal od verzije 10. Svakom WAL-U potrebno je 16mb prostora osim ako izričito niste promenili veličinu WAL segmenta.
* archive\_command: Ovaj parametar uzima šel komandu ili spoljašnje programe. To može biti jednostavna komanda za kopiranje WAL segmenta na drugu lokaciju ili skritpu koja ima logiku za arhiviranje WAL-ova na C3 ili na udaljeni rezervni server.
* listen\_addresses: Određuje koji IP interfejsi mogu da prihvate veze. Moće odrediti sve TCP/IP adrese na kojima bi server mogao da sluša veze sa klijentima. “\*” znači da su sve IP adrese dostupne. Podrazumevano stoji localhost, osnosno samo lokalnu TCP/IP konekciju može da uspostavi PostgreSQL server.
* hot\_standby: Mora biti postavljen na ON na standby/replica i nema efekta na master. Međutim kada postavite vašu replikaciju, parametar postavljen na masteru se automatski kopira. Ovaj parametar je važan kako bi se omogućilo čitanje u slave. U suprotnom ne može se pokrenuti SELECT upit na slave.

Parametre je moguće postaviti na master koristeći sledeće naredbe nakon čega je potreban restart:

*ALTER SYSTEM SET wal\_level TO 'hot\_standby';*

*ALTER SYSTEM SET archive\_mode TO 'ON';*

*ALTER SYSTEM SET max\_wal\_senders TO '5';*

*ALTER SYSTEM SET wal\_keep\_segments TO '10';*

*ALTER SYSTEM SET listen\_addresses TO '\*';*

*ALTER SYSTEM SET hot\_standby TO 'ON';*

*ALTER SYSTEM SET archive\_command TO 'test ! -f /mnt/server/archivedir/%f && cp %p /mnt/server/archivedir/%f';*

*$ pg\_ctl -D $PGDATA restart -mf*

Dalje je potrebno dodati unos u pg\_hba.conf fajl da bi se omogućile replikacijske veze sa slave. Podrazumevana lokacija pg\_hba.conf je u direktorijumu podataka.

*host replication replicator 192.168.0.28/32 md5*

IP adresa u prethodnoj linije mora se podudarati sa IP adresom slave servera.

Da bi promene stupile na snagu, izdaje se SIGHUP:

*$ pg\_ctl -D $PGDATA reload*

*ili*

*$ psql -U postgres -p 5432 -c "select pg\_reload\_conf()"*

pg\_basebackup služi za strimovanje podataka kroz wal sender proces iz mastera to slave da bi se postavila replikacija.

Sledeći korak se može koristiti za streamovanje direktorijuma podataka iz mastera u slave-u. Ovaj korak se može izvesti na slave-u.

*$ pg\_basebackup -h 192.168.0.28 -U replicator -p 5432 -D $PGDATA -P -Xs -R*

IP adresa u komandi iznad mora biti adresa mastera.

U gornjoj naredbi stoji i neobavezan argument -P. Kad prođe -P, automatski se kreira recovery.conf datoteka koja sadrži ulogu DB instance i detalje njenog mastera. Obavezno je kreirati recovery.conf datoteku na slave kako bi se postavila streaming replikacija. Izgled recovery.conf fajla:

*$ cat $PGDATA/recovery.conf*

*standby\_mode = 'on'*

*primary\_conninfo = 'host=192.168.0.28 port=5432 user=replicator password=replicator'*

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

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

U datoteci uloga servera je definisana pomoću standby\_mode. standby\_mode mora biti postavljen na ON za slave u PostgreSQL. I da bi se prenosili WAL podaci, detalji master servera se konfigurišu korišćenjem parametra primary\_conninfo.

Dva parametra standby\_mode i primary\_confinfo se kreiraju automatski kada se koristi opcioni argument -P kad se pozove pg\_basebackup. Ova datoteka mora da postoji u direktorijumu podataka ($PGDATA) na slave.

Nakon što su backup i restore kompletirani može se pokrenuti slave.

Ako je konfigurisan backup (na daljinu) korišćenjem pomenutog streaming metoda, to će samo kopirati sve fajlove i direktorijume u direktorijum podataka na slave. Što znači da su to oba backup-a direktorijuma master podataka i da omogućavaju vraćanje u jednom koraku.

