|  |  |  |
| --- | --- | --- |
| C:\Users\Buki\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\E93782CF.tmp | UNIVERZITET U NIŠU ELEKTRONSKI FAKULTET | C:\Users\Buki\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\1F0924B5.tmp |

|  |
| --- |
| Oporavak baze podataka u PostgreSQL-u |
| Seminarski rad |

|  |  |  |
| --- | --- | --- |
| Student: Anastasija Bukumira, br. indeksa 1847 |  | Mentor:  Doc. dr. Aleksandar Stanimirović |

Sadržaj:

[1. Uvod 3](#_Toc212400992)

[1. Koncept oporavka u sitemima baza podataka 4](#_Toc212400993)

[2.1 Uloga i značaj oporavka u bazama podataka 4](#_Toc212400994)

[2.3 Vrste otkaza u bazama podataka 5](#_Toc212400995)

[2.4 Tehnike i algoritmi oporavka baze podataka 6](#_Toc212400996)

[2.5 Razlika između Backup i Recover 6](#_Toc212400997)

[2. Oporavak baze podataka u PostgreSQL-u 7](#_Toc212400998)

[3.1 Značaj backup-a PostgreSQL baza podataka 7](#_Toc212400999)

[3.2 Razlika između fizičkog i logičkog backup-a u PostgreSQL-u 7](#_Toc212401000)

[3.3 Fizički backup u PostgreSQL-u 8](#_Toc212401001)

[**3.3.1 WAL u PostgreSQL-u** 8](#_Toc212401002)

[**3.3.2 Point-in-time recovery (PITR) u PostgreSQL-u** 9](#_Toc212401003)

[**3.3.3 Alati za fizički backup u PostgreSQL-u** 9](#_Toc212401004)

[3.4 Logički backup u PostgreSQL-u 10](#_Toc212401005)

[3.5 Korišćenje logičkih naspram fizičkih backup-a u PostgreSQL-u 10](#_Toc212401006)

[3. Praktični primer: Backup i oporavak podataka u PostgreSQL-u 11](#_Toc212401007)

[4. Zaključak 18](#_Toc212401008)

[6. Literatura 19](#_Toc212401009)

# Uvod

Danas, podaci su nešto najvrednije što svaka organizacija poseduje. Baze podataka se svakodnevno koriste za skladištenje, obradu i upravljanje ogromnim količinama informacija, koje su često ključne za poslovanje. Međutim, kako je razvoj tehnologije omogućio različite načine za rad sa podacima, tako je povećan i rizik od gubitka podataka usled različitih faktora – kao što su hardverski kvarovi, greške u softveru, hakerski napadi ili ljudski faktor.

Tehnologije nastavljaju da se razvijaju, uređaji se koriste sve češće za posao ili zabavu, i to dovodi do stalnog generisanja podataka. Sa porastom količine podataka, postaje sve važnije obezbediti njihovu sigurnost. Pravilno skladištenje i zaštita podataka su od suštinske važnosti, kako bi se izbegao trajni gubitak u slučaju nepredviđenih situacija. U takvim okolnostima, posedovanje odgovarajućih metoda za bekap (*backup*) i oporavak podataka (*recovery*) postaje ključno.

Zbog toga je važno da sistemi za upravljanje bazama podataka, obezbede pouzdane mehanizme za oporavak podataka. Backup i recovery predstavljaju ključne procese koji omogućavaju očuvanje integriteta podataka i kontinuitet rada čak i nakon ozbiljnih kvarova.

PostgreSQL, kao jedan od najpopularnijih sistema za upravljanje relacionim bazama podataka, pruža napredne i fleksibilne alate za backup i oporavak. Njegovi mehanizmi, kao što su Write-Ahead Logging (*WAL*), Point-In-Time Recovery (*PITR*), logički i fizički backup, omogućavaju korisnicima da zaštite svoje podatke i obezbede kontinuitet rada čak i u slučaju grešaka.

Ovaj rad se fokusira na teorijske osnove oporavka baze podataka, konkretne mehanizme koje nudi PostgreSQL, kao i praktičan primer njihove primene.

# Koncept oporavka u sitemima baza podataka

**Koncept oporavka u sistemima baza podataka** zasnovan je na obezbeđivanju da se baza podataka nakon bilo kakvog kvara ili greške može vratiti prvobitno stanje. Oporavak obuhvata skup mehanizama koji sprečavaju trajni gubitak podataka i omogućavaju nastavak rada sistema bez narušavanja integriteta informacija. Savremeni sistemi za upravljanje bazama podataka implementiraju ove procese kroz logovanje transakcija, sigurnosne kopije i kontrolu tačaka oporavka.

## 2.1 Uloga i značaj oporavka u bazama podataka

Kratak odgovor na pitanje zašto je oporavak neophodan u bazama podataka glasi – *„oporavak podataka“*. Bez rezervne kopije baze podataka, oporavak je nemoguć. Ako nešto pođe po zlu, a organizacija ne može da povrati svoje podatke, posledice mogu biti katastrofalne. Štaviše, nemogućnost oporavka podataka nije retka situacija, jer bez skoro napravljene rezervne kopije, odgovor na pitanje povraćaja podataka najčešće je – *„nije mogu*će*“*. Bezbednost podataka predstavlja dodatni razlog zbog kojeg je strategija bekapa od presudnog značaja. U slučaju napada, kao što je, na primer, ransomware, jedini pouzdan način da se sistem vrati u prethodno ispravno stanje jeste primena procesa oporavka, koji omogućava povratak baze na tačku pre incidenta. Ukoliko rezervna kopija ne postoji, ne samo da nije moguće povratiti izgubljene podatke, već postoji rizik od potpunog prekida poslovanja.

Sa tehničkog stanovišta, uloga oporavka usko je povezana sa pravilnim izvršavanjem transakcija unutar DBMS-a. Svaki put kada se neka transakcija pošalje sistemu na izvršavanje, neophodno je da sve operacije koje ta transakcija obuhvata budu uspešno završene. Njihov efekat mora biti ili u potpunosti upisan u bazu podataka ili, u slučaju neuspeha, transakcija ne sme ostaviti nikakav trag niti uticati na druge transakcije. Sistem za upravljanje bazama podataka ne sme dozvoliti da se samo deo operacija transakcije primeni nad bazom, jer bi to dovelo do neočekivanog stanja baze podataka. Ovakva situacija može nastati ukoliko dođe do prekida transakcije nakon što su neke operacije već izvršene, ali pre nego što je ceo proces završen. Upravo zbog toga mehanizmi oporavka obezbeđuju vraćanje baze u prethodno ispravno stanje, čuvajući integritet i pouzdanost podataka.   
  
