

UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

**BACKUP/RESTORE POSTGRESQL BAZE PODATAKA**

**-Seminarski rad-**

Student: Mentor:

Sara Savić, br.ind. 1758 Prof. dr Aleksandar Stanimirović

**SAŽETAK**

Backup i Restore predstavljaju krucijalne elemente u kontekstu baza podataka. Poznavanje ovih tehnika pruža siguronosnu mrežu zaštite podataka od gubitka, omogućavajući brz oporavak u slučaju problema i obezbeđujući kontinuitet poslovanja.

Seminarski rad “Backup/Restore PostgreSQL baze podataka”, kroz pet poglavlja, prikazuje osnovne koncepte kojima se PostgreSQL baza podataka služi kako bi zaštitila podatke korišćenjem različitih tehnika backup-ovanja, i ukoliko dođe do nekog oštećenja, oporavila oštećenu bazu podataka. U uvodnom poglavlju ovog seminarskog rada, na koncizan način, opisana je sama struktura seminarskog rada, dajući poseban osvrt na važnost poznavanja tehnika backup-ovanja i restore-ovanja podataka. Poglavlje “Backup/Restore kod relacionih baza podataka”, daje pregled definicija ovih pojmova, i opisuje kako se uopšteno vrše ovi procesi kod relacionih baza podataka. Poglavlje “Backup/Restore kod PostgreSQL baze podataka” fokusira se konkretno na procese backup-ovanja i restore-vanja kod PostgreSQL baze podataka i sadrži opis, ali i praktični pregled različitih tehnika koje PostgreSQL koristi, kao što su SQL Dump, backup na nivou fajl sistema i kontinuirano arhviranje i oporavak u određenoj tački vremena. Naposletku, finalno poglavlje se bavi opisom i prikazom kako je moguće izvršiti backup i restore u PostgreSQL bazi korišćenjem pgAdmin 4 alata.

Poglavlje “Zaključak” sumira celokupan prikaz ovog seminarskog rada, ponovo ukazujući na važnost i kompleksnost procesa backup-ovanja i restore-ovanja.

SADRŽAJ

[1.UVOD 4](#_Toc167620661)

[2.BACKUP/RESTORE KOD RELACIONIH BAZA PODATAKA 5](#_Toc167620662)

[2.1 Definicije i opis pojmova Backup/Restore kod relacionih baza podataka 5](#_Toc167620663)

[3.BACKUP/RESTORE KOD POSTGRESQL BAZE PODATAKA 7](#_Toc167620664)

[3.1 SQL Dump 7](#_Toc167620665)

[3.1.1 Obnavljanje/Vraćanje (Restoring) Dump-a 10](#_Toc167620666)

[3.1.2 pg\_dumpall 15](#_Toc167620667)

[3.1.3 Rad sa velikim bazama podataka 16](#_Toc167620668)

[3.2 Backup na nivou fajl sistema 22](#_Toc167620669)

[3.3 Kontinuirano arhiviranje i obnova do trenutka (PITR) 25](#_Toc167620670)

[3.3.1 Postavljanje WAL arhiviranja 26](#_Toc167620671)

[3.3.2 Pravljenje osnovnog backup-a(base backup) 29](#_Toc167620672)

[3.3.3 Pravljenje osnovnog backup-a(base backup) korišćenjem niskog nivoa API-ja 30](#_Toc167620673)

[3.3.4. Oporavak u tački vremena-Point-In-Time Recovery 34](#_Toc167620674)

[3.3.5. Vremenske linije 39](#_Toc167620675)

[4. KREIRANJE BACKUP-A I RESTORE-A KORIŠĆENJEM pgAdmin 4 ALATA 42](#_Toc167620676)

[5.ZAKLJUČAK 45](#_Toc167620677)

[6.SPISAK KORIŠĆENE LITERATURE 46](#_Toc167620678)

# 1.UVOD

U današnjem digitalnom dobu, informacije su postale najvredniji resurs, pa je samim tim važnost dostupnosti i sigurnosti podataka postala od suštinskog značaja. Baze podataka predstavljaju okosnicu mnogih informacionih sistema, pa zato one moraju biti zaštićene od različitih pretnji koje mogu dovesti do gubitka informacija. U ovom kontekstu, PostgreSQL se ističe kao jedan od najčešće korišćenih sistema za upravljanje relacionim bazama podataka u različitim industrijama.

Ipak, jedan od najvažnijih aspekata upravljanja bazama podataka jeste implementacija efikasnih strategija za backup i restore. Ovi procesi omogućavaju očuvanje integriteta i dostupnosti podataka, čak i u slučaju neočekivanih problema kao što su tehnički kvarovi, ljudske greške ili napadi zlonamernih softvera. Bez adekvatnog sistema za backup i restore, organizacije rizikuju gubitak kritičnih podataka, suočavajući se sa ozbljinim posledicama po poslovanje.

Seminarski rad “Backup/Restore kod PostgreSQL baze podataka”, istražuje različite tehnike koje PostgreSQL nudi za backup i restore, pružajući sveobuhvatan pregled teorijskih osnova i praktičnih primera. Rad je struktuiran tako da najpre uvodi osnovne pojmove i značaj ovih procesa u relacionim bazama podataka, a zatim detaljno, i kroz praktične primere, prikazuje specifične metode i tehnike koje PostgreSQL koristi.

# 2.BACKUP/RESTORE KOD RELACIONIH BAZA PODATAKA

Baze podataka su osnovna infrastruktura svake organizacije, pružajući sposobnost skladištenja, zaštite i pristupa podacima. Međutim, gubitak ovih podataka predstavlja stalnu pretnju za tehnološke departmane. Ljudske greške, prirodne katastrofe ili zlonamerno hakovanje mogu izazvati gubitak bitnih podataka iz baze, čime se ozbiljno ugrožava integritet i funkcionalnost organizacije. Zbog toga je ključno kreirati redovne rezervne kopije podataka (***eng.backup***) kako bi se osigurao integritet podataka. Takođe, važno je poznavati tehnike oporavka (povratka) baze podataka (***eng.restore***) kako bi se efikasno reagovalo u slučaju oštećenja ili gubitka podataka, čime se obezbeđuje stabilnost i funkcionalnost organizacije. U nastavku poglavlja biće date definicije pojmova kreiranja rezervnih kopija (***eng.backup***) i povratka baze podataka u konzistentno stanje (***eng.restore***), kao i detaljna objašnjenja ovih postupaka kod relacionih baza podataka.

## 2.1 Definicije i opis pojmova Backup/Restore kod relacionih baza podataka

Backup baze podataka predstavlja proces kreiranja, održavanja i čuvanja kopija podataka u slučaju njihovog gubitka ili oštećenja. Backup omogućava korisnicima da povrate podatke pre nego što postanu neupotrebljivi. Ovo može biti urađeno ručno ili automatski, pomoću specijalizovanih rešenja za backup baze podataka. [1]

Sam proces backup-a baze podataka sprovodi se kroz nekoliko ključnih faza:

1. **Identifikacija ključnih podataka:** Prvi korak je identifikacija ključnih podataka koji se moraju zaštiti. To obuhvata tabele, instance i logove koji su značajni za nesmetan tok poslovanja.

2. **Izbor tipa backup-a**: Nakon identifikacije ključnih podataka, sledeći korak je izbor tipa backup-a koji odgovara specifičnim potrebama poslovanja ili aplikacije.

3**. Lokacija za skladištenje podataka**: Važno je izabrati odgovarajuću lokaciju za skladištenje rezervnih kopija (***eng.backup***) podataka. To može biti lokalno skladište, oblak (***eng.cloud***) ili hibridna lokacija.

4. **Izvršenje backup-a**: Nakon postavljanja parametara backup-a, izvršava se proces backup-ovanja. Radi postizanja efikasnosti, moguće je automatizovati ovaj proces.

5. **Provera integriteta podataka**: Važno je redovno proveravati integritet podataka putem obnavljanja na drugoj lokaciji kako bi se osiguralo da su kopije tačne i pouzdane.

Efikasno backup-ovanje baze podataka počinje razumevanjem poslovnih mogućnosti i izborom odgovarajuće vrste backup-a za određenu situaciju. Postoji nekoliko načina za pravljanje backup-a baze podataka, a relacione baze podataka podržavaju tri sledeće metode [2]: puni (***eng.full***), inkrementalni (***eng.incremental***) i diferencijalni (***eng.differential***) backup.Puni backup predstavlja potpunu kopiju cele baze podataka, uključujući sve podatke i šemu. Ova metoda se smatra najpouzdanijom metodom backup-ovanja, ali istovremeno i najzahtevnijom. Inkrementalni backup predstavlja delimičnu kopiju baze podataka koja uključuje samo promene napravljane od poslednjeg backup-a, bilo da je u pitanju puni backup ili inkrementalni. U odnosu na puni backup, ova vrsta backup-ovanja je brža, ali zahteva sve prethodne backup-ove za obnavljanje baze podataka. Diferencijalni backup je takođe delimičan backup i uključuje samo promene koje su napravljene od poslednjeg punog backup-a. Diferencijalni backup je sporiji i veći od inkrementalnog, ali za obnavljanje baze podataka potreban je samo poslednji puni backup. Ipak, iako postoje tri moguće metode, izbor najbolje zavisi od veličine, učestalosti i ciljeva obnavljanja baze podataka.

Kod relacionih baza podataka postoje dve vrste backup-a: fizički i logički backup. Fizički backup baze podataka predstavlja kopije fajlova i direktorijuma baze podataka. Ova vrsta backup-a je korisna kada je potrebno vratiti celu bazu podataka. Fizički backup može biti ili puni ili inkrementalni. Puni fizički backup obuhvata kompletni skup podataka, arhiviranih i transkacionih fajlova, omogućavajući sveobuhvatno vraćanje podataka bez gubitka informacija. S druge strane, inkrementalna kopija je idealna za skoro obavljene transkacije jer omogućava brzu sigurnosnu kopiju bez prekida rada.Ipak, ovaj oblik može usporiti procese baze podataka.

Logički backup baze podataka obuhvata vitalne informacije kao što su strukutura tabele, funkcije, šeme, prikazi i procedure, i često se izvozi u obliku binarnog fajla. Redovno izvođenje punih logičkih backup-ova je važno kao rezervna opcija u nedostatku fizičkog backup-a. Ipak, važno je napomenuti da logički backup sam po sebi nije dovoljan za potpunu zaštitu podataka jer pruža samo strukturne informacije. Tokom oporavka, ove komponente se vraćaju na osnovnom nivou, zajedno sa svim međusobno povezanim elementima, koristeći odgovarajući alat za uvoz na određenoj platformi baze podataka. [1]

Oporavak podataka (***eng.restore***) poznat i kao vraćanje podataka, predstavlja važan proces u kome se kopirani siguronosni podaci sa sekundarnog skladišta vraćaju na originalnu lokaciju ili na novu lokaciju. Ovaj proces se sprovodi kako bi se povratili podaci koji su izgubljeni, ukradeni ili oštećeni, i vratili u njihovo prvobitno stanje ili premestili na novu lokaciju. Oštećenja mogu biti uzrokovana ljudskom greškom (na primer korisnik može slučajno obrisati ili na neki drugi način oštetiti podatke), korupcijom fajl sistema, nekim zlonamernim aktivnostima ili hardverskim kvarom [3]. Nakon što baza podataka pretrpi neko od ovih oštećenja, obnova (povratak) (***eng.restore***) podataka omogućava povratak sistema u ispravno i konzistetno stanje. Proces obnove podataka uključuje obnavljanje najnovije kopije baze podataka i primenu transkacionih siguronosnih kopija kako bi se ponovo uspostavile operacije. Oporavak podataka je od vitalnog značaja za očuvanje integriteta podataka i sprečavanje gubitka podataka usled različitih nepredviđenih situacija poput ljudskih grešaka, sistemskih havarija ili cyber napada.[4]

Ovime je dat opšti pregled pojmova backup i restore u kontekstu relacionih baza podataka, a u nastavku seminarskog biće opisano i prikazano kako je moguće izvršiti backup, a kasnije i restore, na primeru jedne od najpoznatijih relacionih baza podataka, PostgreSQL baze podataka.

# 3.BACKUP/RESTORE KOD POSTGRESQL BAZE PODATAKA

Baze podataka, kako je već rečeno u poglavlju 2, predstavljaju skladišta bitnih informacija zbog čega su procesi backup-a i restore-a podataka od ključnog značaja. PostgreSQL baza podataka, kao otvoreni sistem za upravljanje relacionim bazama podataka, takođe zahteva redovno backup-ovanje. Ovaj sistem pruža moćne alate i tehnike za očuvanje integriteta podataka kroz operacije backup-ovanja i restore-ovanja.

Postoje tri fundamentalno različita pristupa za backup-ovanje PostgreSQL podataka [5]:

1. SQL Dump (logički backup)

2. Backup na nivou fajl sistema (fizički backup)

3. Kontinuirano arhiviranje

Svaki od ovih pristupa ima svoje prednosti i mane. U nastavku poglavlja biće detaljno opisani i prikazani praktični primeri za svaki od ova tri načina backup-ovanja, kao i načini na koje je moguće izvršiti obnovu korišćenjem ovih backup-ova.

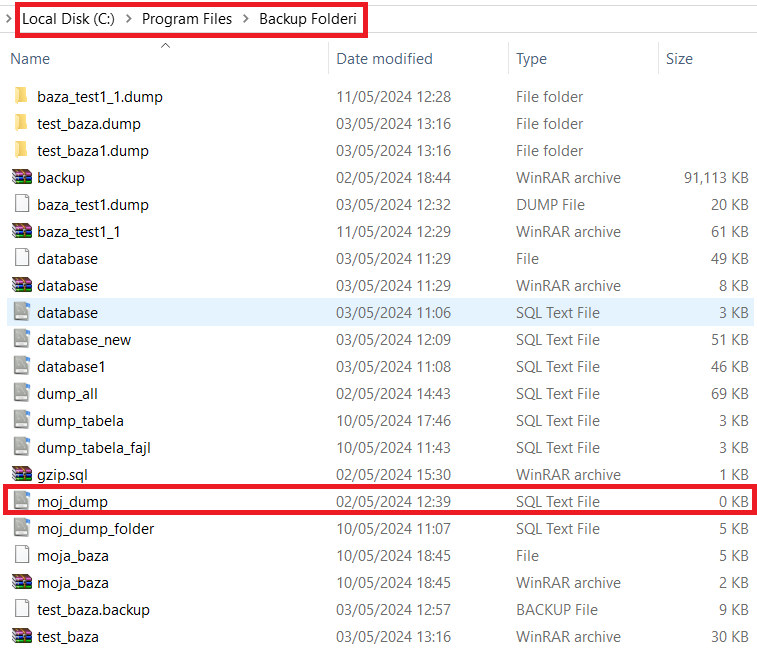
## 3.1 SQL Dump

Osnovna ideja metode SQL Dump jeste generisanje datoteke sa SQL naredbama, koje će, kada se ponovo proslede serveru, ponovo rekreirati bazu podataka u istom stanju kao što je bila u trenutku kreiranja “dump”-a. Za ovu svrhu, PostgreSQL pruža alat koji se naziva **pg\_dump** [6]**.** Osnovna upotreba ove komande je:



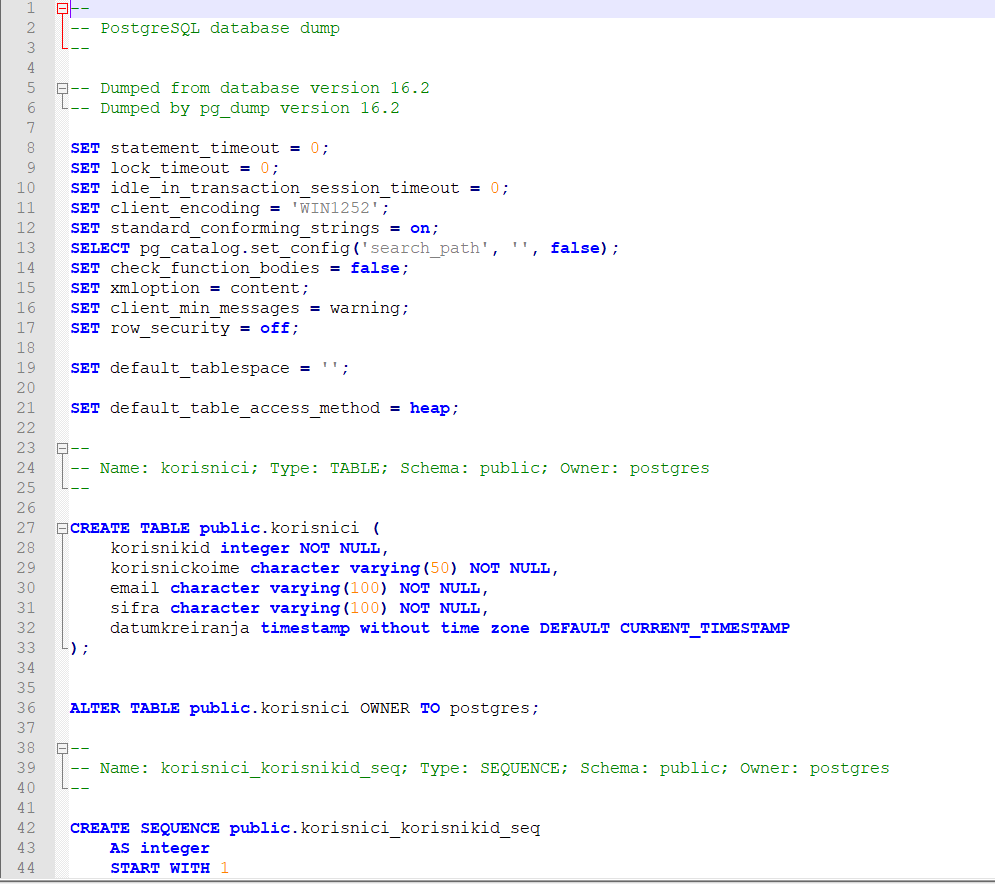
Slika 1-Prikaz kreiranja SQL Dump-a korišćenjem pg\_dump komande

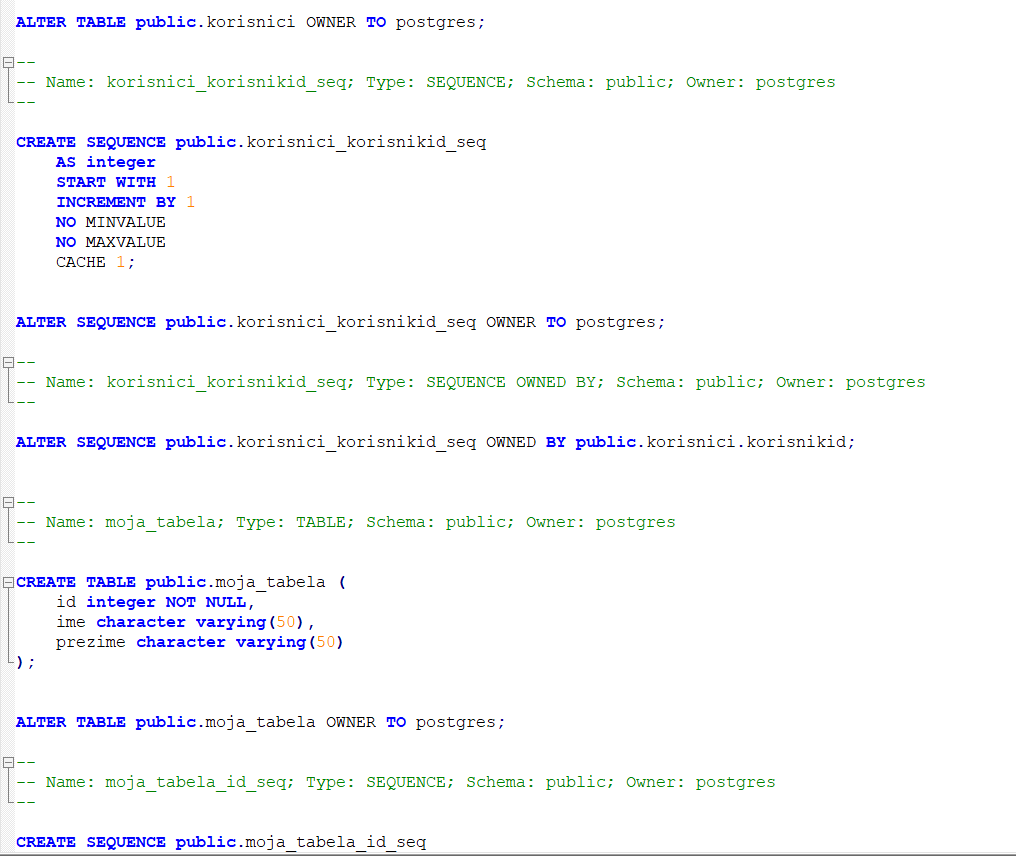
Izvršenjem naredbe koja je prikazana na slici 1 kreiran je SQL Dump za bazu “moja\_baza” na lokaciji "C:\Program Files\Backup Folderi\moj\_dump.sql".

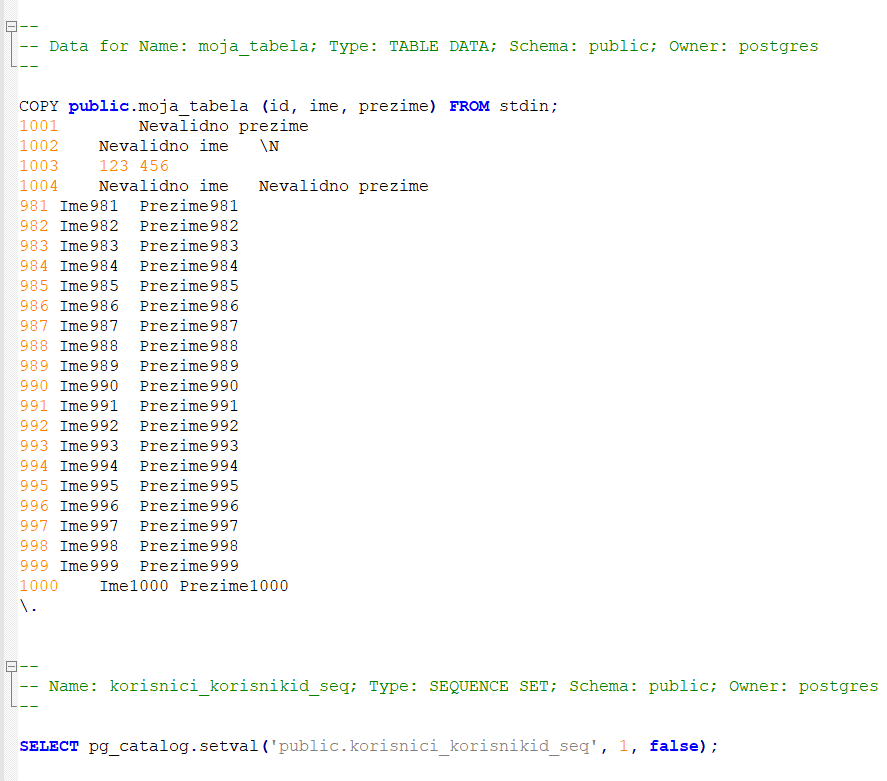


Slika 2-Prikaz kreiranog dump-a za bazu podataka "moja\_baza” na lokaciji "C:\Program Files\Backup Folderi\moj\_dump.sql"

Dump fajl sadrži SQL komande koje opisuju strukturu baze podataka uključujući definicije tabela, indeksa, ograničenja, funkcija i drugih objekata baze podataka. Osim toga, dump fajl sadrži i SQL komande koje sadrže podatke iz tih tabela, omogućavajući rekreiranje baze podataka sa istim sadržajem kao u trenutku kada je dump izvršen. Sadržaj “dump” fajla prikazan je na slici 3.