Ako je preuzet tar backup sa mastera i poslat u slave, morate unzipovati backup u slave direktorijm podataka, prateći kreiranje recovery.cong pomenutog u prethodnim koracima. Jednom kada je urađeno, može se opet startovati PostgreSQL instancu na slave koristeći prateće komande.

*$ pg\_ctl -D $PGDATA start*

U proizvodnom okruženju je uvek poželjno da se parametar restore\_command postavi na odgovarajući način. Ovaj parametar sadrži šel komandu (ili skriptu) koja se može koristiti za pribavaljanje potrebnog WAL za slave ako WAL nije dostupan na masteru.

Na primer:

Ako je problem sa mrežom uzrokovao da slave zaostaje za masterom za značajan period, manje je verovatno da će WAL-ovi koje zahteva slave biti dostupni na lokaciji mastera pg\_xlog ili pg\_wal. Stoga je razumno arhivirati WAL-ove na sigunu lokaciju i imati komande potrebne za vraćanje WAL-a setovane u restore\_command u recovery.conf fajlu na slave. Da bi se to postiglo, u slave se mora dodati linija slična sledećem primeru.

*restore\_command = 'cp /mnt/server/archivedir/%f "%p"'*

Postavljanje gornjeg parametra zahteva ponovo pokretanje i ne može se izvršiti na mreži.

Finalni korak: potvrditi da je PostgreSQL replikacija postavljena

Kao što je već rečeno, wal sender i wal receiver procesi su pokrenuti na masteru i slave nakon postavljanja replikacije. Treba proveriti procese na master i slave serverima sledećim komandama.

Na masteru

==========

$ ps -eaf | grep sender

Na slave

==========

$ ps -eaf | grep receiver

$ ps -eaf | grep startup

U sledećem primeru mogu se videti logovi sva tri procesa.

Na masteru

=========

$ ps -eaf | grep sender

postgres 1287 1268 0 10:40 ? 00:00:00 postgres: wal sender process replicator 192.168.0.28(36924) streaming 0/50000D68

Na slave

=========

$ ps -eaf | egrep "receiver|startup"

postgres 1251 1249 0 10:40 ? 00:00:00 postgres: startup process recovering 000000010000000000000050

postgres 1255 1249 0 10:40 ? 00:00:04 postgres: wal receiver process streaming 0/50000D68

Za više informacija o replikaciji na masteru potrebno je pogledati sledeću stavku pg\_stat\_replication, kao u primeru sa slike.

$ psql

postgres=# \x

Expanded display is on.

**postgres=# select \* from pg\_stat\_replication;**

-[ RECORD 1 ]----+------------------------------

pid | 1287

usesysid | 24615

usename | replicator

application\_name | walreceiver

client\_addr | 192.168.0.28

client\_hostname |

client\_port | 36924

backend\_start | 2018-09-07 10:40:48.074496-04

backend\_xmin |

state | streaming

sent\_lsn | 0/50000D68

write\_lsn | 0/50000D68

flush\_lsn | 0/50000D68

replay\_lsn | 0/50000D68

write\_lag |

flush\_lag |

replay\_lag |

sync\_priority | 0

sync\_state | async

Literatura

[1] PostgreSQL documentation <https://www.postgresql.org/docs/9.2/runtime-config-replication.html>

[2] PostgreSQL documentation <https://www.postgresql.org/docs/9.1/high-availability.html>

[3] PostgreSQL documentation <https://www.postgresql.org/docs/9.0/different-replication-solutions.html>

[4] Set replication on ubuntu <https://www.howtoforge.com/tutorial/postgresql-replication-on-ubuntu-15-04/>

[5] Set up streaming replication PostgreSQL <https://www.percona.com/blog/2019/10/11/how-to-set-up-streaming-replication-in-postgresql-12/>

[6] Set up streaming replication PostgreSQL <https://www.percona.com/blog/2018/09/07/setting-up-streaming-replication-postgresql/>

[7] Automatic failover replication tutorial <https://www.enterprisedb.com/postgres-tutorials/postgresql-automatic-failover-replication-tutorial>