  
2.2 Uzroci gubitka podataka i potreba za oporavkom

Postoje brojni razlozi zbog kojih je mehanizam oporavka neophodan u sistemima baza podataka. Jedan od najčešćih razloga su sistemski kvarovi, koji mogu nastati usled hardverskih oštećenja, grešaka u softveru ili nestanka električne energije. Takve situacije mogu dovesti do prekida rada sistema, oštećenja podataka ili njihovog potpunog gubitka. U takvim uslovima, jedino oporavak omogućava vraćanje baze u konzistentno stanje i nastavak rada bez trajnih posledica.

Pored sistemskih grešaka, česte su i transakcione greške, koje nastaju usled mrežnih prekida, mrtvih petlji (*deadlock*) ili logičkih grešaka u aplikaciji. Ukoliko se transakcija prekine pre nego što je završena, baza podataka mora biti sposobna da poništi sve delimično izvršene operacije. Mehanizmi oporavka omogućavaju poništavanje (*rollback*) ili ponovno izvođenje (*redo*) transakcija kako bi se očuvala doslednost sistema.

Ljudske greške takođe predstavljaju jedan od uzroka gubitka podataka. Nenamerno brisanje, prepisivanje ili pogrešan unos informacija može dovesti do narušavanja integriteta baze. U takvim slučajevima, oporavak je jedini način da se izgubljeni ili oštećeni podaci povrate i vrate u ispravno stanje. Sličan značaj oporavak ima i u slučaju bezbednosnih incidenata, kao što su hakerski napadi ili neovlašćen pristup, koji mogu kompromitovati integritet i poverljivost informacija. Obnova podataka iz bezbednih kopija omogućava organizacijama da izbegnu posledice ovakvih napada i spreče dalje narušavanje sistema.

Takođe, ažuriranje hardvera kao i migracije sistema mogu predstavljati rizik po stabilnost baze podataka. Tokom prenosa podataka na novi sistem, može doći do delimičnog gubitka informacija, te je postojanje rezervnih kopija od ključne važnosti.   
  
Pored toga, prirodne katastrofe, poput poplava, požara ili zemljotresa, takođe mogu uništiti fizičku infrastrukturu i same podatke.

Važno je istaći da oporavak podataka nije potreban samo u vanrednim situacijama, već i zbog zakonskih i regulatornih obaveza. Mnoge industrije zahtevaju čuvanje podataka određeni vremenski period, a gubitak informacija može dovesti do pravnih posledica. Oporavak je ključan i u slučajevima korupcije podataka, koja može nastati usled virusa, grešaka u softveru ili hardverskih kvarova. Drugim rečima, vraćanjem baze u prethodno validno stanje, organizacije obezbeđuju kontinuitet rada i pouzdanost informacija. [1]

## 2.3 Vrste otkaza u bazama podataka

Otkazi u sistemima baza podataka predstavljaju situacije u kojima baza nije u mogućnosti da izvrši neku transakciju ili dolazi do gubitka podataka. Jedan od tipova otkaza je transakcioni otkaz, koji nastaje kada transakcija ne može biti završena ili više nije prihvatljiva za izvršenje. Slično tome, pad sistema ili sistemskikvar javlja se usled grešaka u hardveru, softveru ili mrežnim problemima, najčešće tokom izvršavanja transakcije. Hardverski kvarovi, poput kvara diska ili oštećenja čitača/pisača, takođe spadaju u ovu kategoriju.

Postoje i sistemske greške, koje se javljaju kada neka operacija tokom transakcije dovede do greške, na primer deljenje sa nulom ili neispravan unos parametara. Takve greške mogu nastati i zbog logičkih programskih grešaka, a ponekad ih izazove i prekid izvršavanja od strane korisnika, što može dovesti do neuspeha transakcije.

Takođe, ulogu imaju i mehanizmi kontrole konkurentnosti, koji mogu odlučiti da transakciju ponište ili ponovo pokrenu ukoliko dođe do mrtvih petlji (*deadlock*) između više procesa. Pored toga, kvarovi na disku mogu nastati kada disk izgubi podatke zbog greške pri čitanju ili pisanju ili zbog kvara diska, što može ometati izvršenje transakcija.

U širem smislu, katastrofe obuhvataju fizičke probleme, što podrazumeva nestanak struje, kvar klimatizacije, požar, krađu, sabotažu ili nenamerno prepisivanje diska ili traka - što opet može dovesti do gubitka podataka ako transakcije u toku nisu pravilno obrađene mehanizmima oporavka. Slično tome, mrežni otkazi se javljaju kada dođe do prekida komunikacije između klijenta i servera baze podataka, što može biti izazvano hardverskim kvarom, softverskom greškom ili prirodnom katastrofom.

Još jedan specifičan tip problema je deadlock, koji se javlja kada dve ili više transakcija čekaju da druga oslobodi resurse koji su im potrebni za nastavak, što dovodi do zastoja i eventualnog otkazivanja jedne ili više transakcija.

## 2.4 Tehnike i algoritmi oporavka baze podataka

Recovery manager igra centralnu ulogu u obezbeđivanju konzistentnosti baze podataka primenom *undo* operacija kako bi se povratila stanja transakcija koje još uvek nisu komitovane ili aktivnih transakcija, i *redo* operacija kako bi se povratile komitovane transakcije koje još nisu upisane u bazu podataka. I *undo* i *redo* zapisi se primenjuju na stanje sačuvano u kontrolnoj tački (checkpoint) kako bi se sistem mogao vratiti u prethodno stanje.  
Kontrolna tačka (checkpointing) podrazumeva čuvanje trenutnog stanja baze podataka, koje zapravo služi kao referentna tačka za oporavak. Recovery manager koristi kontrolnu tačku i log fajlove da bi vratio bazu podataka u prethodno stanje, primenjujući po potrebi *undo* i *redo* zapise.  
*Shadow paging* je jednostavan algoritam koji podrazumeva da se svi podaci o izmenama transakcije čuvaju u shadow kopiji baze podataka, dok glavna kopija predstavlja izvršenje svih komitovanih transakcija i nijedne prekinute. Kada se transakcija potvrdi, shadow kopija se zamenjuje glavnom kopijom, čime se izmene instaliraju. Međutim, ova tehnika se ne koristi široko u komercijalnim proizvodima jer se ne skalira dobro pri velikom broju transakcija.   
*Write-ahead logging – WAL* (pravilo prethodnog upisa) zahteva da se *undo* informacije za stranicu upišu u stabilni log pre nego što se ta stranica pošalje u bazu podataka.   
Garbage collection pravilo zabranjuje uklanjanje log zapisa iz stabilnog loga osim ako nije zagarantovano da taj zapis više neće biti potreban za oporavak. Ova pravila su ključna za ispravnost i visoku dostupnost u algoritmima za oporavak. [2]