Slika 3- Prikaz sadržaja “dump” fajla za bazu “moja\_baza”

pg\_dump svoj rezultat ispisuje na standradni izlaz. Iako komanda koja je prikazana na slici 1 kreira tekstualnu datoteku, pg\_dump može kreirati datoteke i u drugim formatima koji omogućavaju paralelizam i detaljniju kontrolu obnove objekata.

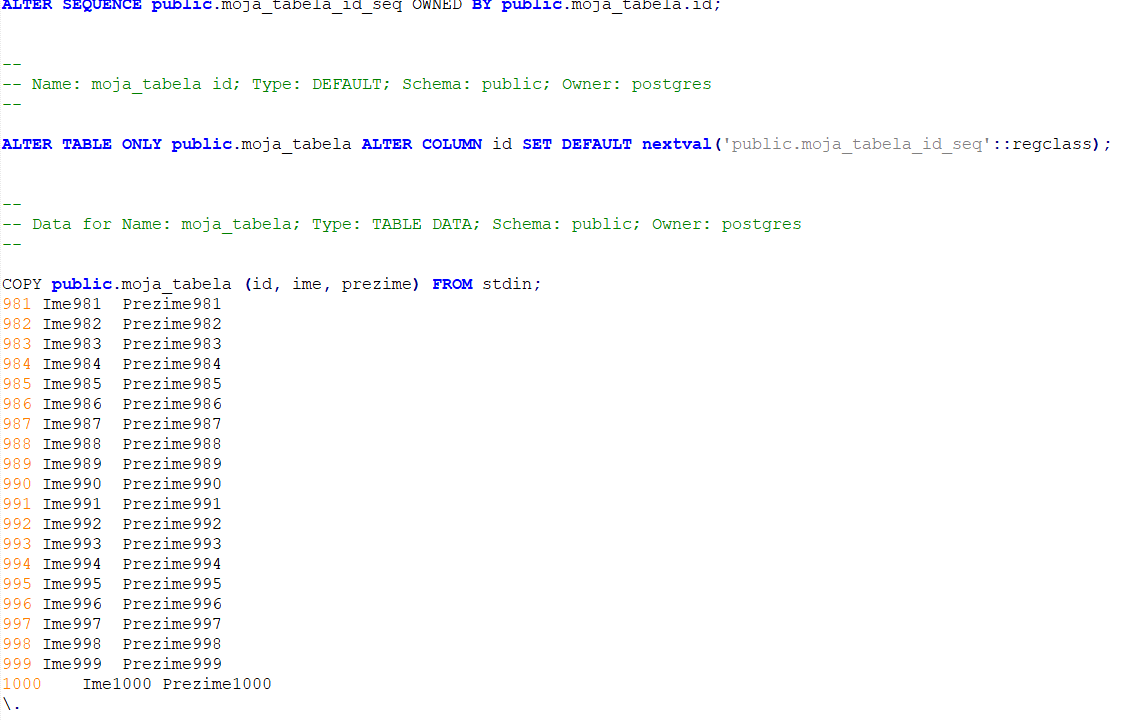
pg\_dump predstavlja klijentski program, što znači da je moguće izvršiti ovaj vid backup-ovanja sa bilo kog udaljenog računara koji ima pristup bazi podataka. Međutim, pg\_dump ne koristi posebne dozvole. Konkretno, potrebno je posedovati pristup za čitanje svih tabela koje je potrebno backup-ovati, pa ukoliko je potrebno backup-ovati celu bazu podataka, ovaj vid backup-ovanja je moguće pokrenuti jedino kao superkorisnik. Ukoliko korisnik ne poseduje dovoljno privilegija za backup-ovanje cele baze moguće je da backup-uje neke njene delove za koje ima dozvolu, recimo odgovarajuće šeme (scheme) ili tabele.



Slika 4-Prikaz kreiranja dump-a za određenu tabelu iz baze

Na slici 4 prikazano je kako je moguće kreirati backup odgovarajuće tabele iz baze “moja\_baza”. Ovaj dump fajl sadržaće SQL komande koje opisuju strukturu te tabele, uključujući definicije kolona, ograničenja, indeksa i drugih elementa koji se odnose samo na tu tabelu. Dodatno, sadržaće i SQL komande za dodavanje podataka u tu tabelu.Sadržaj ovog fajla prikazan je na slici 5.





Slika 5-Prikaz sadržaja kreiranog dump-a za tabelu “moja\_tabela”

Kako bi bilo moguće odrediti koji serverski računar pg\_dump treba da kontaktira, koriste se opcije komandne linije “-h host” i “-p port”. Kao i svaka druga PostgreSQL klijentska aplikacija, pg\_dump će se podrazumevano povezati sa korisničkim imenom baze podataka koje je jednako trenutnom korisničkom imenu operativnog sistema. Kako bi se ovo poništilo, ili se navodi opcija “-U” ili se okolina postavlja na promenljivu “PGUSER”.



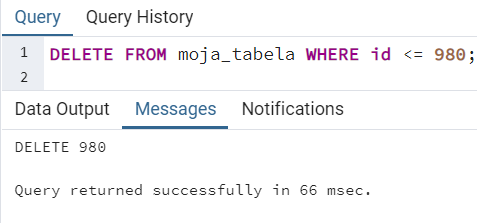
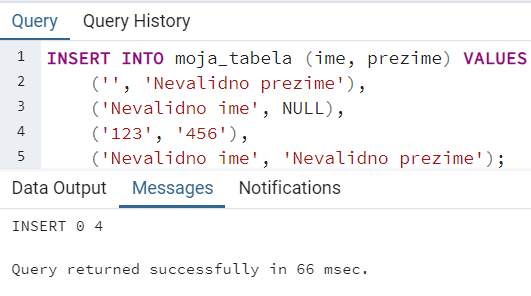
Slika 6- Prikaz kreiranja dump-a kada je potrebno specificirati serverski računar

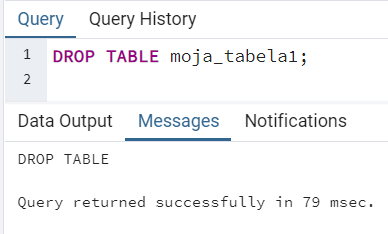
Jedna od važnih prednosti pg\_dump-a je što njegov izlaz može biti ponovo učitan u novijim verzijama PostgreSQL-a. Ovo nije slučaj sa drugim metodama backup-ovanja, kao što su backup-ovanje na nivou fajlova ili kontinuirano arhiviranje, koje su specifične za verziju servera. Takođe, ova metoda kreiranja backup-a je jedina koja će raditi kada se baza podataka premesti na drugu arhitekturu mašine, kao što je prelazak sa 32-bitnog na 64-bitni server.

Dump-ovi kreirani od strane pg\_dump-a su interno konzistentni, što znači da dump predstavlja snimak (***eng.snapshot***) baze podataka u trenutku kada je pg\_dump počeo sa radom. pg\_dump ne blokira druge operacije baze podataka dok ona radi.(Izuzeci su one operacije koje zahtevaju da se pristup bazi podataka zaključa samo za njih, kao što su većina oblika naredbe koja vrši izmenu strukture postojeće tabele u bazi podataka-ALTER TABLE).

### 3.1.1 Obnavljanje/Vraćanje (Restoring) Dump-a

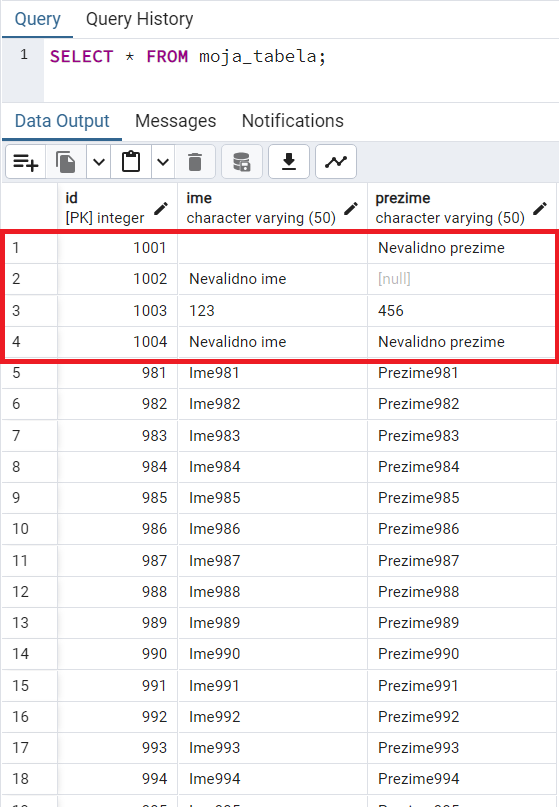
U procesu obnavljanja baze podataka, ključno je imati strategiju koja će omogućiti brzo i efikasno vraćanje podataka u funkcionalno stanje, posebno nakon otkrivanja oštećenja. Oštećenja mogu biti razna poput grešaka prilikom upisivanja ili čitanja podataka u bazu, što dovodi do nedoslednosti i neispravnosti podataka, gubitak podataka usled hardverskih kvarova ili softverskih grešaka, oštećenja tabela, indeksa i drugih objekata baze podataka ili prekidi u radu sistema. Na slici 7 prikazane su neke od mogućih grešaka u bazi podataka.



Slika 7-Prikaz jednostavnih grešaka u bazi podataka “moja\_baza”

Slika 7 prikazuje neke jednostavne greške nad tabelom “moja\_tabela” kao što su brisanje određenih podataka i unos nevalidnih vrednosti u tabelu, kao i brisanje cele tabele “moja\_tabela1”, koje su dovele do oštećenja baze podataka “moja\_baza”. Na slici 8 prikazani su podaci iz tabele “moja\_tabela” nakon oštećenja.



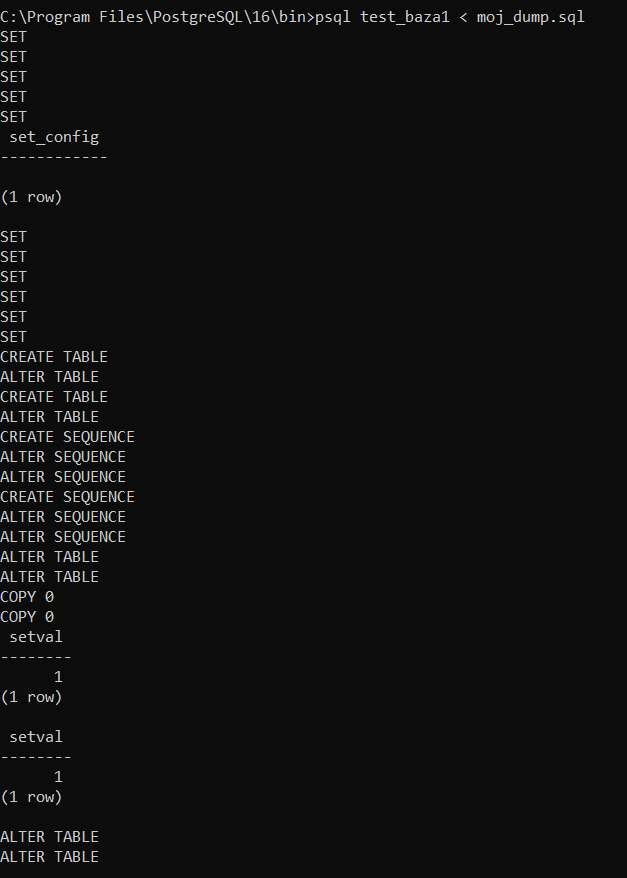
Slika 8-Prikaz podataka iz tabele “moja\_tabela” nakon oštećenja

Ova oštećenja zahtevaju brzu reakciju kako bi se sprečili dalji gubitci i očuvao integritet baze podataka. Ukoliko postoji kreirani SQL dump baze podataka koji sadrži podatke u stanju pre oštećenja, moguće je preduzeti korake za obnavljanje baze podataka. Ovaj proces obično uključuje korišćenje alata poput “psql”-a za izvršavanje SQL skripti koje su generisane dump-om. Opšti oblik komande za obnavljanje “dump”-a je:

psql ime\_baze < dump\_datoteka

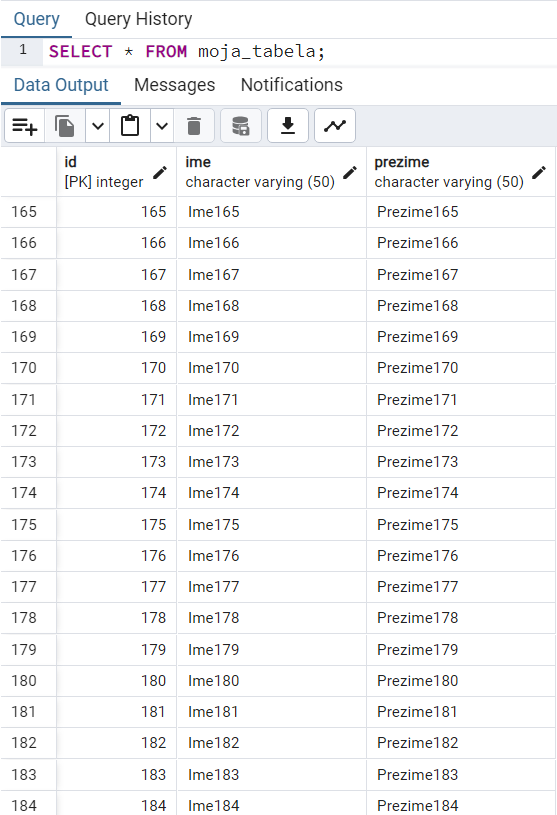
gde je dump\_datoteka datoteka koja je kreirana komandom pg\_dump. Baza podataka ime\_baze neće biti kreirana ovom komandom, već mora prethodno da bude kreirana (npr. sledećom konstrukcijom created –T template0 ime\_baze). Psql podržava opcije slične pg\_dump-u za određivanje servera baze podataka na koji se treba povezati i korisničkog imena koje treba koristiti. Dumpovi koji ne predstavljaju tekstualne datoteke obnavljaju se korišćenjem komande pg\_restore.

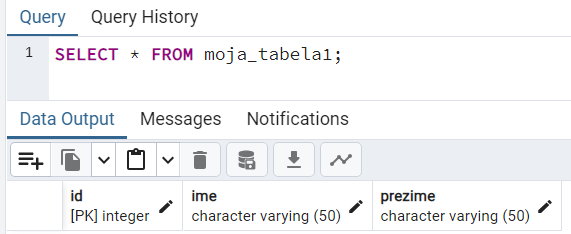
Za obnavljanje baze “moja\_baza”, korišćenjem backup-a čije je kreiranje prikazano na slici 1, dok su nastala oštećenja baze “moja\_baza” prikazana na slici 7, komanda za obnovu baze podataka bi izgledala na način na koji je prikazan na slici 9. Baza podataka “test\_baza1” kreirana je pre izvršenja komande za oporavak baze.



Slika 9-Prikaz izvršenja komande za obnavljanje baze podataka “moja\_baza”

Ovaj proces će obnoviti strukturu baze podataka i ubaciti podatke iz dump fajla. Na slici 10 prikazani su podaci baze “test\_baza1” , koji sada sadrže podatke baze “moja\_baza” iz trenutka pre nego što je nastalo oštećenje.





Slika 10-Prikaz podataka “test\_baza1”

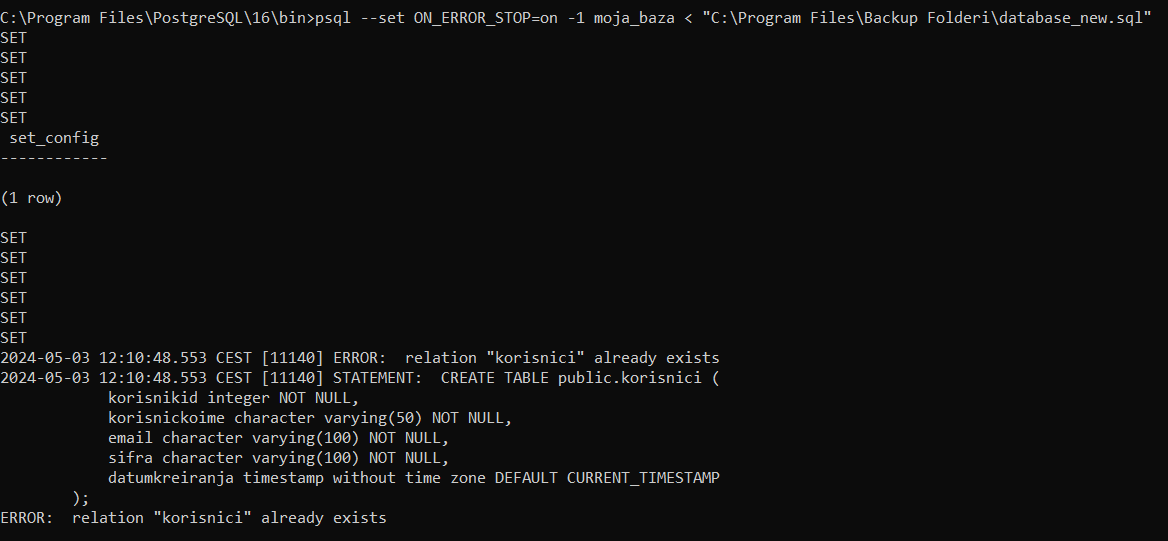
Važno je napomenuti da moraju postojati svi korisnici koji su vlasnici objekata ili kojima su dodeljene dozvole nad objektima u bazi podataka, inače proces obnavljanja može biti neuspešan u rekreiranju objekata sa originalnim vlasništvom i/ili dozvolama.

Podrazumevano, psql skripta će nastaviti sa izvršavanjem čak i nakon što naiđe na SQL grešku. Ukoliko je potrebno promeniti ovo ponašanje, moguće je pokrenuti psql sa postavljenom promenljivom ON\_ERROR\_STOP:

psql --set ON\_ERROR\_STOP=on ime\_baze < dump\_datoteka

Na taj način, psql će izaći sa izlaznim statusom 3 ako dođe do SQL greške.

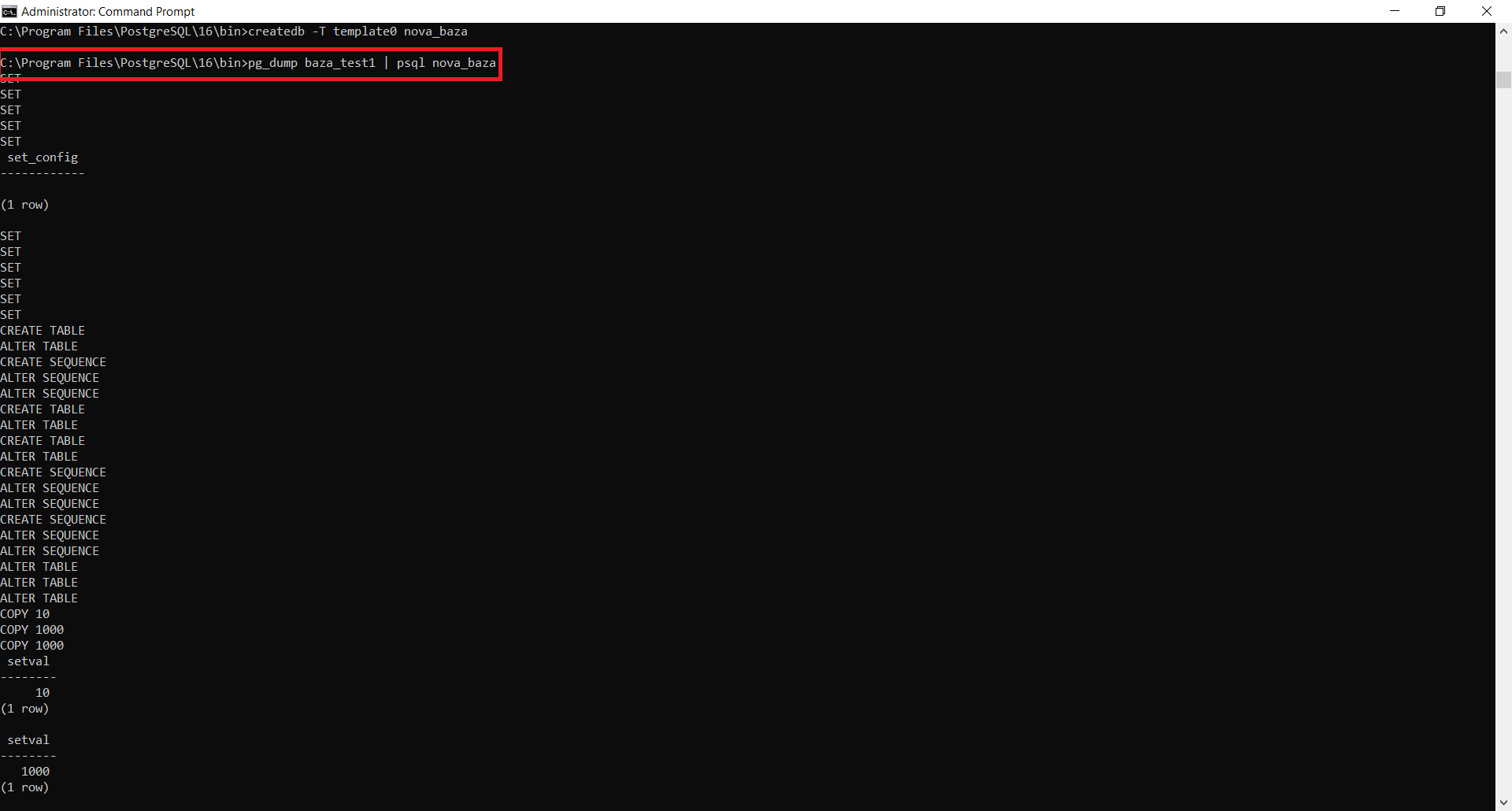
Na jedan ili drugi način, rezultat će biti samo delimično obnovljena baza podataka. Opciono, može se specificirati da ceo dump treba da se obnovi kao jedna transkacija, što znači da će obnova biti ili u potpunosti izvršena ili potpuno poništena. Ovaj režim se može aktivirati korišćenjem opcije -1 ili –single-transaction prilikom pokretanja psql-a.



Slika 11-Prikaz aktiviranja opcije ON\_ERROR\_STOP i postavljanje opcije za obnovu kao jedna transakcija

Primer koji je prikazan na slici 11 prikazuje kako se, kada je komanda ON\_ERROR\_STOP aktivna, momentalno zaustavlja izvršenje skripte nakon pojave greške. U ovom konrektnom slučaju, prilikom obnavljanja određenog SQL Dump-a u bazi podataka “moja\_baza”, psql nailazi na grešku jer pokušava da kreira tabelu “korisnici” koja već postoji u ovoj bazi. Ovime se osigurava da se izvršenje skripte ne nastavlja nakon pojave greške.