## 2.5 Razlika između Backup i Recover

Backup podrazumeva čuvanje kopije originalnih podataka kako bi se oni mogli vratiti ako dođe do gubitka ili oštećenja. Backup predstavlja jednu od metoda zaštite podataka - organizacije prave kopije kritičnih podataka, bilo na drugom uređaju ili na posebnoj lokaciji. Postoje različiti tipovi backup-a, kao što su puni (full), inkrementalni, lokalni ili mirror backup.  
S druge strane, oporavak (recovery) je proces vraćanja izgubljenih ili oštećenih podataka pomoću dostupnih backup kopija. Čak i ako se desi da podaci budu izgubljeni ili oštećeni, oporavak omogućava vraćanje baze u funkcionalno stanje.   
Backup podrazumeva sledeće prednosti: zaštitu od gubitka podataka, omogućavanje vraćanja ranijih stanja i održavanje kontinuiteta rada. Njegove mane jesu visoki troškovi za hardver i softver, potreba za održavanjem i rizik da, ako backup nije pravilno izveden, podaci mogu biti nepovratni izgubljeni.   
Ukoliko govorimo o oporavku, prednosti su što se sprečava trajni gubitak podataka i što se može koristiti u scenarijima katastrofa. Međutim, oporavak nije zagarantovan - nije svaki podatak moguće vratiti uspešno - i alati za oporavak mogu biti skupi ili nepouzdani.   
Dakle, backup je kreiranje dodatne kopije podataka kako bi se imali resursi za vraćanje u slučaju greške, dok je oporavak proces korišćenja tih kopija da se povrate podaci u prvobitno stanje. Backup olakšava oporavak, dok oporavak sam po sebi nema ulogu u kreiranju kopija - on je reakcija na greške i padove sistema. [3]

# Oporavak baze podataka u PostgreSQL-u

PostgreSQL, kao jedan od najrasprostranjenijih relacionih sistema za upravljanje bazama podataka, poseduje napredne mehanizme za zaštitu i oporavak podataka koje će biti obrađene u ovom poglavlju.

3.1 Značaj backup-a PostgreSQL baza podataka   
  
Kada govorimo o backup-u i oporavku, to podrazumeva skup procesa i protokola koji su uspostavljeni kako bi zaštitili podatke od gubitka ili oštećenja i vratili ih u stanje pre nego što je bilo potrebe za oporavkom.   
Backup podrazumeva pravljenje kopija podataka u redovnim intervalima, kopija koje prikazuju stanje PostgreSQL baze podataka u određenom trenutku.   
Oporavak, s druge strane, je proces vraćanja podataka iz tih backup kopija. Ako se oba procesa pravilno sprovedu (to jest, ako uvek postoje ažurne backup kopije i dobra strategija oporavka), sama PostgreSQL baza će biti otpornija na kvarove i biće zaštićena od gubitka podataka.  
Efikasno upravljanje backup-om ne podrazumeva samo pravljenje kopija podataka, već je važno osigurati da te kopije budu ispravne, tačne i ažurne.   
Da bi dobra strategija backupovanja bila definisana za PostgreSQL bazu, potrebno je razmotriti nekoliko aspekata - to uključuje koliko često će se raditi backup, gde će backup biti smešten i koliko često će biti proveravani.   
Međutim, nije dovoljno samo napraviti ažurne i ispravne backup kopije baze, već je potrebno i uspostaviti efikasan protokol za oporavak od katastrofe. Plan za oporavak mora obuhvatiti sve korake potrebne za što brže vraćanje podataka nakon incidenta. Time se obezbeđuje da baza nije samo backupovana, već i da se može efikasno i pravovremeno oporaviti.

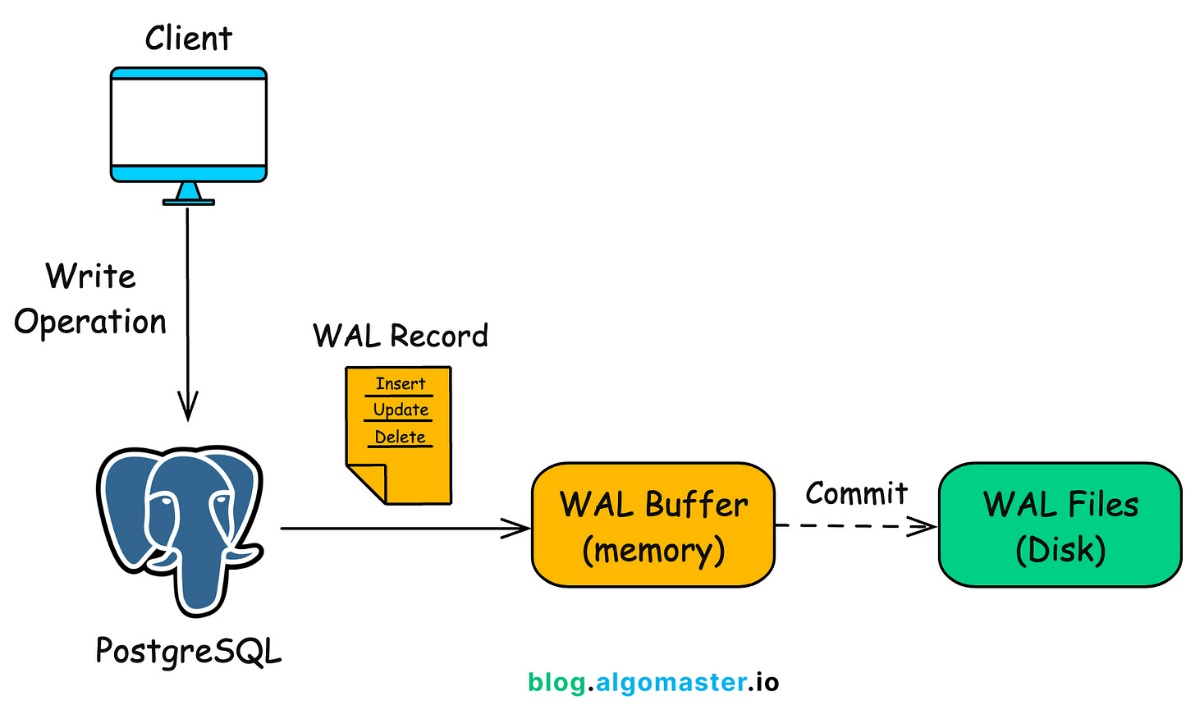
## 3.2 Razlika između fizičkog i logičkog backup-a u PostgreSQL-u