Kako pg\_dump i psql imaju sposobnost da direktno komuniciraju jedan sa drugim, moguće je izvršiti direktno prenošenje podataka između njih, što znači da je moguće “dump”-ovati bazu podataka direktno sa jednog servera na drugi.



Slika 12-Prikaz prenosa podataka između pg\_dump i psql

Primer sa slike 12 ilustruje prenos podataka između pg\_dump-a i psql-a. Najpre je izvršeno kreiranje nove baze podataka (“nova\_baza”), kako bi kasnije bio izvršen oporavak baze “baza\_test1”. Nakon toga, se najpre korišćenjem pg\_dump-a kreira siguronosna kopija baze “baza\_test1”. Korišćenjem takozvane cevi (“|”), ta sigurosnosna kopija se prenosi direktno kao ulazni tok (stdin) psql-u, koji izvršava SQL naredbe. U ovom slučaju, kopija će se koristiti za replikaciju u novu bazu podataka (“nova\_baza1”).

### 3.1.2 pg\_dumpall

pg\_dump dump-uje samo jednu bazu podataku u jednom trenutku, i pri tome ne vrši dump-ovanje informacija o ulogama ili tabelarnim prostorima, jer su oni specifični za čitav klaster umesto za svaku bazu podataka ponaosob. Da bi se izvršilo dump-ovanje celokupnog sadržaja klastera baze podataka koristi se program pg\_dumpall, koji backup-uje svaku bazu podataka u datom klasteru, a takođe čuva podatke specifične za čitav klaster, kao što su definicije uloga i tabelarnih prostora. Primer osnovne upotrebe ove komande prikazan je na slici 13.



Slika 13-Prikaz kreiranja backup-a celokupnog sadržaja klastera baze podataka.Dump fajl koji će biti kreiran kada se izvrši ova komanda imaće istu strukturu kao i dump fajl koji se dobija kreiranjem dump-a za jednu bazu (slika 3), samo će sadržati SQL komande koje opisuju strukturu svih baza podataka koje postoje.

Dobijeni dump se može obnoviti sledećom psql komandom:



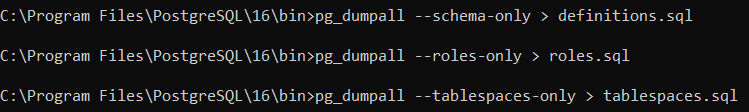
Slika 14-Obnova celokupnog sadržaja klastera

Moguće je koristiti bilo koju postojeću bazu podataka kao početak, ali ako je klaster prazan, preporučljivo je korišćenje baze podataka postgres[[1]](#footnote-1). Prilikom obnavljanja dump-a pg\_dumpall-a, uvek je potrebno imati pristup superkorisniku baze podataka, jer je to neophodno za obnovu informacija o ulogama i tabelarnim prostorima.

pg\_dumpall funkcioniše tako što emituje komande za ponovno kreiranje uloga, tabelarnih prostora i praznih baza podataka, a zatim poziva pg\_dump za svaku bazu podataka. Ovo znači da će svaka baza podataka biti interno konzistetnta, ali snimci (***eng.snapshot***) različitih baza podataka neće biti sinhronizovani.

Podaci koji su specifični za ceo klaster mogu se dump-ovati sami korišćenjem opcije pg\_dumpall –globals-only. Ovo je neophodno kako bi se u potpunosti izvršilo backup-ovanje klastera ukoliko se pg\_dump komanda izvršava nad pojedinačnim bazama podataka.

Ponekad je potrebno napraviti backup samo definicije objekata baze podataka (poput šeme, uloga i tabelarnih prostora), što omogućava obnavljanje samo određenog objekta [7]. Ovo može biti korisno u testnoj fazi, u kojoj nije potrebno zadržati stare testne podatke. Za potrebe ovoga moguće je iskoristiti pg\_dumpall na način na koji je prikazan na slici 15.



Slika 15-Kreiranje backup-a definicije objekata baze podataka korišćenjem dumpall-a

### 3.1.3 Rad sa velikim bazama podataka

Neki operativni sistemi postavljaju ograničenja na maksimalnu veličinu fajla što može predstavljati problem prilikom stvaranja velikih pg\_dump izlaznih fajlova. Međutim, kako pg\_dump vrši ispis na standardni izlaz, moguće je koristiti razne alate kako bi se ovaj problem prevazišao.Postoji nekoliko mogućih pristupa [6]:

**1.Korišćenje kompresovanog dump-a:**

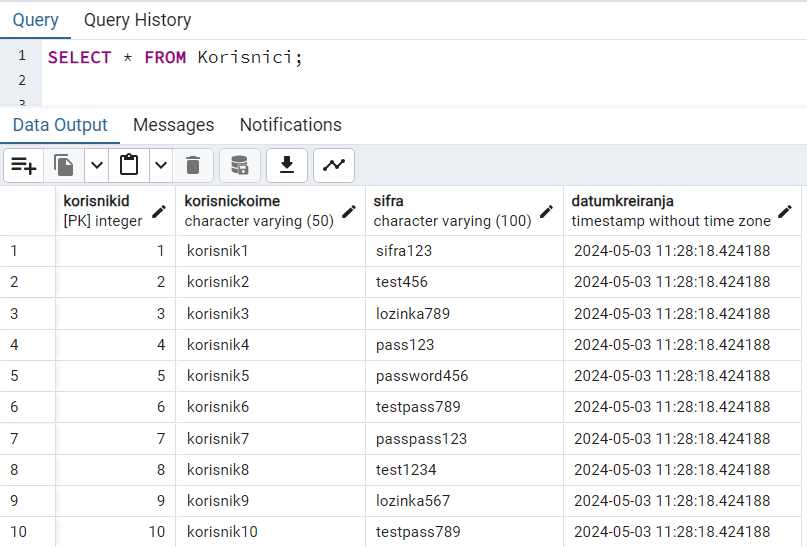
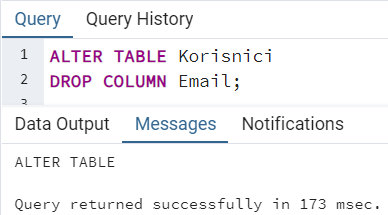
Moguće je izvršiti kompresiju dump-a korišćenjem odgovarajućeg programa [7].



Slika 16-Kreiranje kompresovanog dump-a za bazu “baza\_test”

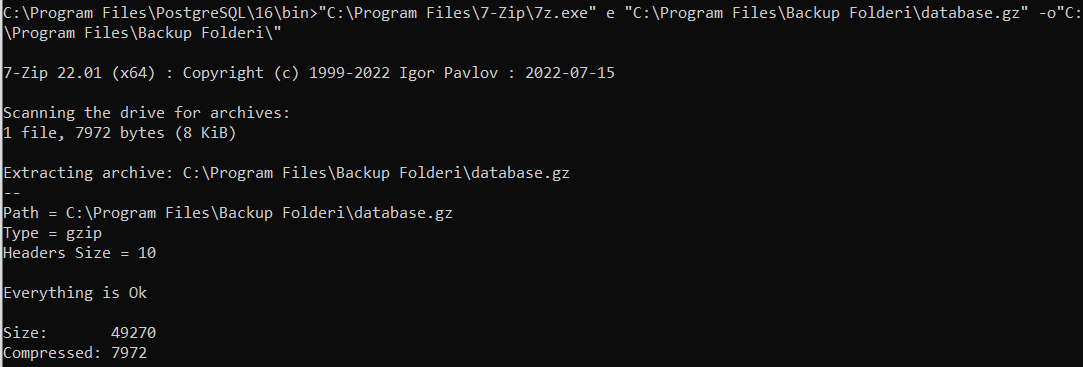
Na slici 16 prikazano je kreiranje kompresovanog dump-a korišćenjem komande Z6. Ova komanda uzrokuje da ceo izlazni fajl bude kompresovan kao da je bio prosleđen kroz program gzip sa nivoom kompresije jednakim 6(može varirati od 0(bez kompresije) do 6(maksimalna kompresija)).

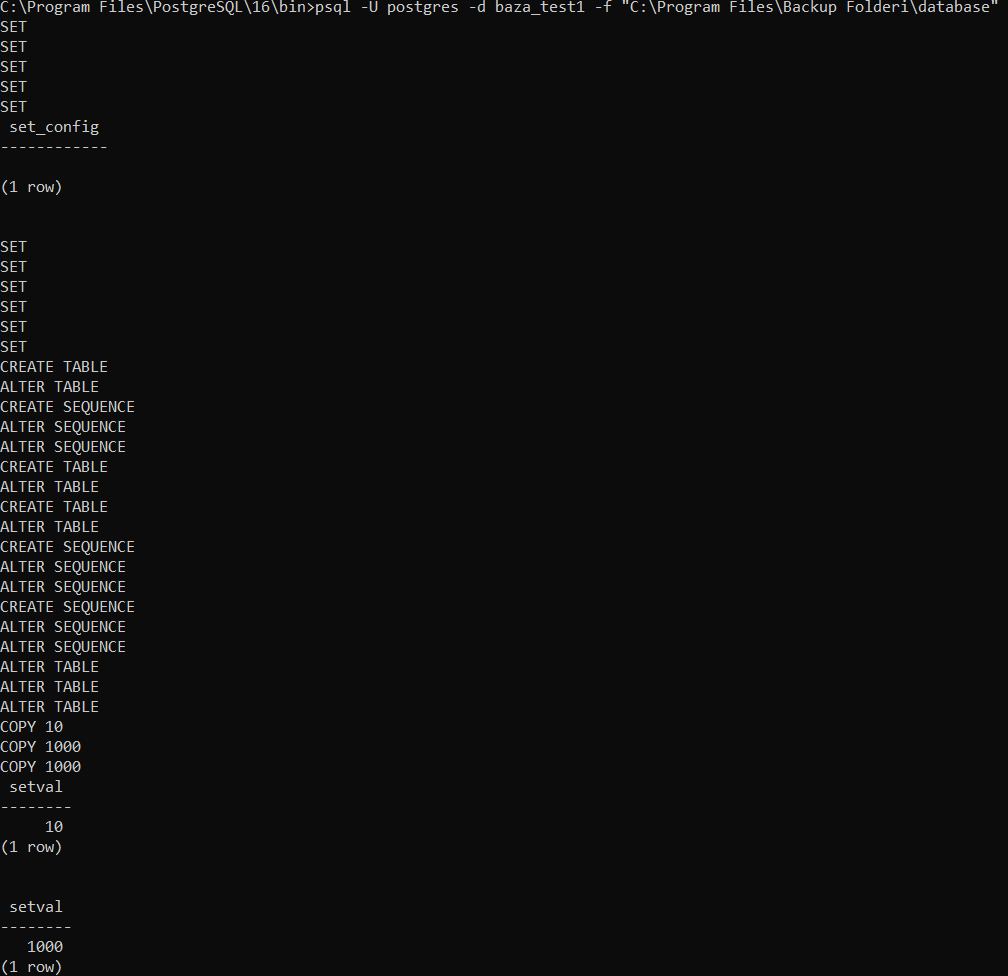
Kada dođe do nekih oštećenja u bazi (prikazano na slici 17), oporavak baze podataka u ovom slučaju moguće je izvršiti na način koji je prikazan na slici 18.

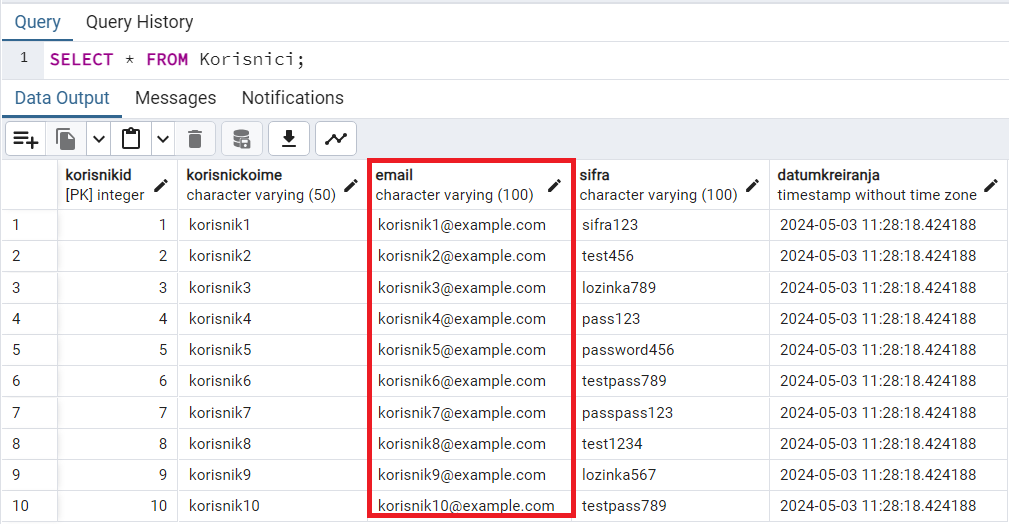


Slika 17-Prikaz simulacije oštećenja nad tabelom Korisnici i izgled tabele nakon oštećenja

Baza podataka “baza\_test” sadrži tabelu “Korisnici” koja se odnosi na određene korisnike. Ova tabela ima kolone kao što su korisničko ime, email, šifra i datum kreiranja. Ponekad može doći do brisanja nekih podataka, kao što je u ovom primeru slučaj, odnosno izbrisana je cela kolona koja sadrži email-ove korisnika. Tada je potrebno oporaviti bazu i vratiti te podatke (slika 18).







Slika 18-Prikaz oporavka kompresovanog dump-a i oporavljene tabele korisnici(crvenom linijom je označena povraćena kolona)

Sa slike 18 može se videti da je, kako se radi o kompresovanom dump-u, najpre izvršena njegova dekompresija korišćenjem programa 7-zip, a nakon toga se vrši obnova baze korišćenjem psql-a.

**2. Korišćenje custom-format arhivne datoteke**

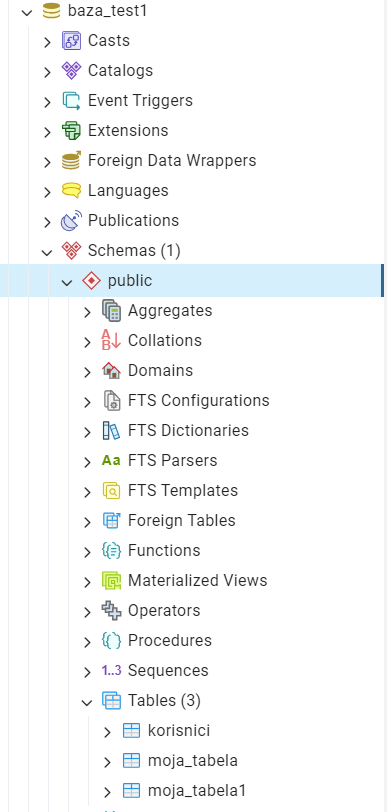
Još jedna opcija, koja rešava problem velikih izlaznih fajlova generisanih od strane pg\_dump-a, jeste korišćenje takozvanog “prilagođenog formata” (***eng.custom format***), formata koji je podrazumevano arhviran (bez dodatnih koraka) i pruža značajnu fleksibilnost, posebno prilikom uvoza [7].

Kako bi se kreirao backup u prilagođenom formatu “dump”-a koristi se opcija –Fc.



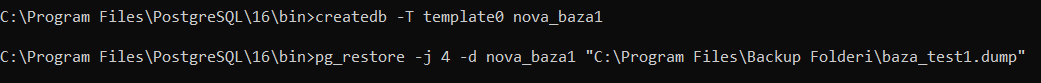
Slika 19-Kreiranje backup-a u prilagođenom (***eng.custom***) formatu

U ovom slučaju kreiran je backup nad bazom “baza\_test1” koja ima tri tabele:”korisnici”,”moja\_tabela” i “moja\_tabela1”.



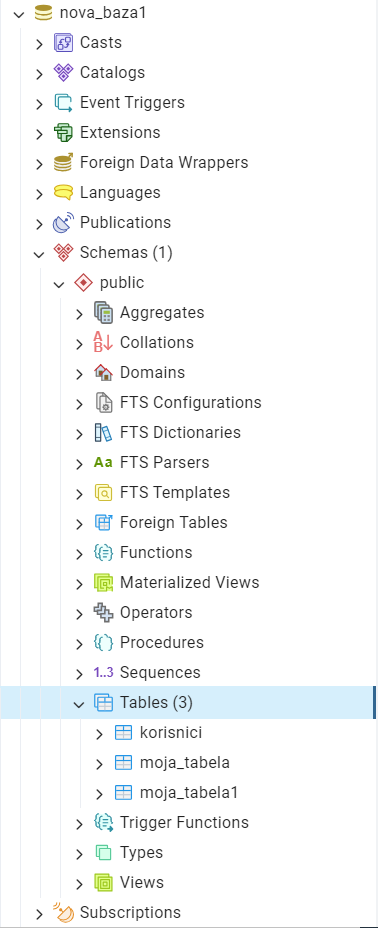
Slika 20-Prikaz strukture baze “baza\_test1”

Ovaj format datoteke koristi se sa pg\_restore komandom kako bi se ponovo izgradila baza podataka. To omogućava pg\_restore-u da bude selektivan u vezi sa tim šta se obnavlja, ili čak da preuredi stavke pre obnavljanja. Arhivirani formati datoteka namenjeni su prenosu podataka između različitih arhitektura. Međutim, vreme potrebno za obnavljanje velike baze podataka na serveru koji radi na multiprocesorskom računaru, moguće je drastično smanjiti korišćenjem opcije –j. To se postiže pokretanjem vremenski najzahtevnijih delova pg\_restore-a, kao što su učitavanje podataka, kreiranje ograničenja ili indeksa, korišćenjem više istovremenih zadataka. Svaki posao predstavlja jednu nit ili jedan proces; koristi zasebnu vezu sa serverom i zavisi od operativnog sistema. Na slici 21 prikazan je postupak obnove baze “baza\_test1”, korišćenjem napredne opcije –j. [7]



Slika 21-Prikaz obnove baze “baza\_test1” korišćenjem opcije –j. Opcija –j 4 označava da će se proces obnavljanja izvšiti korišćenjem 4 paralelne niti(ili procesa), što će značajno ubrzati vreme obnavljanja u slučaju velikih baza podataka.

Sa slike (slika 21) može se videti da je najpre korišćenjem konstrukcije created –T template0 nova\_baza1 kreirana nova baza podataka, a nakon toga je izvršena obnova baze “baza\_test1”. Novokreirana baza (nova\_baza1) imaće istu strukturu kao i “baza\_test1”(Slika 20).

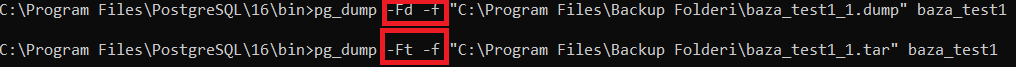


Slika 22- Prikaz strukture baze “nova\_baza1”

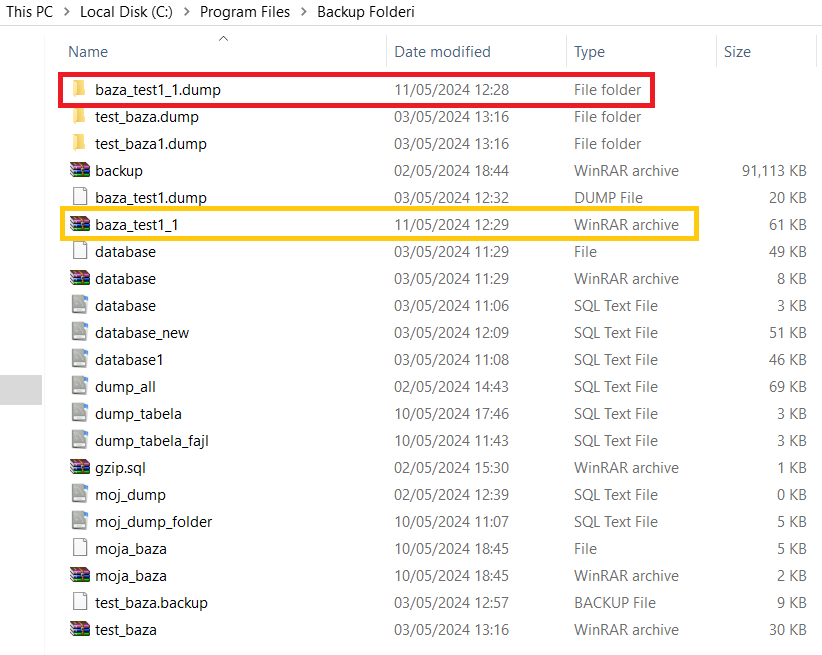
Pored “custom” formata, pg\_dump podržava još dva formata izlaznih datoteka: direktorijumski i tar [7].Oba formata se obnavljaju korišćenjem alata pg\_restore.

Za kreiranje arhive u direktorijumskom formatu koristi se opcija –Fd. Prilikon kreiranja arhive u ovakvom formatu, svaka tabela i “izbačeni blob”[[2]](#footnote-2)(***eng.dumped blob***) će biti smešteni u posebnu datoteku unutar novog direktorijuma.Osim toga, biće generisana i tabela sadržaja datoteka koja opisuje izbačene objekte u formatu koji je pogodan za mašinsko učenje, omogućavajući pg\_restore-u da ih efikasno pročita. Direktorijumski format omogućava upotrebu standardnih Unix alata za manipulaciju arhivom, kao što je gzip koji se koristi za kompresiju datoteka unutar nekompresovane arhive. Ovaj format podržava i paralelne dump-ove sa podrazumevanom kompresijom datoteka.

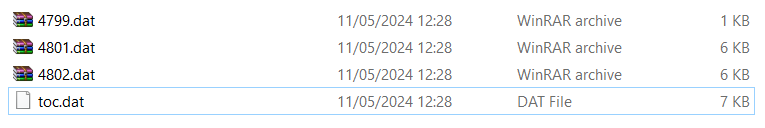
Tar format je kompatabilan sa direktorijumskim formatom: važeća arhiva u direktorijumskom formatu nastaje kada se ekstrakuje arhiva u tar formatu.Međutim, tar format ne podržava kompresiju.Pored toga prilikom korišćenja tar formata, relativni redosled elemenata podataka tabela ne može se promeniti tokom procesa obnavljanja. Za kreiranje tar datoteke koristi se opcija –Ft.

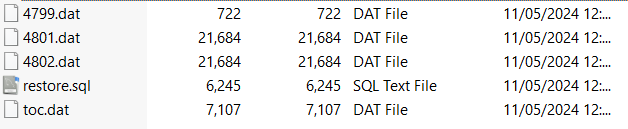


Slika 23-Kreiranje direktorijumske i tar datoteke



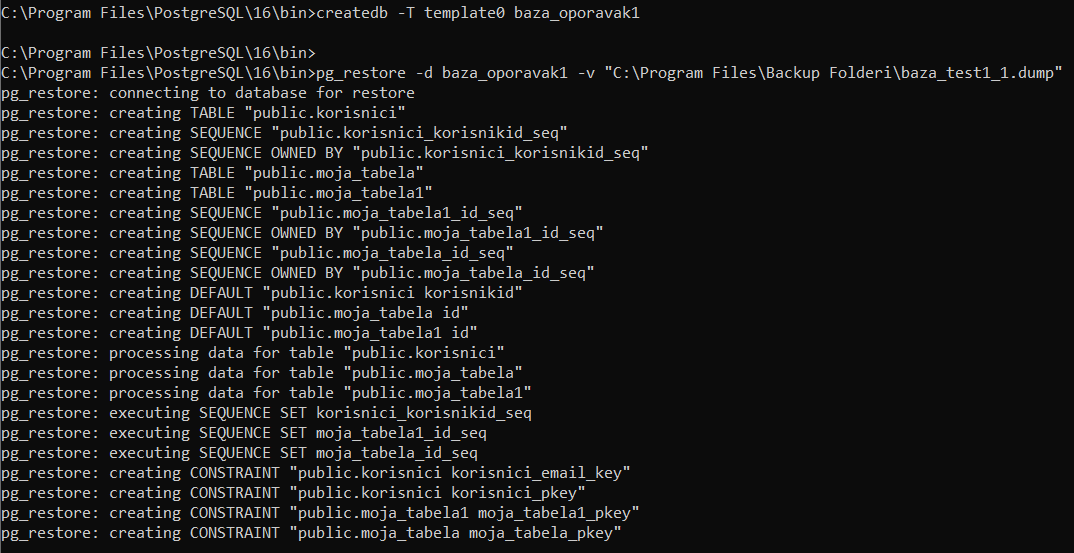
Slika 24-Prikaz kreiranih direktorijumskih (crvena linija) i tar (narandžasta linija) datoteka na lokaciji "C:\Program Files\Backup Folderi\”

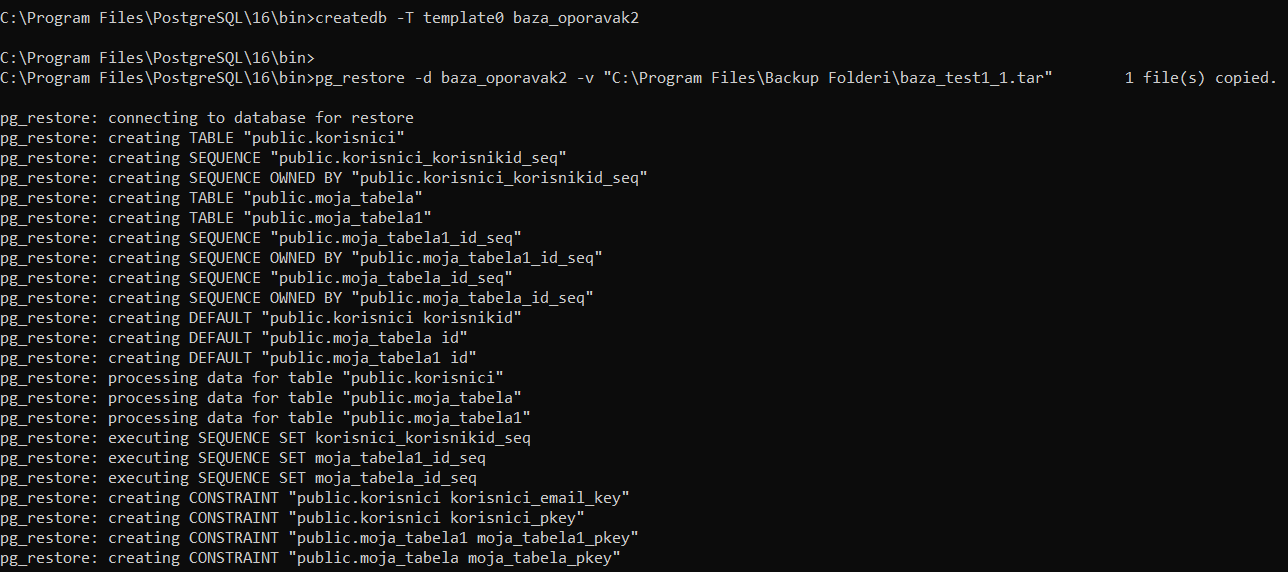




Slika 25-Prikaz sadržaja direktorijumske (slika gore) i tar (slika dole) datoteke.Unutar ovih datoteka nalaze se fajlovi koji sadrže SQL komande koje definišu strukturu baze podataka (u ovom slučaju baze “baza\_test1”)

Oporavak ovakve vrste backup-ova može se izvršiti na sledeći način:



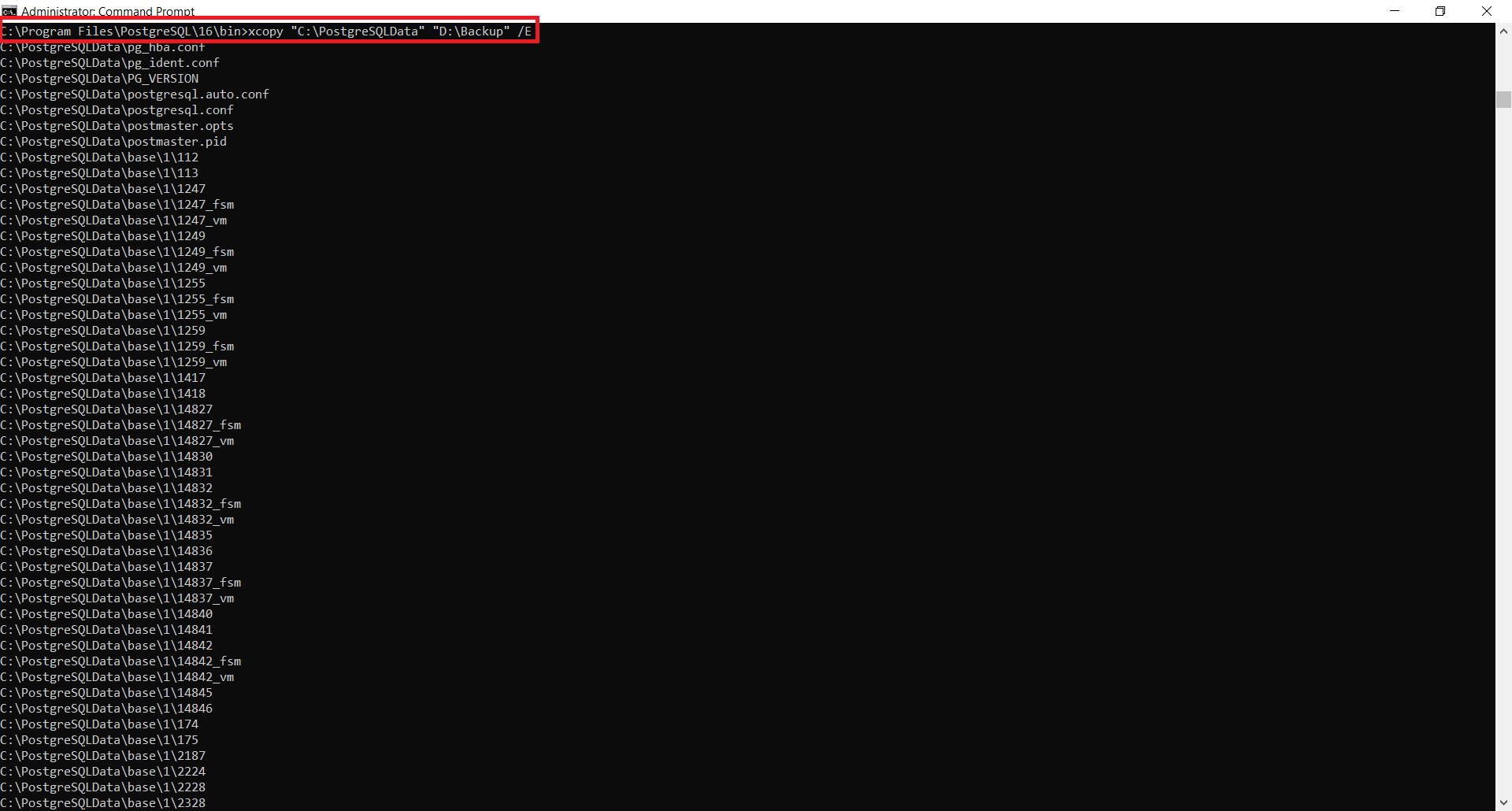


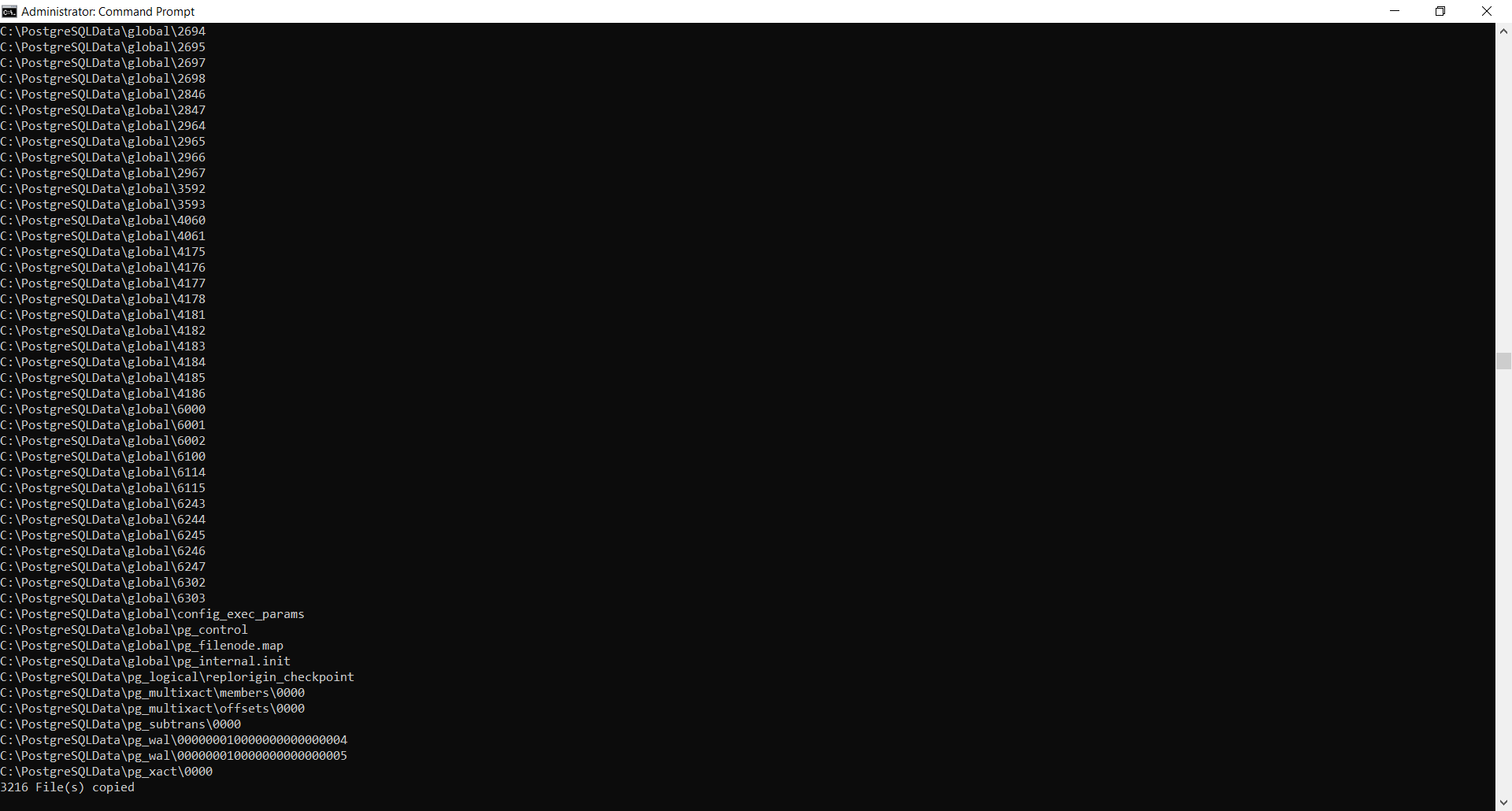
Slika 26-Prikaz obnavljanja baza podataka korišćenjem direktorijumske datoteke(slika gore) i tar datoteke(slika dole)

Sa slike(slika 26) se može videti da se prilikom oporavka u oba slučaja najpre kreira šema “public”, a zatim se postepeno obnavljaju objekti poput tabela, sekvenca i ograničenja za svaku tabelu, što omogućava povratak baze podataka na prethodno definisano stanje.

## 3.2 Backup na nivou fajl sistema

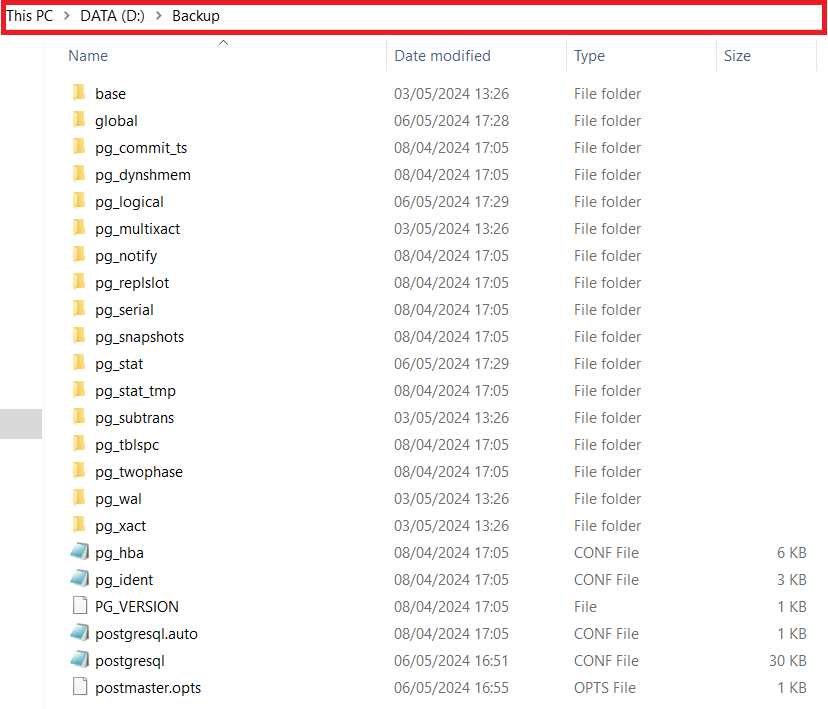
U prethodnom poglavlju je objašnjeno i prikazano kako je moguće kreirati backup baze podataka korišćenjem SQL Dump-a. Međutim, ponekad je korisno kreirati backup fajlova koje PostgreSQL koristi za skladištenje podataka u bazi podataka. Alternativna strategija backup-ovanja koja ovo omogućava jeste direktno kopiranje ovih fajlova na neku drugu lokaciju, poznato još i kao backup-ovanje na nivou fajl sistema [8]. Za ove potrebe moguće je koristiti bilo koji metod koji omogućava pravljanje siguronosne kopije fajl sistema, kako je i prikazano na slici 27[7].



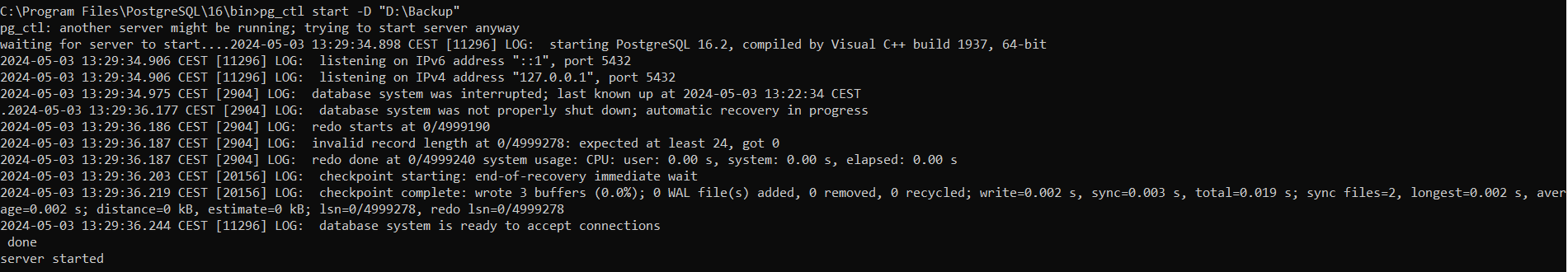


Slika 27-Kreiranje backup-a na nivou fajl sistema

Inicijalno su podaci PostgreSQL baze podataka sladišteni na lokaciji “C:\PostgreSQLData”, ali su komandom xcopy premešteni na lokaciju “D:\Backup”. U ovom folderu smešteni su sada svi folder i direktorijumi koji čine osnovnu strukturu podataka za PostgreSQL bazu podataka, kao što su podaci o samim tabelama, indeksima, transakcijama, logovima i drugim važnim informacijama, na identičan način na koji su bili i na lokaciji “C:\PostgreSQLData”. Sada je moguće server pokrenuti i sa nove lokacije.



Slika 28- Siguronosna kopija na nivou fajl sistema na lokaciji “D:\Backup”



Slika 29-Pokretanje servera sa nove lokacije

Ipak, postoje dva ograničenja koja čine ovu metodu nepraktičnom ili inferiornom u odnosu pg\_dump [8]:

1. **Neophodno je isključiti server baze podataka:** Kako bi kreirani backup bio upotrebljiv, potrebno je potpuno isključiti server baze podataka. Polovični pristupi poput zabrane svih konekcija neće dati željene rezultate. Ovo se delimično dešava zbog toga što alati poput tar-a ne prave atomične snimke stanja fajl sistema, i zbog internog keširanja koje se dešava unutar samog servera. Postoji nekoliko načina na koje je moguće zaustaviti server: **SIGTERM** ili pametno gašenje, pri kome server onemogućava nove konekcija, ali čeka da postojeće sesije završe pre gašenja, **SIGINT** ili brzo gašenje, pri kome server prekida sve sesije i zatvara se i **SIGQUIT**, ili neposredno gašenje, pri kome server šalje signale podprocesima, čekajući njihovo završavanje, ali u hitnim slučajevima može poslati **SIGKILL** za brzo gašenje. Ove signale je moguće poslati korišćenjem pg\_ctl programa ili komande kill na Unix/Linux sistemima. Takođe, server je moguće zaustaviti i komandom: pg\_ctl stop -D /\*PostgreSQL data\*/.
2. **Nemogućnost selektivne obnove**: Nemoguće je napraviti siguronosnu kopiju ili obnoviti samo određenu tabelu ili bazu podataka iz njihovog fajla ili direktorijuma jer informacije koje su sadržane u tim fajlovima nisu korisne bez pripadajućih fajlova dnevnika transakcija (pg\_xact/\*), koji beleže status svih transakcija. Fajl tabele je koristan samo uz ove informacije. Pokušaj obnove samo jedne tabele i pripadajućih pg\_xact podataka doveo bi do neupotrebljivosti svih ostalih baza podataka u klasteru. Zbog toga su siguronosne kopije fajl sistema moguće samo za kompletnu siguronosnu kopiju i obnovu čitavog klastera baze podataka.

Alternativni pristup siguronosnoj kopiji fajl sistema podrazumeva kreiranje “konzistentnog snimka” (***eng.consistent snapshot***) direktorijuma sa podacima, ukoliko fajl sistem podržava tu funkcionalnost. Uobičajan postupak podrazumeva stvaranje takozvanog “zamrznutog snimka” (***eng.frozen snapshot***) prostora gde se čuvaju podaci baze. Nakon toga se cela struktura podataka kopira sa tog snimka na uređaj za siguronosno kopiranje. Nakon završetka ovog procesa, “zamrznuti snimak” (***eng.frozen snapshot***) se oslobađa. Ova metoda funkcioniše čak i dok je server baze podataka aktivan, ali backup koji je napravljen ovim putem čuva fajlove baze podataka u stanju kao da server nije ispravno isključen. Kada server bude ponovo uključen, on će smatrati da je prethodna instanca servera “pala” i reizvešće WAL log[[3]](#footnote-3). Ipak, ovo ne predstavlja problem, samo zahteva da WAL log datoteka bude uključena u siguronosnu kopiju. Takođe, pre pravljenja snimka (***eng.snapshot***), potrebno je izvršiti CHECKPOINT[[4]](#footnote-4) kako bi se smanjilo vreme oporavka. Međutim, ukoliko se baza podataka nalazi na više fajl sistema, nemoguće je dobiti potpuno sinhronizovane “zamrznute snimke” (***eng.frozen snapshot***) svih tih prostora. Recimo, ako se datoteke sa podacima i WAL logovi nalaze na različitim diskovima, ili ukoliko se tabelarni podaci nalaze na različitim fajl sistemima, neće biti moguće koristiti siguronosno kopiranje snimka (***eng.snapshot***), jer svi snimci moraju biti istovremeni. [8]

Ako nije moguće postići sinhronizovane snimke (***eng. simultaneous snapshots***), jedna opcija jeste privremeno isključivanje servera baze podataka dok se vrši kreiranje zamrznutog snimka (***eng.frozen snapshot***). Druga opcija jeste korišćenje kontinuranog arhiviranja baze podataka (biće opisano u potpoglavlju 3.3) jer su takve kopije otporne na promene u fajl sistemu tokom procesa siguronosng kopiranja. Za to je potrebno omogućiti kontinuirano arhiviranje samo tokom procesa siguronosng kopiranja, a obnavljanje se vrši korišćenjem kontinuiranog oporavka iz arhive.

Još jedna opcija jeste korišćenje rsync alata za obnavljanje sigurosnosne kopije fajl sistema. To se postiže prvo pokretanjem rsync-a dok je server baze podataka još uvek aktivan, a zatim se server isključuje dovoljno dugo da se izvrši rsync –checksum. Drugi rsync će biti brži od prvog, jer ima relativno malo podataka za prenos, a krajni rezultat će biti konzistentan jer je server bio isključen. Ovaj metod omogućava obnavljanje siguronosne kopije fajl sistema sa minimalnim prekidom rada. [8]

Važno je napomenuti da će kreirani backup fajl sistema biti veći od SQL dump-a, jer pg\_dump ne mora da dump-uje sadržaj indeksa, već samo naredbe za njihovo ponovno kreiranje, ali obnavljanje backup-a fajl sistema može biti brže.