U PostgreSQL-u postoje dve glavne vrste backup-a baze podataka: fizički backup i logički backup.  
**Fizički backup** sadrži binarne podatke, odnosno stvarne fajlove baze podataka. **Logički backup** čuva podatke u formatu koji je čitljiv čoveku, najčešće u vidu SQL skripti. Logički backup sadrži SQL naredbe za kreiranje objekata baze i unošenje podataka i može biti veoma detaljan, omogućavajući backup specifičnih objekata kao što su tabele, šeme ili cela baza. Takođe, logički backup je prenosiv i može se koristiti na različitim sistemima ili verzijama baza, što ga čini popularnim za migraciju malih i srednjih baza.   
Glavni nedostatak logičkog backup-a je brzina. Za velike baze, proces vraćanja iz logičkog backup-a je prespor da bi bio koristan kao jedini mehanizam za oporavak od katastrofe. Sa druge strane, vraćanje iz fizičkog backup-a je brže i preciznije - kada se pravi strategiju za oporavak od katastrofe, uglavnom se koristiti fizički backup.   
Takođe. postoje i **puni (*full*), inkrementalni (*incremental*)** i **diferencijalni (*differential*) backup.** Puni backup predstavlja kompletnu kopiju svih podataka, dok inkrementalni backup obuhvata samo one podatke koji su promenjeni nakon poslednjeg backup-a (bilo punog ili inkrementalnog). Diferencijalni backup čuva izmene nastale od vremena poslednjeg punog backup-a. Inkrementalni i diferencijalni backup-i uvedeni su kako bi se smanjilo vreme potrebno za izradu kopije baze i potrošnja prostora na disku, što je posebno značajno kod velikih sistema.

3.3 Fizički backup u PostgreSQL-u  
  
Fizički backup-i u PostgreSQL-u se nazivaju backup-ima fajl sistema. To se odnosi na proces direktnog kopiranja direktorijuma i fajlova koje PostgreSQL koristi za čuvanje podataka, što rezultira kompletnom slikom baze u određenom trenutku.   
Održavanje backup-a fajl sistema je ključni deo svake strategije za oporavak od katastrofe i neophodan jeprilikom korišćenja baza podataka. Međutim, kreiranje plana oporavka od katastrofe zahteva i druge tehnike osim redovnog pravljenja fizičkih backup-a.   
Pravljenje fizičkog backup-a veoma velikih baza može biti sporo i zahtevno za resurse, što može ometati druge važne zadatke baze i uticati na ukupne performanse. Fizički backup nije dovoljan za obezbeđivanje konzistentnosti u slučaju kvara, jer on odražava samo stanje baze u trenutku kada je napravljen. Da bi se baza nakon kvara obnovila, potreban je dodatni mehanizam za vraćanje svih transakcija koje su se desile između poslednjeg backup-a i trenutka kvara.

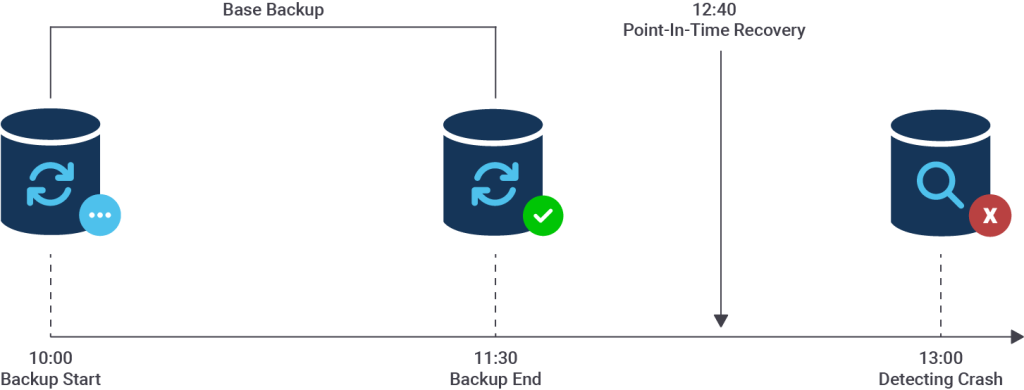
**3.3.1 WAL u PostgreSQL-u**   
  
WAL je sekvencijalni zapis (log) svih promena u bazi podataka, koji se upisuje **pre** nego što se te promene primene na stvarne podatke (na heap ili indeks stranice). Na taj način se obezbeđuje da PostgreSQL, u slučaju pada sistema ili nestanka struje, može da reprodukuje log i vrati bazu u bezbedno i konzistentno stanje.

Kada klijent izvrši operaciju upisa (poput INSERT, UPDATE, DELETE), PostgreSQL prvo generiše WAL zapis koji opisuje tu izmenu i upisuje ga u memoriju (*WAL bafer*). Tek nakon potvrde transakcije (*COMMIT*), ovaj zapis se trajno upisuje na disk pomoću fsync mehanizma, čime se obezbeđuje trajnost transakcije. U slučaju pada sistema, PostgreSQL reprodukuje WAL zapise počevši od poslednjeg checkpoint-a, kako bi vratio bazu u konzistentno stanje i povratio sve izgubljene izmene (slika br. 1). [4]

  
*Slika br. 1 – WAL*

U idealnom scenariju oporavka, prvo se vrati najnoviji backup (na primer, od prethodnog dana), a zatim se primene WAL zapisi nastali nakon tog backup-a kako bi baza bila ažurirana do najnovijeg stanja.

**3.3.2 Point-in-time recovery (PITR) u PostgreSQL-u**   
  
Point-in-time recovery označava vraćanje PostgreSQL baze podataka na tačku u vremenu koju korisnik izabere. Na primer, ako je izvršena neka promena i iz bilo kog razloga je sada potrebno poništiti tu promenu, može biti izabran bilo koji dan pre nadogradnje za oporavak (slika br. 2).

  
*Slika br. 2 – Point in time recovery*

U pozadini, PITR u PostgreSQL-u je zasnovan na WAL-u - arhiviranjem WAL zapisa moguće je vratiti bazu iz backup-a i potom reprodukovati te zapise do određenog trenutka, čime se omogućava svojevrsno „putovanje kroz vreme“ — vraćanje baze tačno u željeno stanje.

**3.3.3 Alati za fizički backup u PostgreSQL-u**   
  
Postoji više alata za pravljenje fizičkih backup-a, a dva najpopularnija su ***pg\_basebackup*** i ***pgBackRest***.