## 3.3 Kontinuirano arhiviranje i obnova do trenutka (PITR)

PostgreSQL održava dnevnik pisanja (***eng.Write Ahead Log-WAL***) koji beleži svaku promenu napravljenu u datotekama podataka baze. Ovaj dnevnik se čuva u poddirektorijumu pg\_wal/ unutar direktorijuma podataka klastera. Glavni razlog za postojanje WAL-a jeste osiguranje stabilnosti u slučaju pada sistema: ako dođe do pada, baza podataka može biti vraćena na stabilno stanje “reprodukcijom” (***eng.replay***) unosa dnevnika (***eng.log entries***) koji su napravljeni od poslednje kontrolne tačke (***eng.checkpoint***). Pored toga, postojanje dnevnika omogućava upotrebu treće strategije za pravljenje rezervnih kopija (***eng.backup***) baze podataka: moguće je izvršiti kombinaciju backup-a na nivou fajl sistema sa backup-ovanjem WAL datoteka. Prilikom obnavljanja (***eng.restore***), prvo se vraća rezervna kopija (backup) sistema datoteka, a zatim se vrši “reprodukcija” (***eng.replay***) unosa iz sačuvanih WAL datoteka kako bi se sistem doveo do trenutnog stanja. Iz administrativnog ugla, ovaj pristup je komplikovaniji u odnosu na prethodni, ali donosi nekoliko značajnih prednosti [9]:

* Nije neophodno posedovati potpuno usklađenu rezervnu kopiju (backup) sistema datoteka kao osnovu. Eventualne unutrašnje razlike biće ispravljene “reprodukcijom” (***eng.replay***) dnevnika (log), tako da nije neophodno "hvatati” trenutno stanje sistema datoteka prilikom arhviranja.
* S obzirom na to da je moguće izvršiti spajanje neograničenog broja WAL datoteka za “reprodukciju”(***eng.replay***), kontinuirani backup (rezervna kopija) se može ostvariti jednostavno nastavljajući sa arhiviranjem WAL datoteka. Ovo je posebno korisno za velike baze podataka, gde možda nije praktično često praviti potpunu rezervnu kopiju (potpuni backup).
* Nije obavezno “reprodukovati” (***eng.replay***) sve WAL zapise do kraja. Moguće je prekinuti “reprodukciju” (***eng.replay***) kada god je to poželjno i imati konzistentan snimak (***eng.snapshot***) baze podataka u tom trenutku. Zato ova tehnika podržava oporavak u određenom trenutku: moguće je vratiti bazu podataka u stanje u kojem je bila u bilo kom trenutku od trenutka kada je napravljena osnovna rezervna kopija (osnovni backup).
* Ako se vrši kontinuirani prenos niza WAL datoteka na drugu mašinu koja ima istu osnovnu rezervnu kopiju (početni backup), kreira se takozvani sistem “tople rezerve”: u svakom trenutku je moguće pokrenuti tu drugu mašinu i biće dostupna gotovo trenutna kopija baze podataka.

Važno je napomenuti, da se kreiranjem SQL Dump-a (koji je opisan u potpoglavlju 3.1) ne proizvode rezervne kopije na nivou sistema datoteka i da se samim tim, ovaj pristup ne može koristiti kao deo kontinuiranog arhiviranja. Ovakvi backup-ovi predstavljaju logičke backup-ove (videti poglavlje 2), i ne sadrže dovoljno informacija koje bi se koristile od strane WAL “reprodukcije” (***eng.replay***).

Kao i kod običnog backup-ovanja sistema datoteka, ovaj pristup omogućava samo obnovu celokupnog klastera baze podataka, a ne i manjih delova. Osim toga, zahteva mnogo prostora za arhiviranje: osnovna kopija može biti jako velika, a zauzet (aktivan-busy) sistem će generisati velike količine podataka u obliku WAL datoteka koje je potrebno arhivirati. Ipak, ovo predstavlja preferiranu tehniku kreiranja backup-a u mnogim situacijama gde je pouzdanost od suštinskoh značaja.

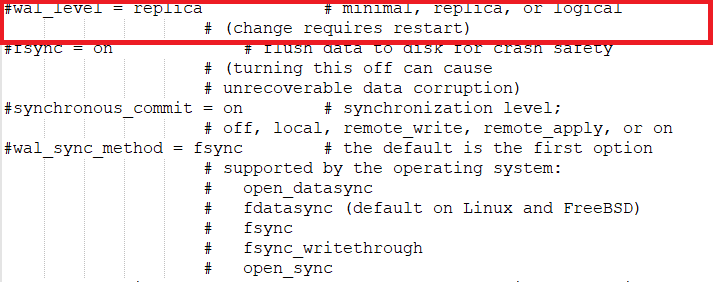
Da bi se uspešno obavila obnova korišćenjem kontinuiranog arhiviranja (takođe poznatog kao “online backup”), neophodna je neprekidna serija arhiviranih WAL datoteka koja seže najmanje do vremena početka rezervne kopije. Stoga je važno najpre postaviti i testirati postupak arhiviranja WAL datoteka. Postupak arhiviranja biće opisan i prikazan u nastavku potpoglavlja (sekcija 3.3.1).

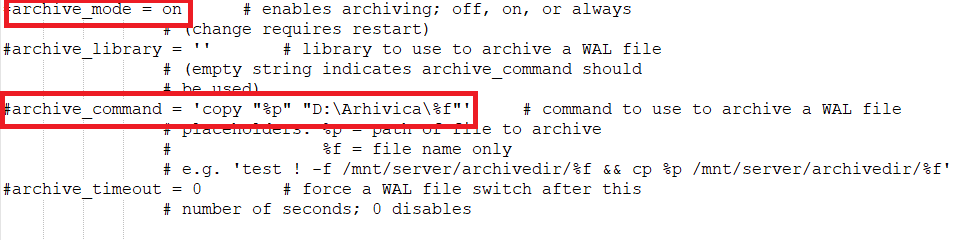
### 3.3.1 Postavljanje WAL arhiviranja

Prilikom pokretanja PostgreSQL sistema, generiše se niz zapisa u dnevniku pisanja unapred (WAL). Ovaj niz se fizički deli na segmentne datoteke dnevnika, koje obično imaju veličinu od 16MB (mada se veličina segmenta može promeniti tokom inicijalizacije baze). Svaka segmentna datoteka dobija numeričko ime koje odražava njen položaj u apstraktnom nizu dnevnika. Kada se ne koristi arhiviranje WAL podataka, sistem obično kreira samo nekoliko segmentnih datoteka, a zatim ih “reciklira” tako što preimenjuje segmentne datoteke koje više nisu potrebne u segmente sa većim brojevima. Smatra se da segmentne datoteke čiji sadržaj prethodi poslednjem checkpoint-u nisu od interesa (njihov sadržaj je već saćuvan u bazi podataka) i mogu biti bezbedno “reciliklirane”. [9]

Kada se arhiviraju WAL podaci, neophodno je sačuvati sadržaj svake segmentne datoteke kada se popuni, pre nego što se ta datoteka ponovo koristi. Postoji mnogo različitih načina čuvanja ovih podataka, u zavisnosti od aplikacije i dostupne hardverske opreme. Na primer, segmentne datoteke se mogu kopirati u direktorijum na drugoj mašini putem NFS-a[[5]](#footnote-5), mogu se zapisivati na tape drive-u[[6]](#footnote-6)(pri čemu je potrebno obezbediti identifikaciju originalnog imena svake datoteke), grupisati i snimiti na CD-ove ili je moguće primeniti neku drugu metodu. Kako bi pružio fleksibilnost administratoru baze podataka, PostgreSQL ne pravi pretpostavke o načinu arhiviranja, već administrator može specificirati shell komandu ili arhivsku biblioteku koja će biti izvršena kako bi se kopirala završena segmentna datoteka na odredište. Ovo može biti jednostvna shell komanda poput cp, ili složena C funckija.

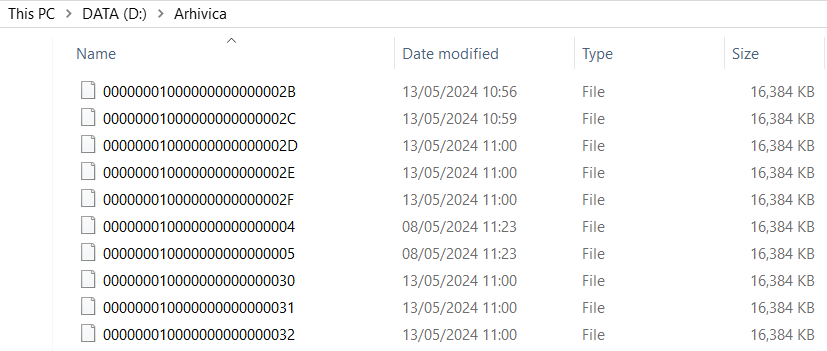
Kako bi bilo moguće koristiti arhiviranje WAL-a, potrebno je u modifikovati konfiguracioni fajl baze podataka (postgresql.conf) postavljanjem parametara “wal\_level”, koji određuje koliko se informacija unosi u wal logove, na vrednost “replica”, koja upisuje dovoljno podataka da bi arhviranje bilo podržano, i parametra “archive\_mode” na vrednost “on”, što omogućava da se vrši arhiviranje WAL segmenata. Dodatno, potrebno je specificirati shell komandu koja će se koristiti u konfiguracionom parametru archive\_command, koja određuje kako se završeni WAL segmenti šalju na arhiviranja, ili koristiti odgovarajuću biblioteku u parametru archive\_library, takođe u konfiguracionom fajlu(slika 30). U archive\_command parametru, placeholder %p će biti zamenjen putanjom datoteke za arhiviranje, dok će se placeholder %f zameniti samo imenom datoteke.Bitno je napomenuti da je putanja relativna u odnosu na trenutni radni direktorijum, odnosno direktorijum podataka klastera.Komanda koja ovo omogućava prikazana je slici 30.





Slika 30-Prikaz postavljanja odgovarajućih parametara kako bi se omogućilo WAL arhiviranje

Na slici 30 prikazano je kako se postavljaju parametri u konfiguracionim falju na odgovarajuće vrednosti kako bi bilo moguće izvršiti WAL arhiviranje. Parametar archive\_command je postavljen na vrednost 'copy "%p" "D:\Arhivica\%f" ', što znači da će se datoteka za arhiviranje koja je zamenjena placeholder-om %p (uglavnom ovo predstavlja pg\_wal datoteku koja se nalazi u direktorijumu gde su smešteni podaci PostgreSQL baze podataka) kopirati u datoteku na lokaciji “D:\Arhivica\%f” (prikazano na slici 31).



Slika 31-Prikaz direktorijuma u kome su kopirane WAL datoteke.

Datoteka za arhiviranje (D:\Arhivica) sada sadrži WAL datoteke iz pg\_wal direktorijuma (kako je i prikazano na slici 31), koji predstavljaju binarne zapise transakcija koje su izvršene nad bazom podataka. Ime svake WAL datoteke sastoji se od nekoliko delova, uključujući identifikator vremenske linije (timelineid, biće detaljnije opisani u sekciji 3.3.5), broj WAL datoteke i heksadecimalnu reprezentaciju broja segmentne datoteke. Na primer, format imena WAL fajla može biti '00000001000000000000002B', gde '00000001' predstavlja identifikator vremenske linije, '00000000' je broj WAL datoteke, a '0000002B' je broj segmenta u heksadecimalnom formatu.

Komanda za arhiviranje treba da se izvrši pod istim vlasništvom kao i PostgreSQL server. Pošto serija arhiviranih WAL datoteka sadrži sve promene u bazi podataka, od suštinskog je značaja zaštiti ove arhivirane podatke od neovlašćenog pristupa. Ukoliko je uspešna, komanda za arhiviranje vraća nulti status, koji PostgreSQL-u signalizira da je datoteka arhivirana i da je može ukloniti ili “reciklirati”. Međutim, ako komanda ne vrati nulti status, PostgreSQL će periodično pokušavati ponovo dok ne uspe.

Još jedan način arhvirianja jeste korišćenje prilagođenog modula za arhiviranje kao archive\_library. Ovi moduli, koji su obično napisani u programskom jeziku C, omogućavaju naprednije funkcionalnosti od arhiviranja putem shell komandi. Iako kreiranje sopstvenog modula može zahtevati više truda nego pisanje shell komande, moduli za arhiviranje mogu biti znatno efikasniji i imati pristup mnogim korisnim resursima servera. Prilikom korišćenja prilagođenih modula za arhiviranje, važno je osigurati da arhivske komande i datoteka budu dizajnirane na način koji sprečava slučajno prepisivanje već postojećih arhivskih datoteka. Ovo je bitna siguronosna mera koja osigurava integritet arhive, posebno u situacijama kada više servera može pristupiti istom arhivskom direktorijumu. Stoga je neophodno temeljno testirati prilagođeni modul za arhiviranje kako bi se utvrdilo da li ispravno funkcioniše i da li zadovoljava siguronosne standarde. [9]

U retkim slučajevima, PostgreSQL može pokušati ponovno arhiviranje već arhivirane WAL datoteke. Ovo se može dogoditi na primer, ako sistem padne pre nego što server zabeleži trajni zapis o uspehu arhiviranja. Nakon ponovnog pokretanja servera, on će pokušati ponovno arhiviranje datoteke. Kada arhivska komanda ili biblioteka naiđe na postojeću datoteku, trebalo bi da vrati nulti status izlaza. Ovo bi značilo da WAL datoteka ima identičan sadržaj kao prethodna arhiva i da je prethodna arhiva potpuno perzistentna na skladištu. Ako prethodna datoteka sadrži različit sadržaj od WAL datoteke koja se arhivira, arhivska komanda ili biblioteka mora vratiti nenulti status izlaza.Brzina arhivske komande ili biblioteke nije presudna sve dok može da drži korak sa prosečnom brzinom kojom server generiše WAL podatke. Normalan rad se nastavlja čak i ako proces arhiviranja malo zaostaje. Međutim, ukoliko arhiviranje značajno zaostaje, povećaće se količina podataka koja bi se izgubila u slučaju katastrofe. Takođe, to znači da će direktorijum koji čuva logove (pg\_wal) sadržati veliki broj segmentnih datoteka koje još uvek nisu arhivirane, što bi na kraju moglo da premaši dostupan prostor na disku. Iz svih ovih razloga, potrebno je redovno pratiti proces arhiviranja.

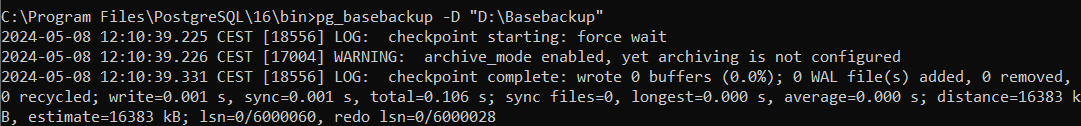
Još je važno napomenuti, da iako arhiviranje WAL-a omogućava vraćanje svih izmena koje su napravljene u podacima u PostgreSQL bazi podataka, neće vratiti promene koje su napravljene u konfiguracionim datotekama (postgresql.conf, pg\_hba.conf i pg\_ident.conf), pošto se one uređuju ručno, a ne putem SQL operacija. Zato je potrebno smestiti ove datoteke na lokaciju koja će redovno biti backup-ovana korišćenjem backup-a na nivou fajl sistema.

### 3.3.2 Pravljenje osnovnog backup-a(base backup)

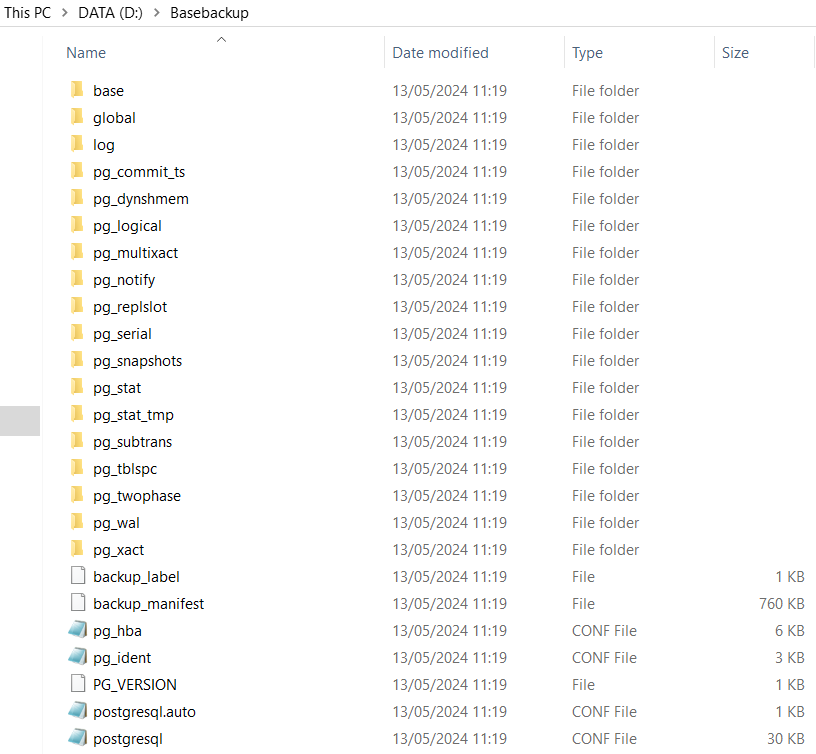
Backup baze podataka, se kod relacionih baza podatala kako je već ranije opisano, može grubo podeliti u dve kategorije: logički i fizički backup-ovi. Obe kategorija backup-ova imaju svoje prednosti i mane, a jedna od mana logičkog backup-a, koji je opisan u potpoglavlju 3.1(SQL Dump), jeste što mogu biti vremenski jako zahtevni. Konkretno, kreiranje logičkog backup-a, kao i sam proces oporavka velike baze podataka, može potrajati jako dugo. S druge strane, fizički backup-ovi se mogu napraviti i oporaviti znatno brže, sto ih čini veoma važnom i korisnom funkcijom u praktičnim sistemima. Snimak čitavog klastera baze podataka (tj.podaci fizičkog backup-a) poznat je pod nazivom osnovni backup (***eng.base backup***) [10].

Nakon što se omogući WAL arhiviranjen kreira se osnovni backup. Najjednostavniji način za kreiranje osnovnog bakup-a jeste korišćenje pg\_basebackup alata. Ovaj alat kreira osnovni backup ili u vidu obične datoteke ili u vidu tar arhive. Ipak, ukoliko je potrebno postići veću fleksibilnost od one koju pruža pg\_basebackup, osnovni backup je moguće kreirati korišćenjem API-ja nižeg nivoa (biće opisano u potpoglavlju 3.3.3).

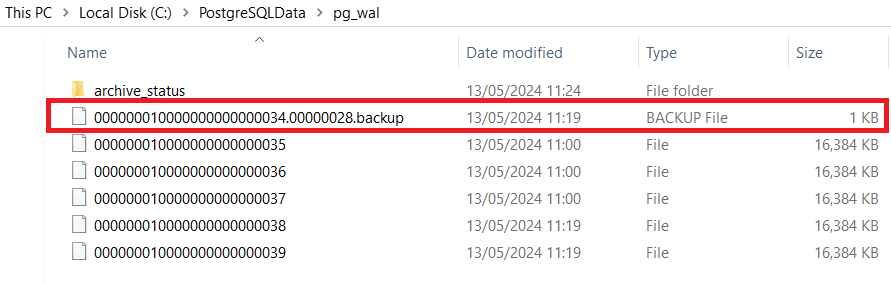
Da bi se omogućilo korišćenje ovako kreiranog osnovnog backup-a, neophodno je zadržati sve segmentne datoteke WAL-a koje su generisane tokom i nakon kreiranja backup-a fajl sistema. Kako bi se olakšao ovaj proces, prilikom pravljenja osnovnog backup-a, stvara se datoteka istorije backup-a koja se odmah smešta u oblast arhive WAL-a. Ova datoteka nosi naziv po prvoj segmentoj datoteci WAL-a koja je potrebna za kreiranje backup-a fajl sistema (na primer ukoliko je početna WAL datoteka nazvana 0000000100001234000055CD datoteka istorije backup-a imaće naziv 0000000100001234000055CD.007C9330.backup) [9]. Nakon što je uspešno sačuvan backup sistema datoteka, kao i segmentne datoteke WAL-a koje su korišćene tokom tog procesa (kako je navedeno u datoteci istorije backup-a), sve starije arhivirane WAL datoteke više nisu potrebne za obnovu sistema i mogu se bezbedno izbrisati (Ipak, preporučuje se čuvanje nekoliko setova backup-a kako bi mogli u potpunosti da se obnove ukoliko je tako nešto potrebno). Datoteka istorije backup-a predstavlja mali tekstualni dokument koji sadrži informacije o samom backup-u. Unutar nje se nalazi oznaka koja je dodeljenja prilikom korišćenja pg\_basebackup alata, kao i informacije o vremenu početka i završetka kreiranja backup-a, zajedno sa segmentima WAL-a koji su obuhvaćeni ovim backup-om. Na slikama u nastavku ovog potpoglavlja prikazano je kreiranje osnovnog backup-a korišćenjem ovog alata i zatim kako izgleda datoteka istorije backup-a.[9]

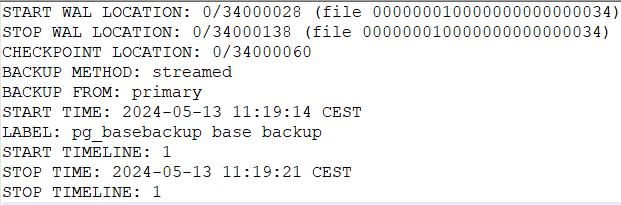


Slika 32-Prikaz korišćenja pg\_basebackup alata za kreiranje osnovne rezervne kopije.Ovo će kreirati osnovni baskup na lokaciji “D: \Basebackup”, što je prikazano na slici 33.



Slika 33-Prikaz kreiranog osnovnog backup-a na lokaciji “D: \Basebackup”





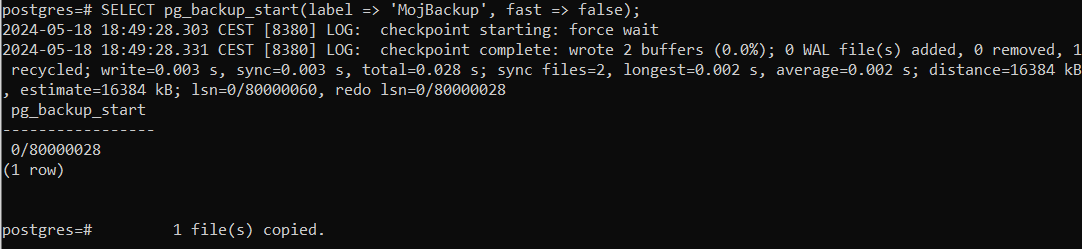
Slika 34-Prikaz kreirane datoteke istorije backup-a u datoteci pg\_wal (slika gore,označeno crvenom linijom) i sadržaj datoteke istorije(slika dole)

Važno je napomenuti, da interval između kreiranja osnovnih backup-ova treba odabrati pažljivo, uzimajući u obzir koliko prostora je potrebno alocirati za arhivirane datoteke WAL-a. Dodatno, potrebno je u obzir uzeti i vreme potrebno za proces obnavljanja, jer će sistem morati da reprodukuje sve segmentne datoteke WAL-a, a to može potrajati ukoliko je prošlo dosta vremena od poslednjeg osnovnog backup-a.

### 3.3.3 Pravljenje osnovnog backup-a(base backup) korišćenjem niskog nivoa API-ja

Još jedan način na koji je moguće kreirati osnovni backup jeste korišćenje niskog nivoa API-ja. Postupak kreiranja osnovnog backup-a korišćenjem niskog nivoa API-ja obuhvata nekoliko koraka više u odnosu na pg\_basebackup metodu, ali je relativno jednostavan. Veoma je važno da se ovi koraci izvrše u nizu i da uspeh jednog koraka bude potvrđen pre prelaska na sledeći korak. Koraci koje ovaj metod obuhvata jesu sledeći:

1. Omogućavanje i pokretanje WAL arhiviranja na način na koji je opisan u sekciji 3.3.1
2. Povezivanje sa serverom kao korisnik sa pravima da pokrene pg\_backup\_start (obično je ovo superkorisnik) i izdavanje komande koja je prikazana na slici 35:

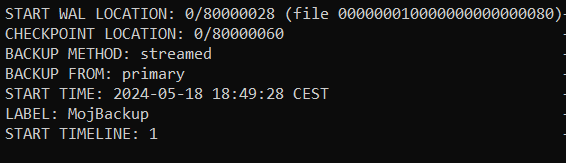


Slika 35-Prikaz izvršenja pg\_backup start komande

Oznaka “MojBackup” na slici 35 predstavlja bilo koji string koji se koristi kako bi se jedinstveno identifikovala ova operacija backup-ovanja. Veza koju pg\_backup\_start poziva se mora održavati do kraja backup-a, ili će backup automatski biti prekinut. Online backup-ovi uvek počinju na početku chechpoint-a. Podrazumevano, pg\_backup\_start će čekati da se završi sledeći redovno zakazani checkpoint, što može potrajati. Ovo je obično poželjno jer minimizira uticaj na radni sistem, međutim, ukoliko je potrebno što pre započeti kreiranje backup-a, onda se kao drugi parametar pg\_backup\_start metode prosleđuje vrednost true, i pri tome se zahteva trenutni checkpoint, koji će se izvršiti što je brže moguće, koristeći sto više I/O resursa.

Komanda pg\_backup\_start, u suštini, izvodi 4 osnovne operacije [10]:

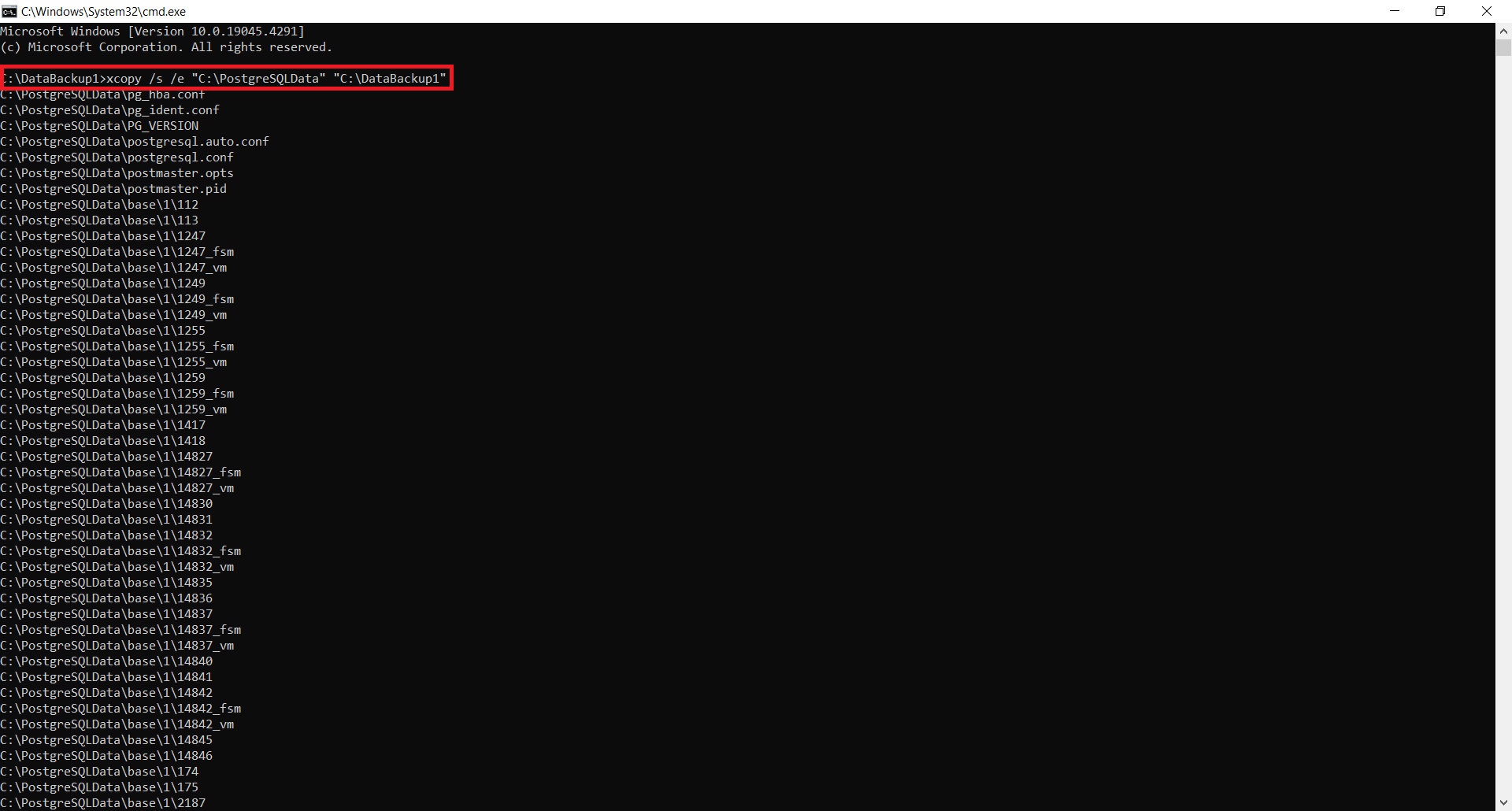
1. Postavljanje baze podataka u full-page write režim (režim pisanja celih stranica). Ovaj režim omogućava serveru da upiše celokupan sadržaj svake disk stranice u WAL, što garantuje da će se te stranice ispravno obnoviti u slučaju pada sistema.
2. Prebacivanje na trenutnu segmentnu datoteku WAL-a.
3. Izvršavanje checkpoint-a (kontrolne tačke).
4. Kreiranje backup\_label datoteke, koja sadrži ključne informacije o samom backup-u. backup\_label se sastoji od 7 elementranih polja koja su prikazana na slici 36.

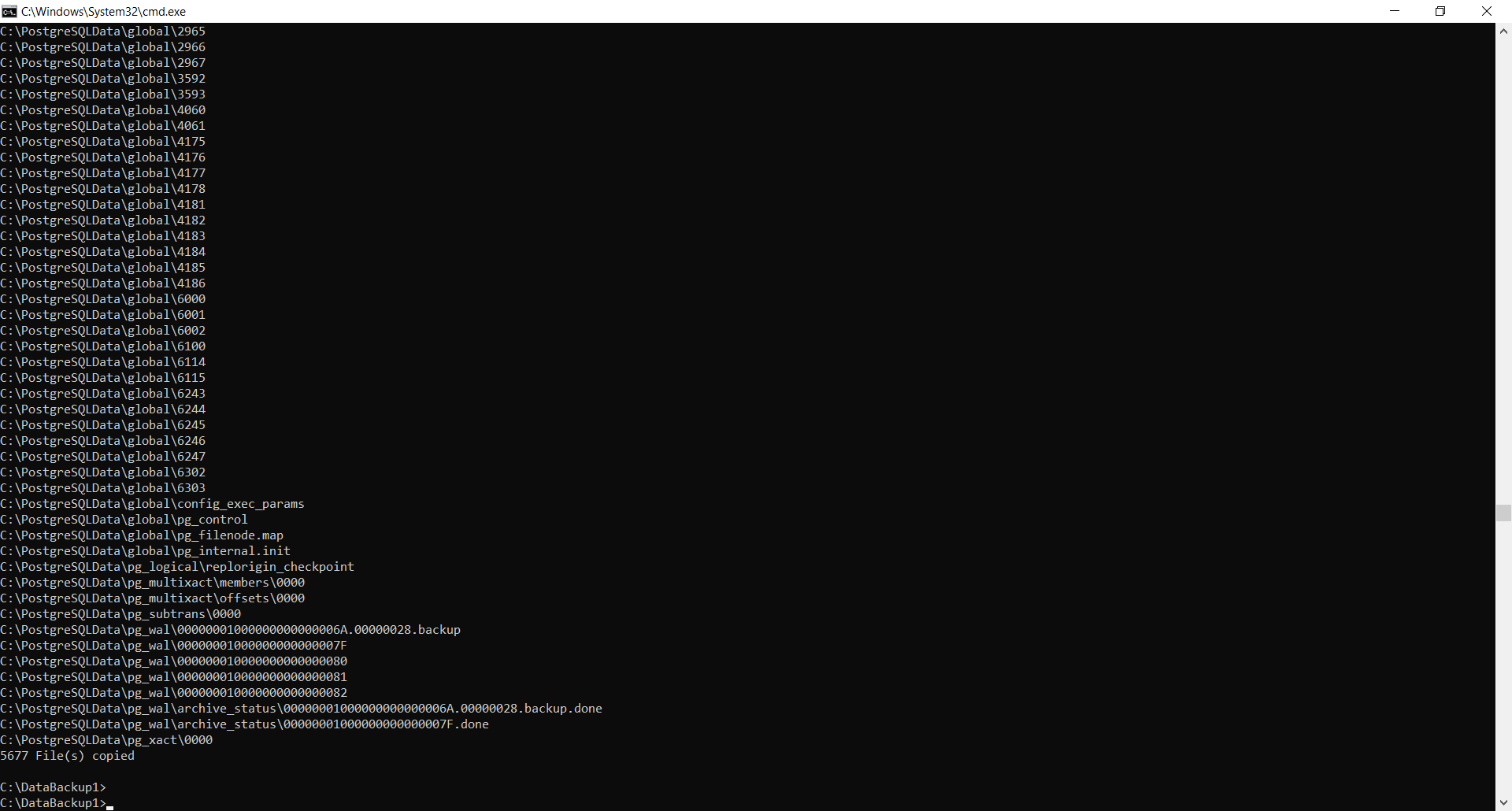


Slika 36-Prikaz sadržaja backup\_label datoteke

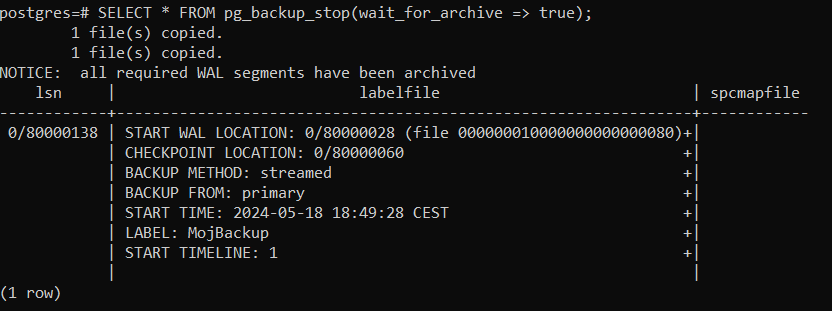
Prvi parametar na slici 36(START WAL LOCATION) se ne koristi prilikom oporavka, već ga koristi rezervni server prilikom replikacije. Drugi parametar (CHECKPOINT LOCATION) predstavlja lokaciju LSN-a[[7]](#footnote-7) na kojoj je zabeležen checkpoint kreiran ovom komandom.BACKUP METHOD se odnosi na metodu koja se koristi prilikom pravljenja backup-a, a BACKUP FROM pokazuje da li je backup kreiran na primarnom ili rezervnom serveru. START TIME je vremenska oznaka koja prikazuje kada je izvršena komanda pg\_backup\_start, LABEL predstavlja oznaku specificiranu komandom pg\_backup\_start, a START TIMELINE je vremenska linija sa koje je backup započeo. [10]

1. Izvršavanje backup operacije korišćenjem alata za backup na nivou sistema datoteka kao što su tar ili cpio (ali ne pg\_dump ili pg\_dumpall) ili korišćenjem xcopy komande. Prilikom izvršavanja ovog procesa, nije potrebno zaustaviti normalan rad baze podataka.Slika 37 ilustruje ovaj postupak korišćenjem xcopy komande.





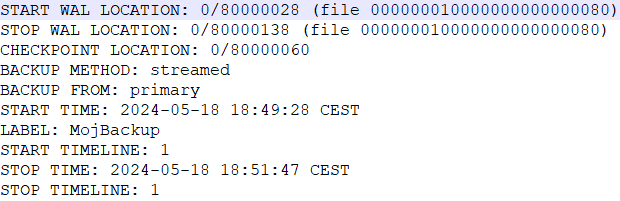
Slika 37-Prikaz operaciju backup-ovanja korišćenjem xcopy komande

1. Za zaustavljanje pg\_backup\_start komande koristi se komanda prikazana na slici 38. 

Slika 38-Prikaz komande pg\_backup\_stop

Komanda pg\_backup\_stop izvršava sledećih pet operacija kako bi završila backup[10]:

1. Vraća bazu podataka u non-full-page writes režim ukoliko je to bilo izmenjeno od strane pg\_backup\_start komande.
2. Piše XLOG[[8]](#footnote-8) zapis o završetku backup-a.
3. Prebacuje WAL segmentnu datoteku.
4. Kreira datoteku istorije backup-a. Ova datoteka sadrži sadržaj backup\_label datoteke i vremenski pečat kada je izvršena komanda pg\_backup\_stop (prikazano na slici 39).



Slika 39-Prikaz sadržaja datoteke istorije vremenske linije

1. Briše backup\_label datoteku, koja je potrebna za oporavak iz osnovnog backup-a, ali jednom kada je kopirana, nije potrebna u originalnom klasteru baze podataka.

Ovime se završava režim backup-ovanja i na serveru se vrši automatska

zamena za sledeći WAL segment.

1. Kada su sve aktivne datoteke segmenta WAL-a, koje su bile u upotrebi tokom backup-a, uspešno arhivirane, postupak je završen. Datoteka koja identifikuje prvu povratnu vrednost pg\_backup\_stop-a predstavlja poslednji segment potreban za formiranje kompletnog backup-a. Ukoliko je archive\_mode omogućen i parametar wait\_for\_archive postavljen na true, pg\_backup\_stop se neće završiti dok se i poslednji segment ne arhivira. Arhiviranje ovih datoteka se obavlja automatski jer su archive\_command i archive\_library već konfigurisane. Uobičajeno, ovaj proces se odvija brzo, ali može doći do kašnjenja, te je potrebno nadgledati sistem. Ako proces arhiviranja zaostaje zbog neuspeha arhivske komande ili biblioteke, nastaviće da pokušava sve dok arhiva ne uspe i dok backup ne bude završen. Kada proces backup-ovanja osigura da su sve datoteke segmenta WAL-a, potrebne za backup, uspešno arhivirane, moguće je postaviti parametar wait\_for\_archive(podrazumevano na true) na false kako bi se funkcija pg\_backup\_stop vratila čim je zapis backup-a napisan u WAL-u.

**Backup direktorijuma sa podacima**

Neki alati za backup-ovanje fajl sistema mogu prijaviti upozorenja ili greške ako se datoteke koje pokušavaju da kopiraju menjaju tokom kopiranja. Prilikom kreiranja osnovnog backup-a aktivne baze podataka, ova situacija je normalna i ne predstavlja grešku. Recimo, alat kao što je rsync daje poseban izlazni kod za takozvane “nestale izvorne datoteke”, pa je potrebno napisati script koji će taj kod tumačiti kao da se ne radi o grešci. Dodatno, neke verzije tar-a daju grešku koja se ne može razlikovati od fatalne greške ukoliko je datoteka prekinuta tokom kopiranja. Iz ovih razloga je jako važno razlikovati obična upozorenja od stvarnih grešaka. Još jedna bitna stvar kod kreiranja backup-a, jeste ta da backup mora da sadrži sve datoteke koje se nalaze u direktorijumu klastera baze podataka, ali je poželjno izuzeti datoteke unutar poddirektorijum pg\_wal iz backup-a jer se time smanjuje rizik od grešaka prilikom obnavljanja sistema. Ovo je jednostavno izvesti ukoliko je ovaj poddirektorijum simbolička veza koja pokazuje na neko drugo mesto izvan direktorijuma klastera, što je često slučaj. Takođe, preporučuje se i izostavljanje postmaster.pid i postmaster.opts, koji beleže informacije o pokrenutom postmaster-u, a ne o postmaster-u koji će koristiti ovu rezervnu kopiju. Preporučuje se i izostavljanje datoteka unutar direktorijuma pg\_replslot, kako bi se izbeglo njihovo uključivanje u backup, jer bi kasnija upotreba ove kopije za kreiranje rezervnog servera mogla izazvati zadržavanje WAL datoteka na rezervnom serveru, što može povećati veličinu primarnog servera. Čak i ako je backup namenjen samo za kreiranje novog primarnog servera, kopiranje slotova za replikaciju nije korisno jer će njihov sadržaj verovatno biti zastareo kada novi primarni server bude pokrenut. [9]

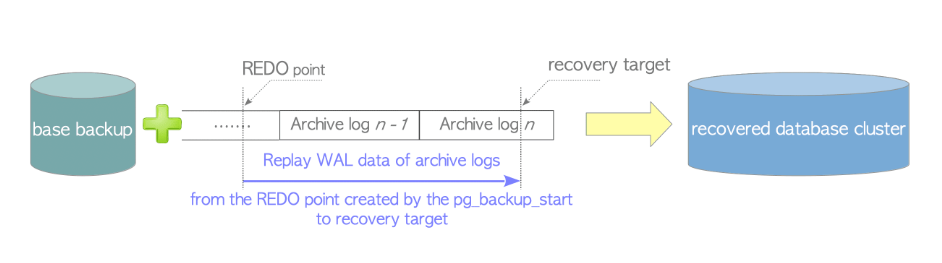
Sve datoteke ili direktorijumi koji počinju sa pgsql\_tmp mogu biti izostavljeni iz rezervne kopije, jer se ove datoteke automatski uklanjaju prilikom pokretanja postmaster-a, a direktorijumi će biti ponovo kreirani po potrebi. Datoteke pg\_internal.init mogu biti izostavljane iz rezervne kopije kada se pronađe datoteka sa istim imenom.Ove datoteke sadrže podatke o keširanju relacija koji se uvek ponovo izgrađuju prilikom oporavka.

Backup kopiju je moguće napraviti i kada je server ugašen, ali u tom slučaju ne bi bilo moguće koristiti pg\_backup\_start ili pg\_backup\_stop, pa je potrebno pratiti koje kopije postoje i koliko daleko idu povezane WAL datoteke. Uglavnom je bolje pratiti i koristiti kontinuirano arhiviranje.

### 3.3.4. Oporavak u tački vremena-Point-In-Time Recovery

Pri oštećenju baze podataka, PostrgreSQL omogućava oporavak do određene tačke u vremenu. Ovaj proces poznat je pod nazivom oporavak u tački vremena (***eng.Point-In-Time-Recovery-PITR***).PITR omogućava povratak baze podataka do tačno definisane tačke u vremenu, što je korisno u situacijama kada je potrebno vratiti bazu podataka na stanje pre određenog događaja ili greške. Za to koristi osnovni backup i arhivirane log-ove (zapise) [10].

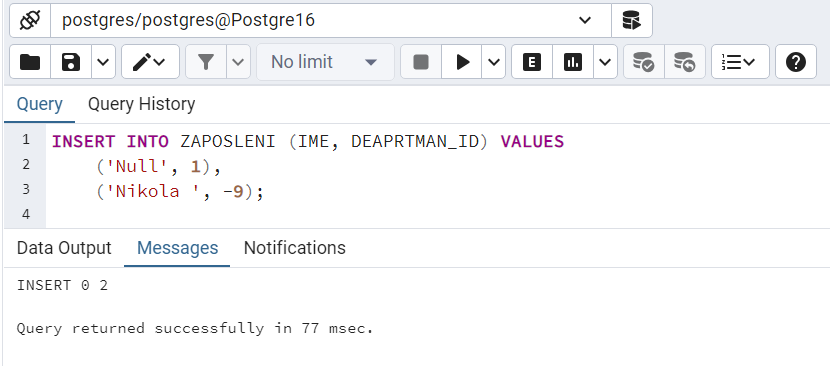
U PITR režimu, PostgreSQL reprodukuje WAL podatke iz arhivskih zapisa (log-ova) na osnovni backup, počevši od REDO[[9]](#footnote-9) tačke koju stvara osnovni backup, do tačke koju je potrebno oporaviti. Tačka oporavka, u ovom slučaju, naziva se cilj oporavka (***eng.recovery target***).

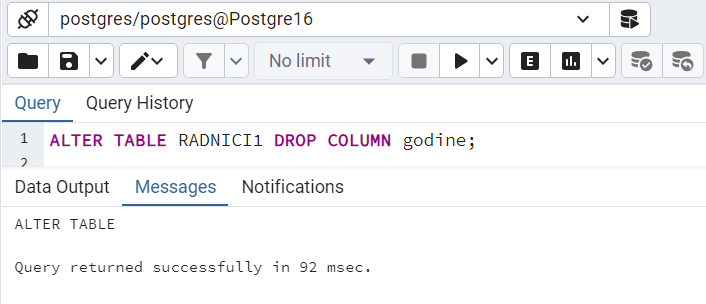


Slika 40-Prikaz osnovnog koncepta PITR-a[[10]](#footnote-10)

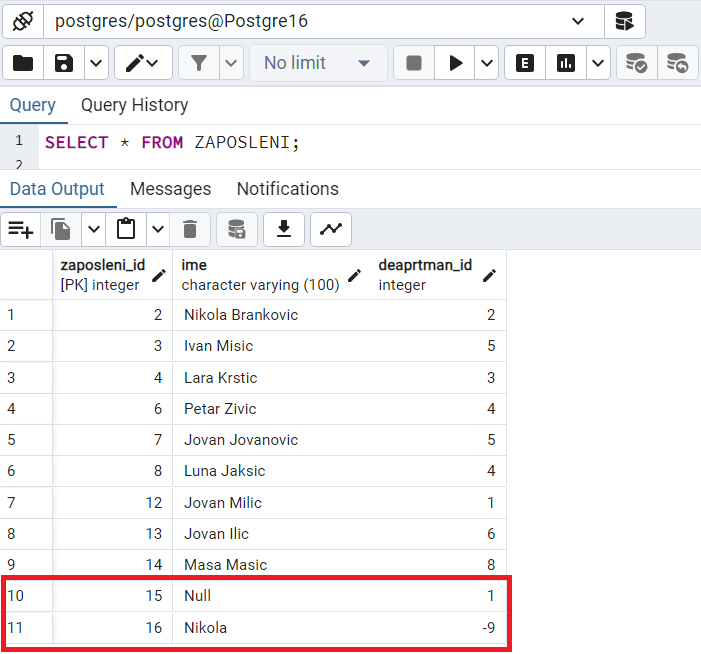
Proces PITR-a(Oporavak u tački vremena) podatke WAL segmenta čita iz arhivskog direktorijuma koji je postavljen u parametru archive\_command, dok se lokacija kontrolne tačke(checkpoint-a) čita iz datoteke backup\_label. Prvo, PostgreSQL koristi internu funkciju read\_backup\_label() da pročita checkpoint vrednost iz datoteke backup\_label kako bi pronašao REDO tačku, a zatim vrši čitanje vrednosti koje su potrebne za oporavak iz konfiguracionog fajla, kao što je komanda za oporavak. Nakon toga, započinje reprodukcija WAL podataka od REDO tačke, koja se određuje iz vrednosti CHECKPOINT LOCATION iz backup\_label datoteke. WAL podaci se čitaju iz arhivskih logova, koji se prebacuju na privremeno područje izvršavanjem komande za oporavak (restore\_command). Kada se proces oporavka završi, kreira se datoteka istorije vremesnke linije u poddirektorijumu pg\_wal(više detalja o ovome biće u sekciji 3.3.5). [10]

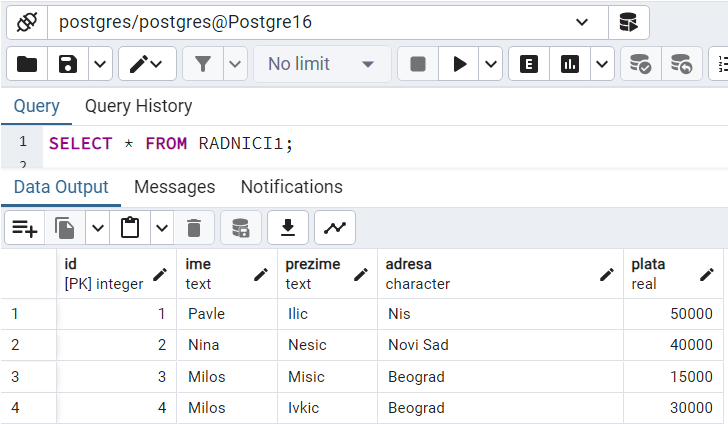
Primeri simulacije oštećenja koja mogu nastati prikazani su na slici 41.