***pg\_basebackup*** je jednostavan, ali veoma snažan alat koji se koristi za pravljenje osnovne kopije tekućeg PostgreSQL klastera baze podataka. Proces izrade kopije baze obavlja se bez prekidanja rada drugih klijenata i može se koristiti kako za oporavak baze do određene tačke u vremenu (point-in-time recovery). Ovaj alat omogućava pravljenje punog ili inkrementalnog osnovnog backup-a baze podataka. Backup se uvek odnosi na čitav klaster baze podataka, što znači da nije moguće napraviti kopiju pojedinačnih baza ili objekata. Za selektivne backup-e potrebno je koristiti druge alate, kao što je ***pg\_dump.*** [5]

***pgBackRest*** je open-source alat namenjen pravljenju fizičkih backup-a, koji uvodi brojna poboljšanja u odnosu na klasični *pg\_basebackup*. Može se koristiti za inicijalno kopiranje baze u svrhu *streaming* replikacije koristeći postojeći backup, ili za rekonstrukciju starog *standby* servera pomoću *delta* opcije. Neke od najvažnijih karakteristika alata *pgBackRest* uključuju:

* Paralelno backupovanje i vraćanje (*Parallel Backup & Restore*)
* Lokalni i udaljeni rad (*Local or Remote Operation*)
* Podršku za pune, inkrementalne i diferencijalne backup-e
* Proveru integriteta backup-a
* Nastavak nekompletnih backup procesa (*Backup Resume*)
* Delta vraćanje (*Delta Restore*)

## 3.4 Logički backup u PostgreSQL-u

Logički backup podrazumeva izvoz podataka u format koji je čitljiv čoveku, poput SQL naredbi. Ovaj tip backup-a je uglavnom fleksibilniji pa je koristan za reprodukciju baze podataka na drugoj arhitekturi (npr. za migracije). Međutim, vraćanje baze iz logičkog backup-a je prilično spor proces. Zato su praktični samo za migracije malih i srednjih PostgreSQL produkcionih baza.

Najčešći način za pravljenje logičkih backup-a i njihovo vraćanje je korišćenjem alata ***pg\_dump*** *i* ***pg\_restore****:*

***pg\_dump*** izdvaja PostgreSQL bazu podataka u skript fajl ili neki drugi arhivski fajl. Ovaj alat služi za pravljenje rezervnih kopija baza podataka. Alat pravi konzistentne bekapove čak i kada se baza koristi u tom trenutku. Čitaoci, pisci i drugi korisnici **neće biti blokirani** tokom korišćenja baze dok se *pg\_dump* izvršava. Ovom komandom se bekapuje **samo jedna baza podataka**. Ako postoji čitav klaster baza koji treba da bude bekapovan, koristi se komanda *pg\_dumpall*.

***pg\_restore*** se koristi za vraćanje baza podataka iz arhive kreirane komandom *pg\_dump*tj.uzima arhivski fajl kreiran komandom *pg\_dump* i vraća izabranu PostgreSQL bazu podataka. Kada se *pg\_dump* koristi sa jednim od formata koji nisu običan tekst, komanda za obnovu (*pg\_restore*) vraća bazu u stanje u kojem je bila prilikom pravljenja bekapa. Ovaj alat zna kako da izvrši komande u pravilnom redosledu kako bi se baza rekonstruisala tačno onako kako je bila sačuvana.   
Pošto je ovaj format prenosiv između različitih arhitektura, korisnik može odabrati šta će biti vraćeno i kojim redosledom.   
Opcije za rad sa podacima zavise od toga kako je izvorni fajl generisan. Komanda ne može ponovo generisati podatke koji nisu u fajlu i ne može promeniti prirodu komandi koje su korišćene za njegovo kreiranje.

3.5 Korišćenje logičkih naspram fizičkih backup-a u PostgreSQL-u   
  
Logički backup je najkorisniji za pravljenje test baza ili za migracije baza podataka. Što se tiče migracija, ukoliko se sa produkcionom bazom, preporučuje se korišćenje pg\_dump/pg\_restore samo ako je baza mala (manja od 100 GB). Migracija većih i složenijih baza pomoću pg\_dump/pg\_restore može na duze vreme da izbaci produkcionu bazu iz funkcije.

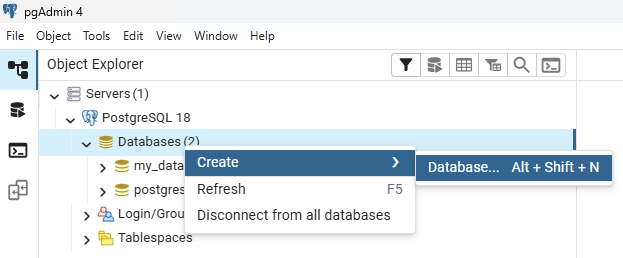
Fizički backup se uglavnom koristi za oporavak od nekog prekida i arhiviranje podataka. Ako se radi sa produkcionom bazom, važno je održavnje fizičkog backup-a ažurnim, kao i WAL zapisa, kako bi povratak baze u slučaju kvara bio moguć.

U sistemima se uglavnom koristi kombinacija logičkih i fizičkih backup-a. Za oporavak, fizički backup će biti osnovna linija odbrane, dok logički backup služi kao dodatna sigurnost.

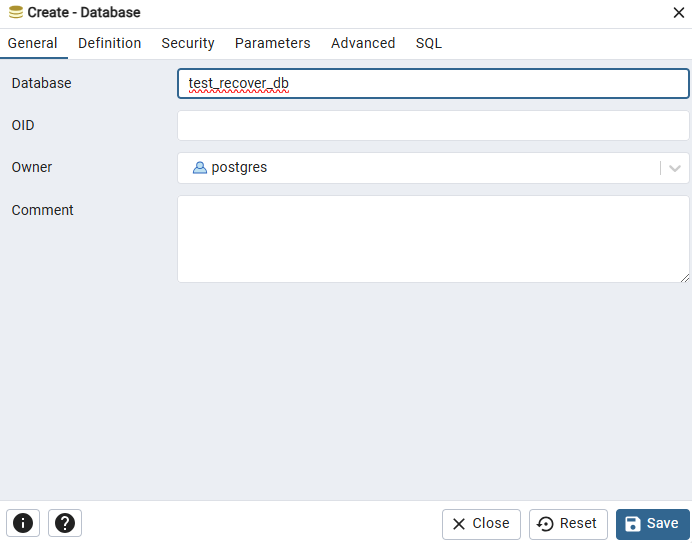
# Praktični primer: Backup i oporavak podataka u PostgreSQL-u

Sledeći primer će demonstrirati oporavak baze podataka u PostgreSQL-u korišćenjem *pgAdmin*-a. *PgAdmin* zapravo predstavlja grafički korisnički interfejs (*GUI*) i najčešći je izbor korisnika kojima je potreban vizuelni alat za upravljanje bazom.   
*PgAdmin* omogućava izvršavanje svih ključnih administrativnih zadataka nad PostgreSQL bazom podataka. Najnovija verzija, *pgAdmin 4*, predstavlja kompletnu rekonstrukciju originalnog alata *pgAdmin* i razvijena je korišćenjem kombinacije *JavaScript/jQuery i Python* tehnologija. *pgAdmin 4* pruža okruženje koje vizuelno i funkcionalno podseća na klasične desktop alate. Korisnički interfejs je intuitivan, pa je učenje i korišćenje ovog alata veoma jednostavno. [6]

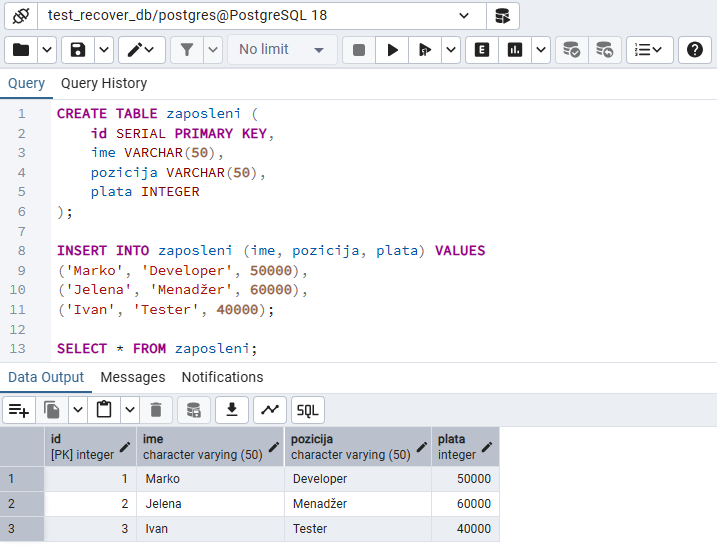
Prvo je potrebno kreirati bazu podataka (slika br. 3) koja će poslužiti za demonstraciju oporavka. Bazu u *pgAdmin*-u kreiramo tako što u okviru *Object Explorer*-a, kliknemo desni klik na *Databases* i odaberemo opciju *Create*, zatim opciju *Database*.

  
*Slika br. 3 – Kreiranje baze podataka*

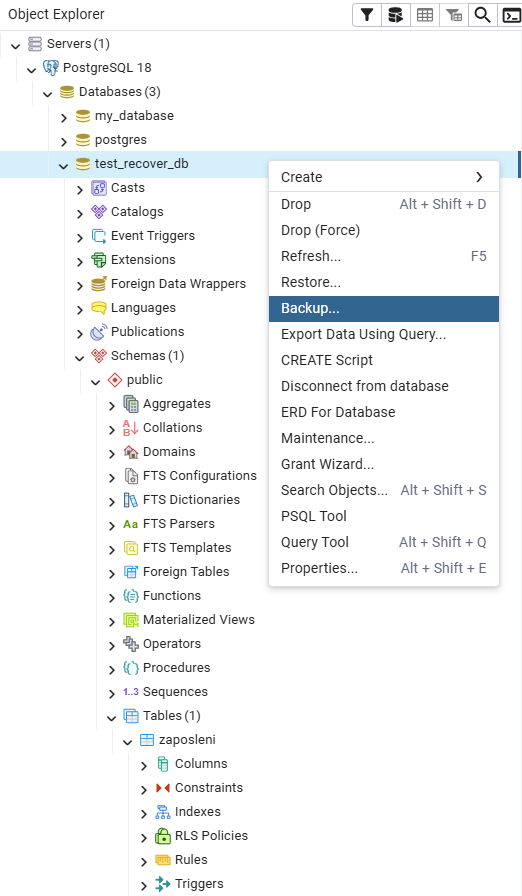
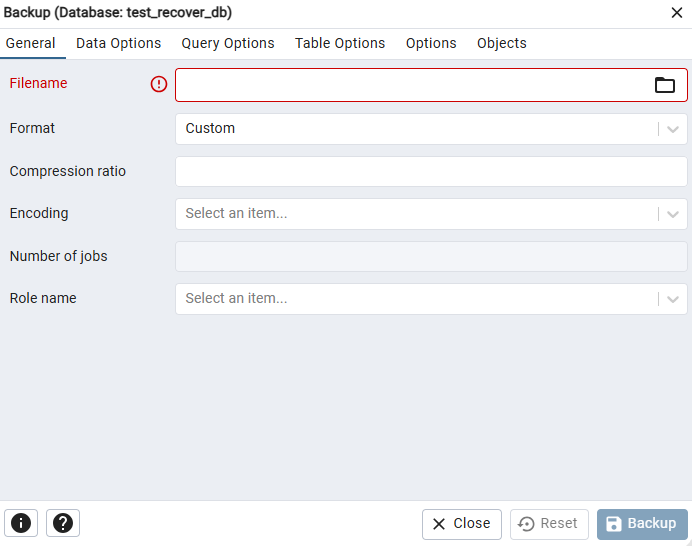
Zatim će se pojaviti forma koja zahteva da imenujemo bazu podataka (slika br. 4). Baza podataka koja će se koristiti za demonstraciju nazvana je *test\_recover\_db*.

  
*Slika br. 4 – Imenovanje baze podataka*

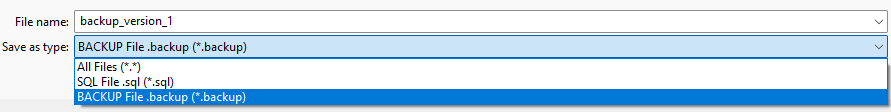
Potrebno je kreirati *zaposleni* (slika br. 5), koja predstavlja jednostavan primer realnog entiteta u poslovnom sistemu. Tabela je definisana korišćenjem SQL naredbi za kreiranje tabele (*CREATE TABLE*) i popunjavanje početnih podataka (*INSERT INTO*).