Slika 41-Prikaz oštećenja koja su napravljena nad tabelama RADNICI1 i ZAPOSLENI baze postgres. Gornja slika odnosi se na oštećenje baze unosom nevaladnih vrednosti u neku od tabela (u ovom slučaju u tabelu ZAPOSLENI), dok donja slika podrazumeva oštećenje baze brisanjem određene kolone iz neke od tabele u bazi podataka (tabela RADNICI1).

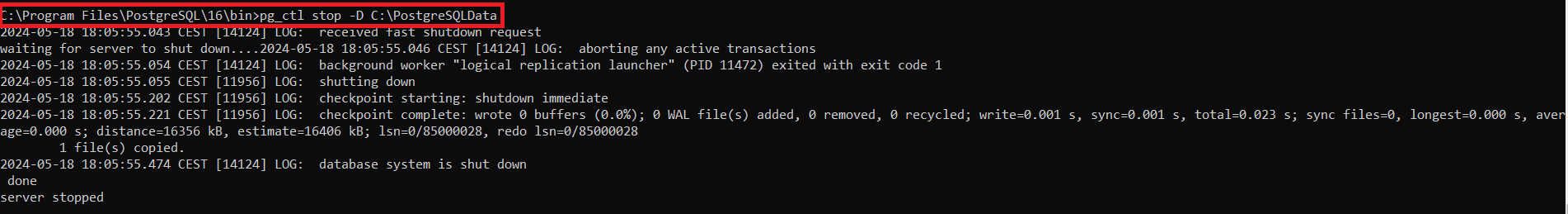




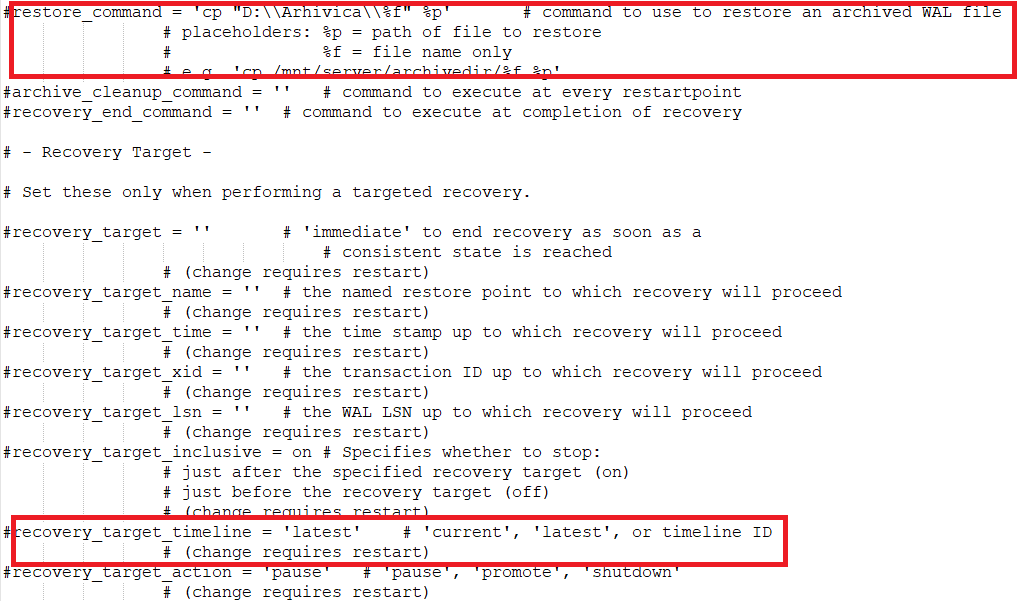
Slika 42-Prikaz tabela RADNICI1 i ZAPOSLENI nakon oštećenja

Obnova baze podataka korišćenjem procesa oporavka u tački vremena (PITR), uključuje nekoliko koraka. Prvi korak je zaustavljanje servera baze podataka. Nakon što se server baze podataka zaustavi, koristi se osnovni backup za oporavak baze podataka. Ključan korak u procesu oporavka do tačke vremena, jeste podešavanje konfiguracionog fajla backup-a. Ovaj korak definiše kako će se baza podataka oporaviti i do koje tačke je potrebno ići. Glavna stavka koja se navodi u ovom koraku je komanda restore\_command. Ova komanda pruža PostgreSQL-u instrukcije o tome kako da povrati arhivirane segmente WAL datoteka. U okviru komande, koriste se placeholderi %f, koji se sada zamenjuje imenom željene WAL datoteke, i %p koji se zamenjuje putanjom do kopirane WAL datoteke (Putanja je relativna u odnosu na trenutni radni direktorijum, tj. direktorijum podataka klastera). [11]

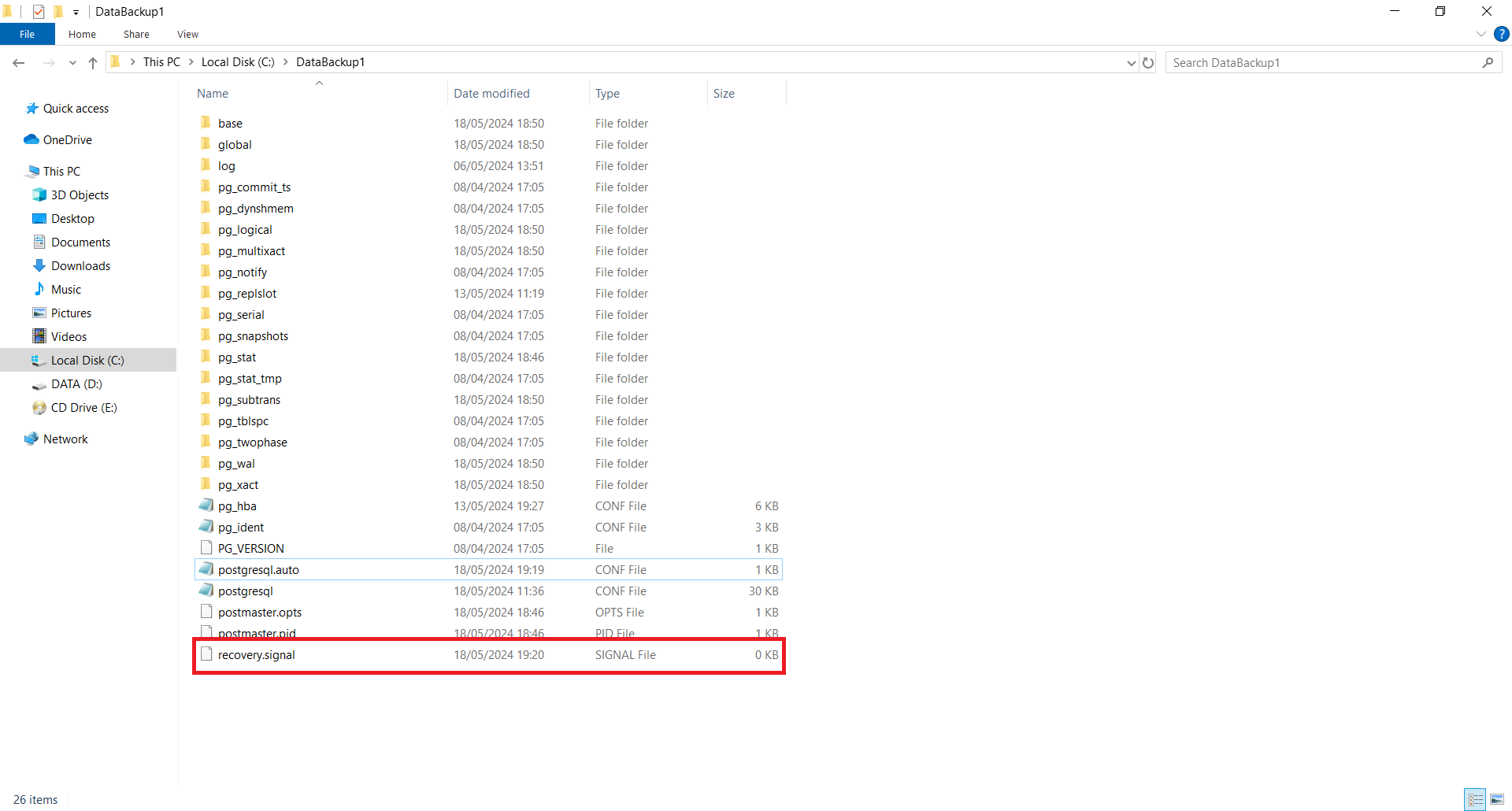
Nakon postavljanja komande za oporavak (restore\_command), potrebno je kreirati recovery.signal fajl u direktorijumu koji sadrži backup baze podataka, koji služi da obavesti bazu da server treba da pokrene u režimu oporavka. Koraci koji ilustruju ovaj postupak prikazani su na slikama koje su date u nastavku teksta.



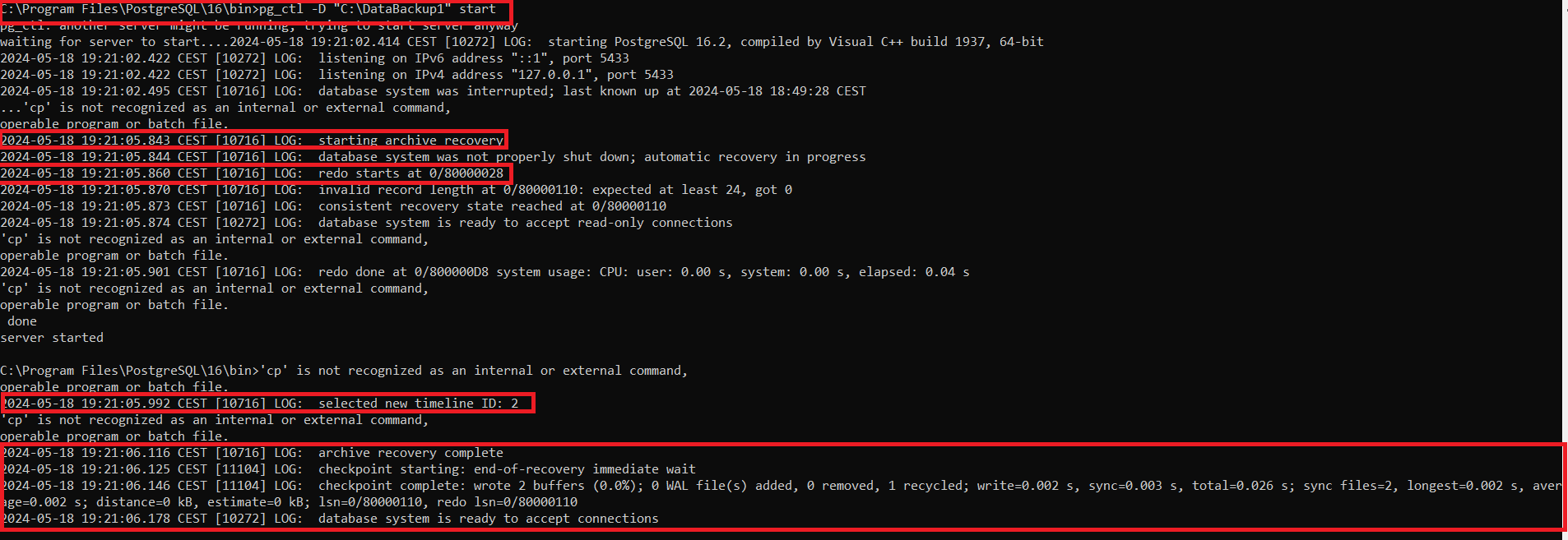
Slika 43-Korak 1:Zaustavljanje glavnog servera baze podataka



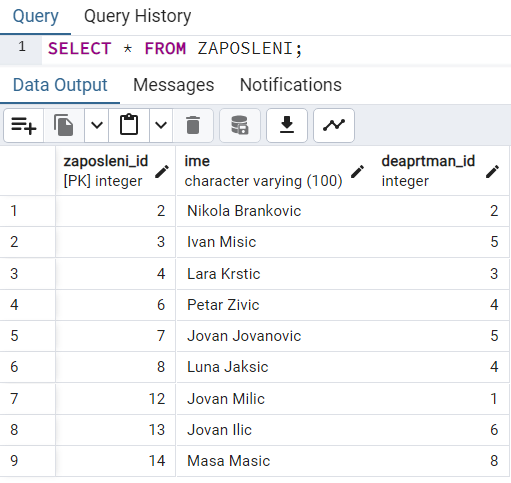
Slika 44-Korak 2:Postavljanje parametara za oporavak u konfiguracionom fajlu backup-a baze podataka. Komanda restore\_command postavlja se na istu vrednost kao i komanda archive\_command kako bi pronašla arhivirane segmente WAL-a.Komanda recovery\_target\_timeline govori o tome da će se baza podataka oporaviti do najnovijeg trenutka. 

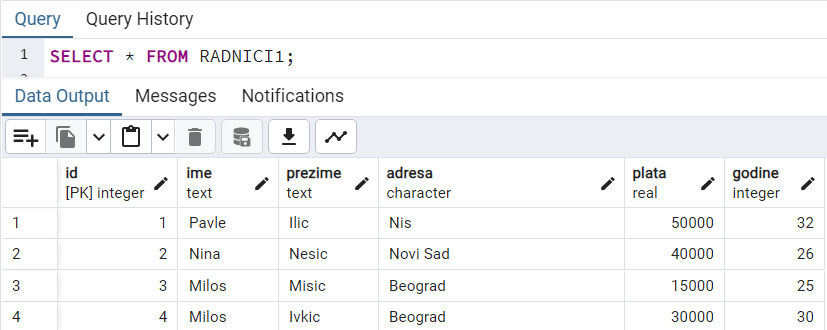


Slika 45-Korak 3: Izvršavanje komande za kreiranje recovery.signal fajla. Naredba “type nul” kreira praznu datoteku recovery.signal u trenutnom direktorijumu (u ovom slučaju direktoijumu backup-a klastera baze podataka), kako je i prikazano na slici 45.



Slika 46- Korak 5: Pokretanje servera baze podataka koji inicira proces oporavka iz osnovnog backup-a(prvi crveni pravougaonik). Na slici se može videti da kada je pronađena REDO tačka (treći crveni pravougaonik), započinje proces oporavka, i nakon uspešnog oporavka, baza podataka je spremna da prihvata nove konekcije, kao što se može videti na slici (četvrti crveni pravougaonik).





Slika 47-Prikaz oštećenih tabela RADNICI1(slika dole) i ZAPOSLENI(slika gore) nakon oporavka. Može se videti sa gornje slike, koja se odnosi na tabelu ZAPOSLENI da ne postoje više nevalidni podaci (kao na slici 41), dok je u tabelu RADNICI1 vraćena kolona godine.

Važno je da komanda restore\_command vrati nenulti izlazni status u slučaju neuspeha. Kada se poziva, komanda će tražiti datoteke koje nisu prisutne u arhivi i treba da vrati nenulti status kada se takvi zahtevi jave. Ovo se ne smatra greškom. Izuzetak su situacije kada je komanda prekinuta signalom ili ako je došlo do greške u shell-u (na primer, komanda nije nađena). U ovakvim situacijama, oporavak će biti prekinut, a server se neće pokrenuti.Iako se očekuje da će komanda tražiti segmente WAL datoteka, važno je napomenuti da sve tražene datoteke neće biti samo segmenti WAL datoteka, već se očekuje i traženje datoteka sa nastavkom .history.Ako segmenti WAL datoteka nisu pronađeni u arhivi, PostgreSQL će ih tražiti u pg\_wal direktorijumu. Međutim, segmenti koji su dostupni u arhivi biće korišćeni pre datoteka koje se nalaze u pg\_wal poddirektorijumu.

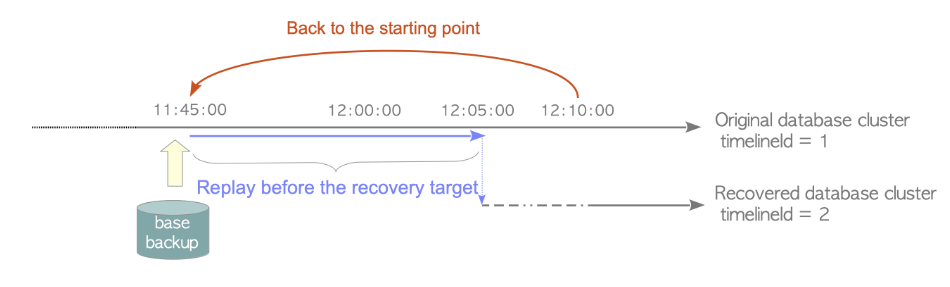
Oporavak baze podataka obuhvatiće sve dostupne segmente, vraćajući bazu podataka na određenu tačku u vremenu, ukoliko je ta tačka navedena u konfigiracionom fajlu. U slučaju da tačka nije navedena, oporavak će se odvijati do trenutne tačke vremena ili što je moguće bliže toj tački. Kada se definiše tačka vremena u konfiguracionom fajlu, važno je da ta tačka bude tačka nakon završetka kreiranja osnovnog backup-a. Ovo se postiže postavljenjem komande recovery\_target\_time na odgovarajuću vrednost.

Ako oporavak otkrije oštećene podatke WAL-a, proces oporavka će se zaustaviti na tom mestu i server se neće pokrenuti. U takvim situacijama, oporavak može biti ponovo pokrenut od početka, uz navođenje “cilja oporavka” pre tačke oštećenja kako bi se proces oporavka normalno nastavio. U slučaju neuspeha oporavka izazvanog spoljnom greškom, kao što je pad sistema ili nedostupnost arhiva WAL segmenata, oporavak se može jednostavno prekinuti, a nastaviće se od mesta gde je prekinut.

### 3.3.5. Vremenske linije

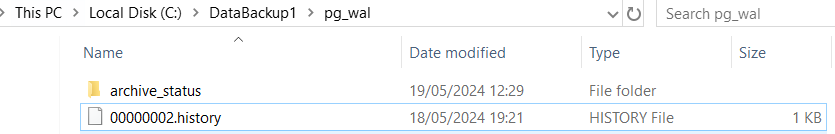
Mogućnost vraćanja baze podataka na tačku u vremenu donosi određene složenosti. Na primer, prilikom korišćenja baze podataka, može se desiti da neka ključna tabela bude izbrisana ili oštećena, recimo u utorak uveče, ali da korisnik to primeti tek dan kasnije. Korisnik zatim odlučuje da se vrati nazad, u utorak uveče, vreme pre brisanja tabele, koristeći backup, i baza se ponovo pokreće. U toj alternativnoj verziji, baza podataka nikada i nije bila izbrisana. Ipak, korisnik (administrator) može shvatiti da to i nije najbolja ideja i da poželi da se vrati dan kasnije (sredu ujutru) u originalnu istoriju. Međutim, to neće biti moguće učiniti, ukoliko je baza podataka, dok je bila aktivna, prebrisala neke od segmenta WAL datoteka koji vode do vremena u koje korisnik želi da se vrati. Zbog toga je važno razlikovati seriju WAL zapisa koja je generisana nakon oporavka do tačke u vremenu od onih koji su generisani u originalnoj istoriji baze podataka.

Kako bi rešio ovaj problem, PostgreSQL koristi koncept vremenskih linija (***eng.timelines***). Svaki put kada se završi proces arhivskog oporavka, kreira se nova vremenska linija koja označava seriju WAL zapisa generisanih nakon tog oporavka. Identifikacioni broj vremenske linije je deo imena datoteka segmenta WAL-a, što sprečava preklapanje podataka WAL-a generisanih od strane prethodnih vremenskih linija. Ovaj identifikacioni broj naziva se timelineid i predstavlja 4-bajtni neoznačeni ceo broj koji počinje od 1.Svakom klasteru baze podataka se dodeljuje pojedinačni timelineid. timelineid originalnog klastera koji se kreira prilikom inicijalizacije baze podataka je 1, ali se ovaj broj povećava za 1 prilikom svakog oporavka klastera baze podataka. Ovaj postupak prikazan je na slici 48.



Slika 48–Prikaz dodeljivanja timelineid-a[[11]](#footnote-11)

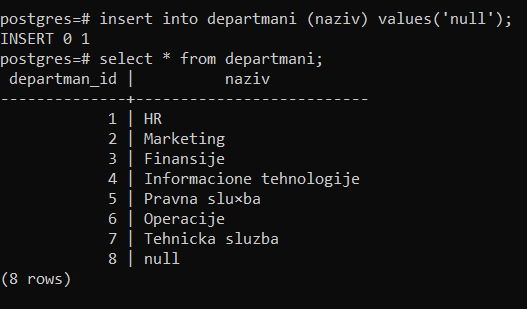
U okviru ovog procesa, najpre se uklanja trenutni klaster baze podataka i vraća se osnovna rezervna kopija koja je kreirana, kako bi bilo moguće vratiti se na početnu tačku oporavka (prikazano crvenom strelicom na slici 48). Zatim, kada se ponovo pokrene PostgreSQL server on ponovo obrađuje WAL podatke iz arhiviranih logova počevši od REDO tačke, koja je kreirana pomoću pg\_backup\_start metode, pa sve do cilja oporavka, prateći početnu vremensku liniju (timelineid 1, predstavljenu plavom strelicom na slici 48). Nakon toga, novom klasteru baze podataka se dodeljuje novi timelineid 2 i PostgreSQL radi na novoj vremenskoj liniji. Prvih 8 cifara naziva WAL segmente su jednake timelineid-u klastera baze podataka koji je kreirao ovaj segment. Prilikom promene timelineid-a, promeniće se i naziv WAL segmenta. Na primer, ukoliko se vrši oporavak koristeći arhivske logove sa nazivima 0000000100000000000000009 i 000000010000000000000000A, redom, novooporavljeni klaster baze podataka će dobiti timelineid 2, i PostgreSQL kreira WAL segment 000000020000000000000000A.