  
*Slika br. 5 – Kreiranje tabele zaposleni*

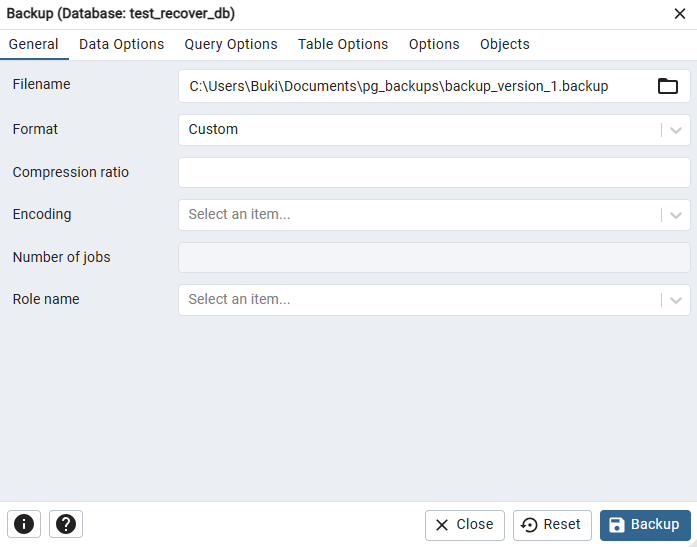
Koristeći pgAdmin, backup se može odraditi vrlo jednostavno i potrebno je ispratiti nekoliko koraka.   
Prvo ćemo u okviru Object Explorer-a kliknuti desni klik na bazu podataka koju smo kreirali i odabrati *Backup* (slika br. 6).   
Nakon toga, pojaviće se prozor prikazan na slici br. 7, koji zahteva da unesemo naziv fajla, što podrazumeva da treba da odaberemo putanju na kojoj ćemo sačuvati naš backup fajl. Što se formata tiče, korisniku je ponuđeno nekoliko formata backup-a, od kojih svaki ima svoju namenu i način upotrebe. Najčešće se koristi takozvani *Custom* format, koji se čuva sa ekstenzijom *.backup*. Ovaj format je internog PostgreSQL tipa, omogućava kompresiju podataka i pruža potpunu kontrolu prilikom oporavka. Njegova najveća prednost je što omogućava selektivan restore, odnosno vraćanje samo određenih objekata, kao što su pojedinačne tabele ili čitave šeme.   
Pored Custom formata, postoji i *Tar* format, koji čuva bekap u obliku TAR arhive. Ovaj format je pogodniji za premeštanje baze na drugi server ili čuvanje kompletne kopije baze, ali ne pruža istu fleksibilnost prilikom vraćanja, jer ne omogućava izdvajanje pojedinačnih objekata. Treći format je *Plain*, koji generiše običan SQL skript. U tom slučaju bekap se čuva kao tekstualni fajl koji sadrži sve SQL naredbe potrebne za kreiranje strukture baze i unos podataka. Plain format nije moguće vraćati pomoću *pg\_restore* alata, već se izvršava ručno preko Query Tool-a, zbog čega se najčešće koristi za pregled strukture baze ili za migracije. PostgreSQL podržava i Directory format, koji čuva rezervnu kopiju u vidu posebnog direktorijuma sa više fajlova. Za razliku od jedinstvenog .backup ili .tar fajla, Directory format raspodeljuje sadržaj baze na više delova, što omogućava paralelni bekap i oporavak, posebno kod većih baza. Ovaj format se takođe koristi uz *pg\_restore*, ali zahteva da se bekap izvršava na fajl sistemu, a ne kao jedan kompaktni fajl.

*****Slika br 6. – Prvi korak prilikom kreiranja backup-a* *****Slika br. 7 – Drugi korak prilikom kreiranja backup-a*

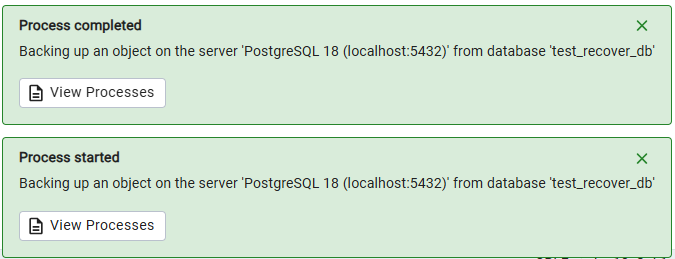
Nakon što se u *pgAdmin*-u otvori prozor za kreiranje bekapa i odabere željena putanja za čuvanje fajla, neophodno je uneti i naziv fajla. Takođe, moguće je odabrati i ekstenziju pod kojom će se backup sačuvati (*.backup* ili *.sql),* što je prikazano na slici br. 8. Iako oba formata čuvaju podatke baze, razlika između njih je značajna. Fajl sa ekstenzijom *.backup* predstavlja binarni format namenjen alatu *pg\_restore*, jer omogućava kontrolu nad oporavkom.   
Sa druge strane, ukoliko se backup sačuva sa ekstenzijom *.sql*, to će biti obična tekstualna skripta koji sadrži SQL naredbe za kreiranje baze i unos podataka. Ovaj format ne omogućava parcijalni restore i ne koristi *pg\_restore*, već se izvršava ručno kroz Query Tool.

  
*Slika br. 8 – Treći korak prilikom kreiranja backup-a*

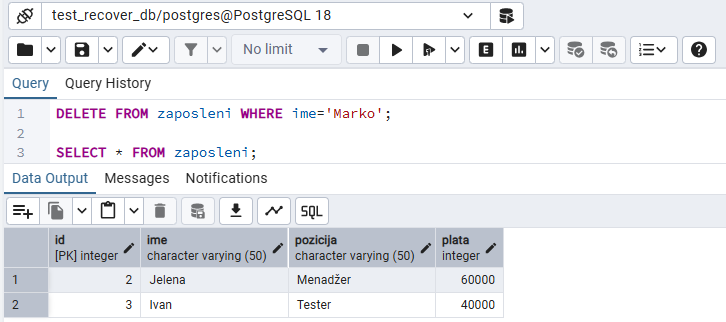
Na slici br. 9 prikazana je putanja na kojoj se naš backup fajl nalazi, kao i naziv samog backup fajla zajedno sa odabranom ekstenzijom, i izabrani format backup-a.

  
*Slika br. 9 – Četvrti korak prilikom kreiranja backup-a*

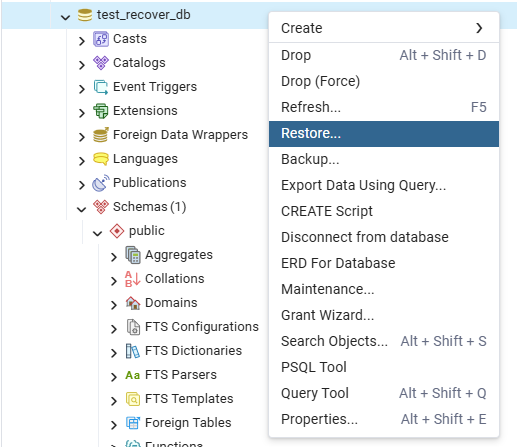
U donjem desnom uglu, pojaviće se prozori koji će nas obavestiti o tome da je backup uspešno obavljen (slika br. 10).