Kako bi bio omogućen povratak u bilo koju tačku vremena, moguće je arhivirati različite vremenske linije, što omogućava vraćanje na bilo koje prethodno stanje. Svaki put kada se kreira nova vremenska linija, PostgreSQL pravi “datoteku istorije vremenske linije” koja pokazuje od koje vremenske linije je grananje nastalo i kada (prikazano na slici 49). Ove istorijske datoteke su neophodne da bi sistem mogao pravilno da odabere segmente WAL datoteka prilikom oporavka iz arhive koja može sadržati više vremenskih linija. Stoga se ove datoteke arhiviraju u području arhive WAL datoteka, baš kao i segmenti WAL datoteka. Istorijske datoteke su male tekstualne datoteke, što omogućava njihovo čuvanje bez ograničenja, za razliku od segmenata datoteka koji su veliki. Podrazumevano ponašanje tokom oporavka je da se oporavi do poslednje pronađene vremenske linije u arhivi. Ukoliko je potrebno izvršiti oporavak na vremensku liniju koja je bila aktuelna kada je napravljan osnovni backup, ili na određenu podvremensku liniju, neophodno je navesti trenutnu ili ciljnu identifikaciju vremenske linije u recovery\_target\_timeline-u (prikazano na slici 50). Dodatno, nemoguće je izvršiti oporavak na vremenske linije koje se nastale pre osnovnog backup-a. 

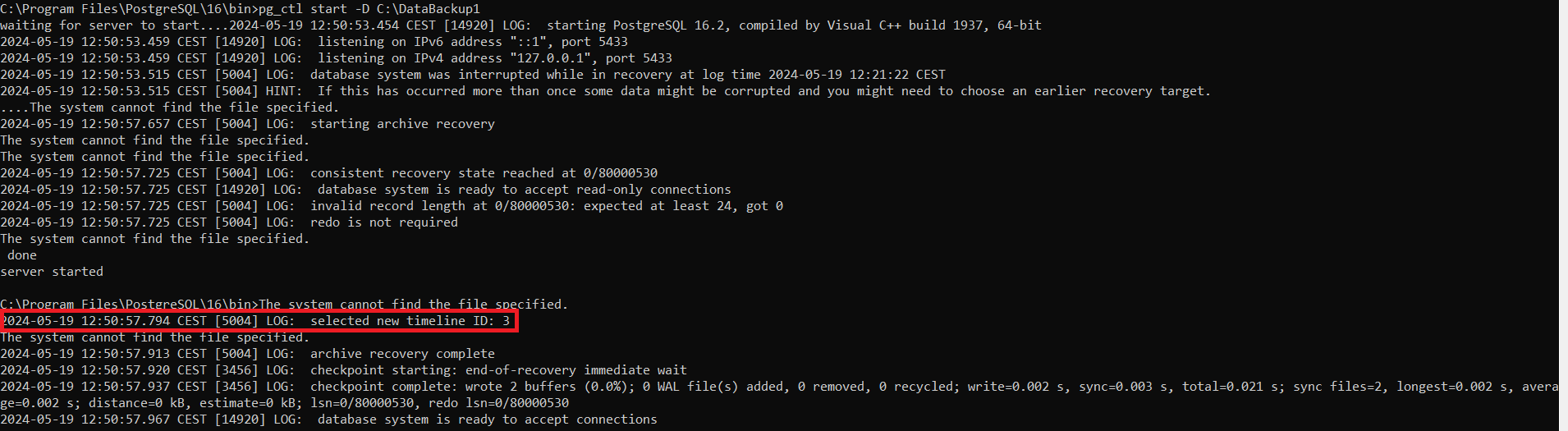
Slika 49-Prikaz kreirane datoteke istorije vremenske linije i njen sadržaj



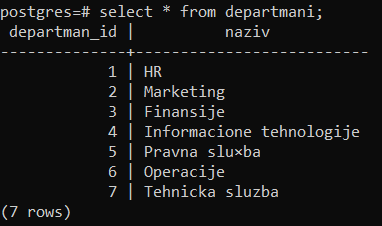
Slika 48-Prikaz postavljanja odgovarajuće vremenske linije kako bi oporavak počeo od te vremenske linije



Slika 50-Prikaz simulacije oštećenja tabele DEPARTMANI



Slika 51-Prikaz uspešnog oporavka baze podataka korišćenjem vremenske linije. Proces oporavka postiže se na isti način kako je opisano sekciji 3.3.4, s tim što se u konfiguracionom fajlu dodatno postavlja odgovarajuća vremenska linija od koje je potrebno započeti oporavak. Na slici se može videti da je timelineid sada postavljen na 3 a da je oporavak krenuo od vremenske linije 2.

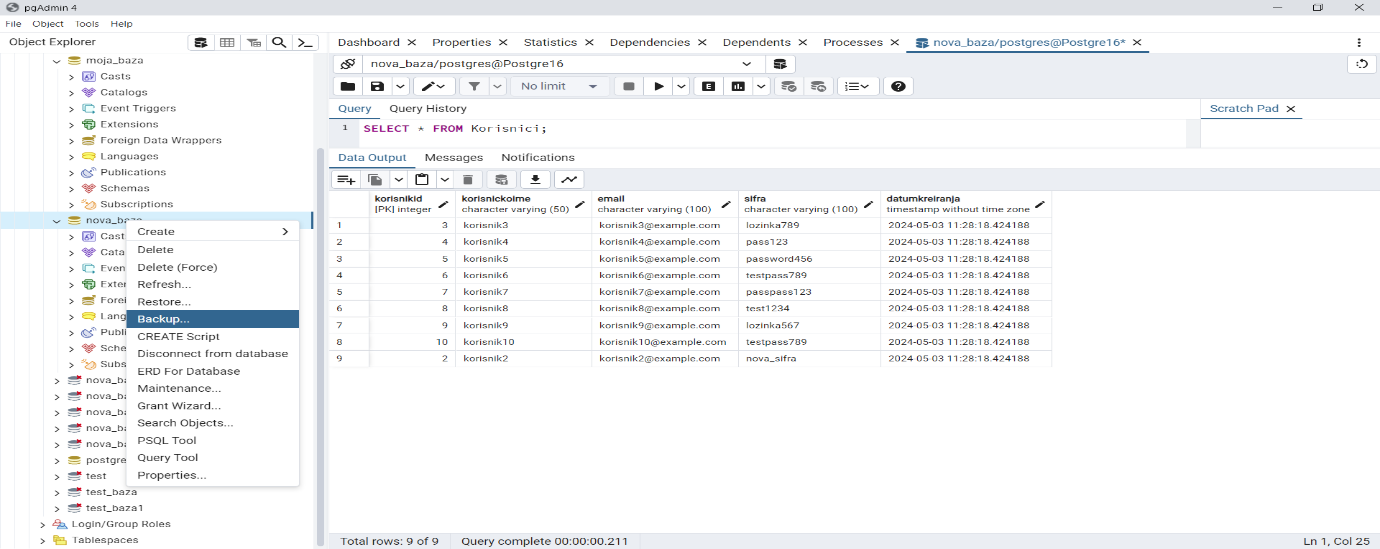


Slika 52-Uspešan oporavak tabele DEPARTMANI

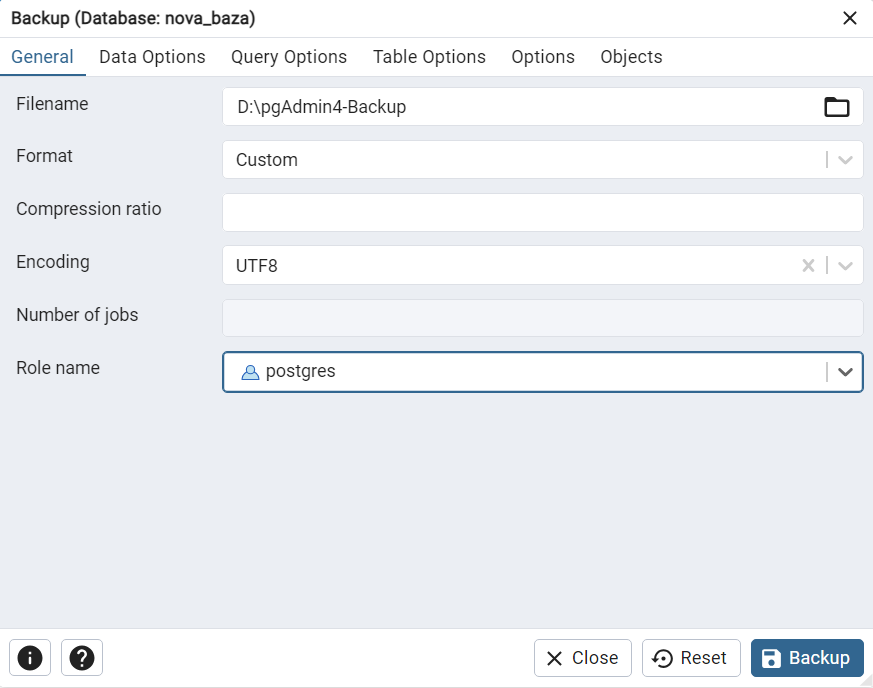
# 4. KREIRANJE BACKUP-A I RESTORE-A KORIŠĆENJEM pgAdmin 4 ALATA

Backup i restore baze podataka kod PostgreSQL-a moguće je kreirati i korišćenjem pgAdmin 4 alata.

Kreiranje backup verzije podataka direktno iz baze postiže se selektovanjem odgovarajućeg modela i desnim klikom na selektovani model, pri čemu se otvara meni iz kog se bira opcija Backup. Izborom ove opcije, otvara se prozor u kom se navode odgovarajući parametri poput tačne lokacije na kojoj će backup biti kreiran, vrste kodiranja koje će se obaviti, formata (najpraktičnije je izabrati Custom format), i naposletku se vrši specifikacija uloga (definiše se vlasništvo nad kreiranom kopijom). Klikom na dugme Backup, kreiran je backup na odgovarajućoj lokaciji. Ovaj postupak prikazan je na slikama 53 i 54 .

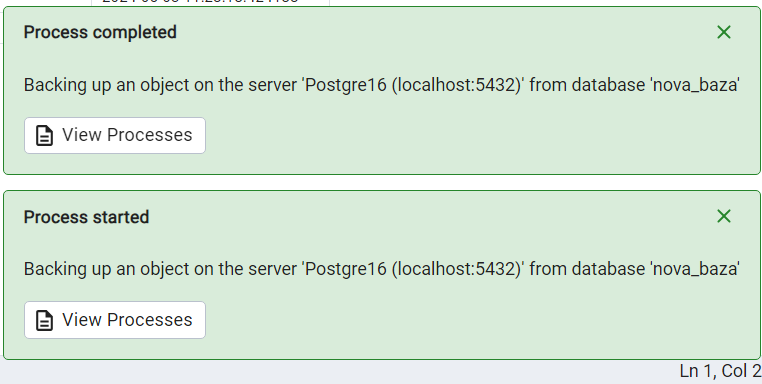


Slika 53–Prikaz početnog koraka za kreiranje backup verzije podataka korišćenjem pgAdmin 4 alata

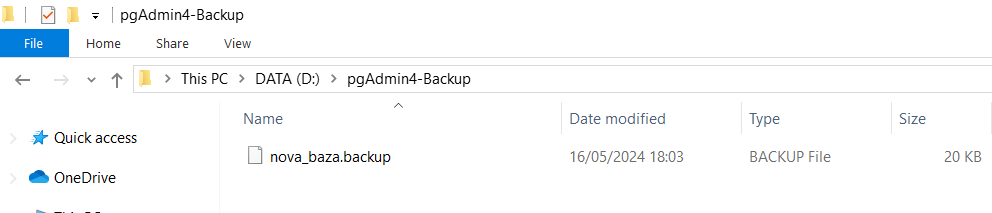


Slika 54 –Prikaz forme za unos informacija o backup verziji prilikom njenog kreiranja

Nakon kreiranja backup-a, sistem obaveštava administratora o ishodu kreiranja i ukoliko je proces izvršen korektno, backup fajl se nalazi na odgovarajućoj lokaciji.

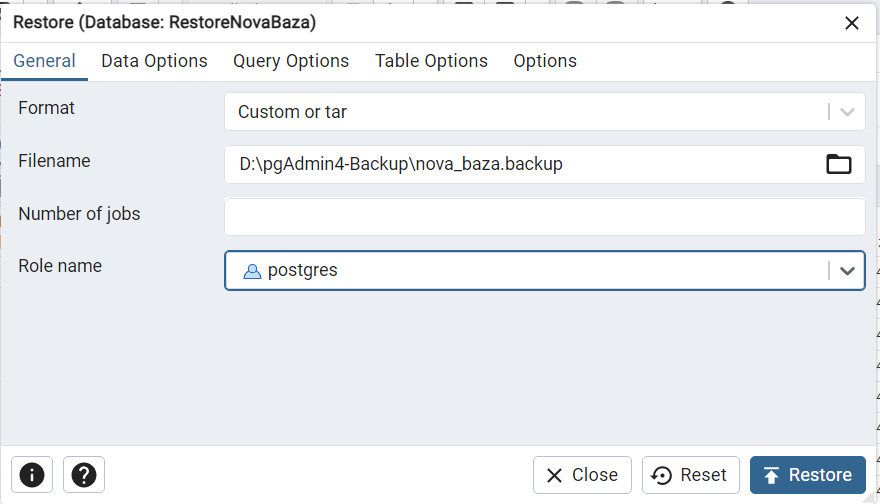


Slika 55-Prikaz poruka koje obaveštavaju administratora o ishodu procesa

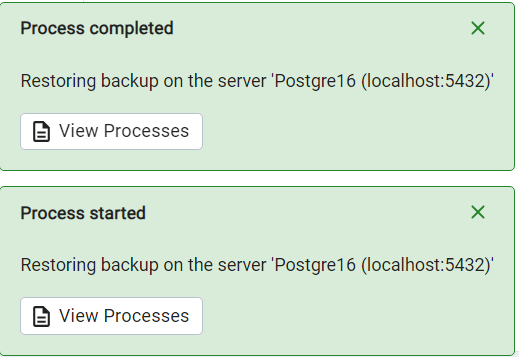


Slika 56- Prikaz kreiranog backup fajla na zadatoj lokaciji

Restore proces u ovom slučaju predstavlja uvoz sačuvanih fajlova iz kreiranog backup-a u sistem iznova. Kao i prilikom korišćenja pg\_dump-a kod kreiranja SQL Dump-a, s obzirom da se i u ovom slučaju vrši backup-ovanje samo jedne baze podataka u jednom trenutku, potrebno je najpre kreirati potpuno praznu bazu podataka. Zatim se ta novokreirana baza selektuje i na isti način na koji je izbarana opcija Backup, bira se opcija Restore. Tom prilikom se navodi ime backup fajla sa tačnom putanjom do njega, odakle će biti ekstrakovani podaci. Nakon završetka, novokreirana baza sadrži sve elemente iz backup-a ukoliko se proces završio korektno. Ukoliko nije, izdaje se poruka o tipu greške koja je sprečila korektan razvoj događaja. Proces restore-ovanja backup-ovanih podataka izabrane baze je prikazan na slikama 57 i 58.



Slika 57- Prikaz forme za izvršenje povratka backup-ovanih podataka u sistem



Slika 58- Prikaz poruka o ishodu Restore procesa

# 5.ZAKLJUČAK

Pregledom seminarskog rada i sticanjem novih znanja o PostgreSQL bazi podataka, ističe se da očuvanje integriteta podataka predstavlja jedan od glavnih prioriteta u toku rada sa bazama podataka, zahtevajući pritom, primenu različitih efikasnih mera i tehnika kako bi se to postiglo. U ovom kontekstu, procesi backup-ovanja i restore-ovanja zauzimaju izuzetno važno mesto.

Backup proces ne samo da omogućava sigurnost podataka tokom potencijalnih kvarova sistema ili gubitka podataka, već predstavlja vitalni deo strategije oporavka podataka. Implementacija adekvatnog backup sistema, koji obuhvata redovno kreiranje siguronosnih kopija podataka i njihovo čuvanje na sigurnom mestu, ključna je za brz i efikasan oporavak u slučaju neželjenog gubitka podataka. PostgreSQL pruža različite mehanizme za kreiranje backup-a, kao što su kreiranje sql dump-a, kreiranje backup-a na nivou celog fajl sistema i korišćenje kontinuiranog arhiviranja kako bi se napravila rezervna kopija, što omogućava prilagođavanje strategija backup-a specifičnim potrebama i okruženjima organizacije.

Sa druge strane, proces restore-ovanja podataka takođe je važan jer omogućava povratak podataka na njihovo originalno mesto ukoliko dođe do gubitka ili oštećenja ovih podataka. Ovaj proces zahteva preciznost i pažljivo planiranje, kako bi se osiguralo da se podaci vrate u ispravno stanje. PostgreSQL pruža mehanizme za jednostavan i efikasan povratak podataka iz siguronosnh kopija, kao što je i prikazano u seminarskom radu, poput komandi psql i pg\_restore, ili korišćenja tehnike oporavka u određenoj tački u vremenu, čime se obezbeđuje kontinuitet poslovanja.

Iz svega priloženog, može se zaključiti da pravilno upravljanje backup-om i restore-om podataka, u svim okruženjima, pa tako i u PostgreSQL okruženju predstavlja ključan korak ka očuvanju bezbednosti podataka i osiguranju pouzdanosti sistema. Razumevanje i primena najboljih praksi u ovim procesima postaju imperativ za organizacije koje žele da osiguraju dostupnost svojih podataka u svakom trenutku.

# 6.SPISAK KORIŠĆENE LITERATURE

[1] Keerthi Rangan, “What Is Database Backup?”, Januar 2022.

Dostupno na: <https://www.g2.com/articles/database-backup?fbclid=IwZXh0bgNhZW0CMTAAAR0nzdBSZVdHXK2LJLrRD1oO4LvDu01g4hT7FGTya9DSzNkYsAfIoFYkA4Y_aem_AQiq3dwl2J7lgMUjA5_VWHnEoRZgyxmUP6X2cOzKZ1D0YxCYg5MdPHRBAV2hS6EF1XuWRfs2ygEpWNkL9aTcWY9w>

[2] Kinza Yasar, Brien Posey, Stacey Peterson ,“What are the best practices for backing up an RDBMS?”.

Dostupno na: <https://www.linkedin.com/advice/0/what-best-practices-backing-up-rdbms-skills-database-development-krsmc>

[3] Kinza Yasar, Brien Posey, Stacey Peterson, “Data restore”.

Dostupno na: <https://www.linkedin.com/advice/0/what-best-practices-backing-up-rdbms-skills-database-development-krsmc>

[4] "How to restore a database from backup?"

Dostupno na: <https://www.acronis.com/en-us/blog/posts/how-to-restore-database-from-backup/>

[5] PostgreSQL 16 Documentation -Backup and Restore, The PostgreSQL Global Development Group 1996–2024.

Dostupno na: <https://www.postgresql.org/docs/current/backup.html>

[6] PostgreSQL 16 Documentation -Backup and Restore-SQL Dump, The PostgreSQL Global Development Group 1996–2024.

Dostupno na: <https://www.postgresql.org/docs/current/backup-dump.html>

[7] “How to Backup and Restore PostgreSQL Database on Windows”.

Dostupno na: <https://sqlbackupandftp.com/blog/how-to-backup-and-restore-postgresql-database/?fbclid=IwZXh0bgNhZW0CMTAAAR0UEHNE2Pr54fBh6mY7S-UsEx8pQjBdCRWf-ETlivjwga5yt3Vtfr9Z580_aem_AXXTS3GrS1kWalf0IavDZy2aBUUicrcI3A45sBeGN8iZdqUiLcGB7hx0ZXIatuHIrGb2GVfoEbFsyeo3OPAUnZK2>

[8] PostgreSQL 16 Documentation -Backup and Restore-File System Level Backup, The PostgreSQL Global Development Group 1996–2024.

Dostupno na: <https://www.postgresql.org/docs/current/backup-file.html>

[9] PostgreSQL 16 Documentation -Backup and Restore-Continuous Archiving and Point-in-Time Recovery (PITR), The PostgreSQL Global Development Group 1996–2024.

Dostupno na: <https://www.postgresql.org/docs/current/continuous-archiving.html>

[10] H. Suzuki, "The Internals of PostgreSQL for database administrators and system

developers",2019.

Dostupno na: <https://www.interdb.jp/pg/index.html>

[11] Tristen Raab,"Various Restoration Techniques Using PostgreSQL Point-In-Time Recovery".

Dostupno na: <https://www.highgo.ca/2023/05/09/various-restoration-techniques-using-postgresql-point-in-time-recovery/>

[12] Talha Saif Malik, "How to Backup and Restore PostgreSQL Databases Using pgAdmin".

Dostupno na: <https://www.commandprompt.com/education/how-to-backup-and-restore-postgresql-databases-using-pgadmin/>

1. postgres- uobičajena baza podataka u PostgreSQL okruženju koja se automatski kreira prilikom instalacije okruženja. [↑](#footnote-ref-1)
2. “Izbačeni blob” se u ovom kontekstu odnosi na binarne podatke koji su ekstrakovani iz baze podataka i smešteni u posebne datoteke unutar direktorijuma koji se koristi za arhiviranje. [↑](#footnote-ref-2)
3. WAL log je skarćenica od Write-Ahead Logging koji predstavlja standardnu metodu za osiguravanje integriteta podataka. [↑](#footnote-ref-3)
4. CHECKPOINT-Tačka u sekvenci WAL loga na kojoj su svi podaci ažurirani tako da održavaju informacije iz loga. [↑](#footnote-ref-4)
5. NFS je skraćenica od Network File System koji predstavlja mehanizam za skladištenje fajlova na mreži i omogućava korisnicima da pristupaju fajlovima i direktorijumima koji su smešteni na udaljenim računarima i da ih tretiraju kao da su lokalni. [↑](#footnote-ref-5)
6. Tape drive je uređaj koji čuva podatke računara na magnetnoj traci, posebno u svrhu backup-ovanja. [↑](#footnote-ref-6)
7. LSN-Broj redosleda zapisivanja, pokazivač na lokaciju u WAL logu. [↑](#footnote-ref-7)
8. XLOG-skraćenica za transaction log, odnosno zapisnik koji čuva informacije o promenama izvršenim nad podacima u bazi tokom transkacija. [↑](#footnote-ref-8)
9. REDO tačka je lokacija XLOG zapisa koji je zabeležen u trenutku kada je pokrenut poslednji checkpoint. [↑](#footnote-ref-9)
10. Izvor slike- https://www.interdb.jp/pg/pgsql10/02.html [↑](#footnote-ref-10)
11. Izvor slike-https://www.interdb.jp/pg/pgsql10/03.html [↑](#footnote-ref-11)