*****Slika br. 10 – Potvrda da je backup uspešno obavljen*

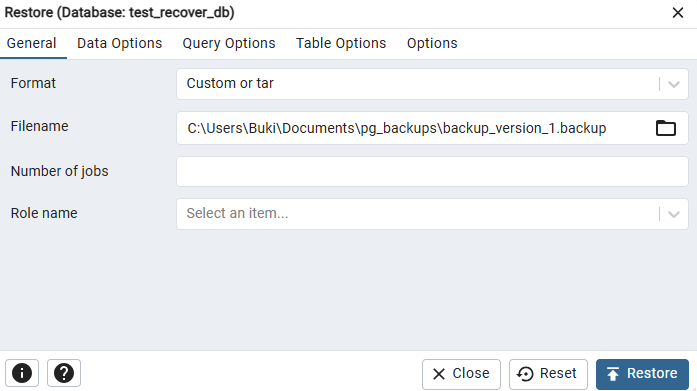
Slika br. 11 prikazuje izmenu tabele *zaposleni*. Obrisaćemo radnika koji se zove Marko nakon što je upit uspešno izvršen, možemo videti kako tabela *zaposleni* sada izgleda.

  
*Slika br. 11 – Izmena tabele zaposleni*

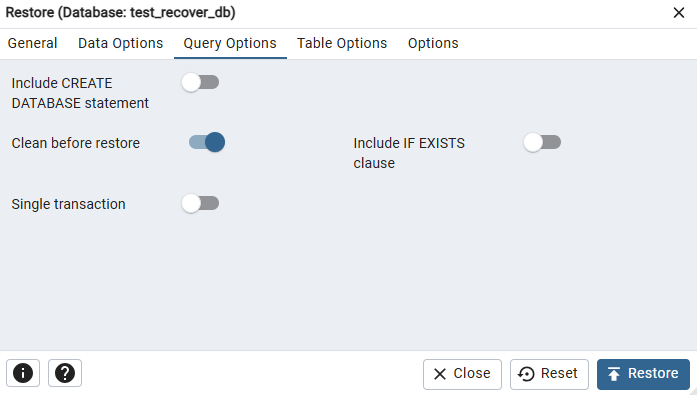
Recimo da je napravljena greška, jer je iz baze obrisan radnik Marko, i sada je potrebno vratiti bazu podataka u pređašnje stanje. U ovakvim situacijama koristi se prethodno napravljeni backup kako bi se izvršio oporavak baze. Prvi korak podrazumeva pronalaženje baze nad kojom želimo da izvršimo oporavak. Desnim klikom na naziv baze otvara se meni u kojem se bira opcija Restore (slika br. 12).

*  
Slika br. 12 – Prvi korak oporavka baze podataka*

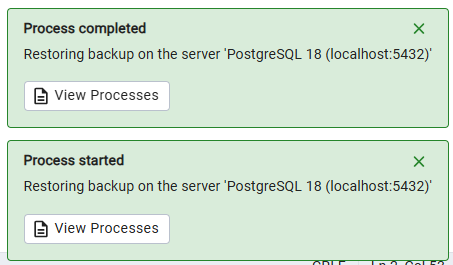
Zatim će se otvoriti prozor u kojem je potrebno odabrati format backup-a i učitati prethodno kreirani backup fajl. Budući da je korišćen *Custom* format, *pgAdmin* automatski prepoznaje tip fajla i priprema ga za oporavak.

  
*Slika br. 13 -* *Drugi korak oporavka baze podataka*

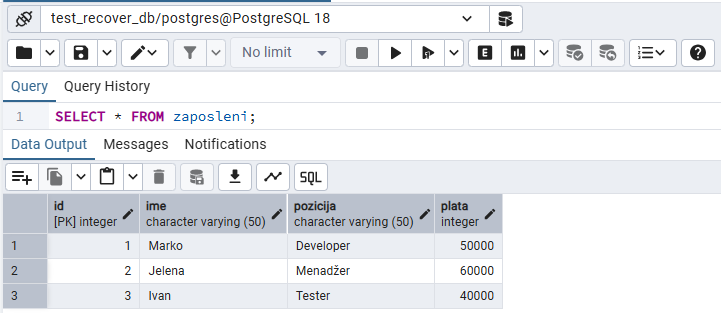
U ovom koraku moguće je uključiti opciju ***Clean before restore* (slika br. 14)*.*** Kada je ova opcija uključena, *pgAdmin* pre nego što učita podatke iz backup fajla, najpre briše sve postojeće objekte u bazi koji se poklapaju sa onima u backup-u. To uključuje tabele, šeme, funkcije, sekvence i sve druge elemente koji bi mogli da izazovu konflikt ili dupliranje podataka. Ova funkcionalnost je posebno korisna kada želimo da baza bude vraćena u tačno stanje iz trenutka kreiranja bekapa.   
Bez uključene opcije Clean before restore, *pgAdmin* će pokušati da izvrši restore nad postojećim objektima, što može dovesti do konflikata i grešaka, naročito ako su struktura tabele ili podaci promenjeni nakon kreiranja backup-a.

  
*Slika br. 14 – Uključivanje opcije Clean before restore*

U donjem desnom uglu, pojaviće se prozori koji će nas obavestiti o tome da je restore uspešno obavljen (slika br. 15).

  
*Slika br. 15 – Potvrda da je restore uspešno obavljen*

Kada je restore baze podataka obavljen i pokrenemo upit *SELECT \* FROM zaposleni;* (slika br. 16), možemo videti da je baza podataka uspešno vraćena u prethodno stanje.

*****Slika br. 16 – Izgled tabele zaposleni nakon restore-a baze podataka*

U ovom praktičnom prikazano je kako se pravi **logički backup baze podataka** koristeći *pgAdmin*. Kada se aktivira opcija Backup i odabere format .backup, u pozadini se automatski pokrenuo alat ***pg\_dump***, koji je kreirao rezervnu kopiju baze. Nakon izvršenih izmena u tabeli, (brisanje jednog zaposlenog, kako bismo simulirali grešku), aktivira se opcija Restore, učitava prethodno kreirani backup fajl. U pozadini se tada pokreće alat ***pg\_restore***, koji će vratiti bazu u stanje pre izmene. Na taj način demonstrirano je kako logički backup omogućava vraćanje baze u prethodno stanje, uključujući poništavanje grešaka poput slučajnog brisanja podataka.

# Zaključak

Može se zaključiti da očuvanje integriteta i dostupnosti podataka predstavlja jedan od najvažnijih aspekata modernog upravljanja informacionim sistemima. U savremenom poslovanju, podaci nisu samo sredstvo za svakodnevne operacije, već i ključni resurs čija vrednost direktno utiče na efikasnost i stabilnost organizacije.

PostgreSQL pruža moćne i fleksibilne mehanizme za zaštitu podataka, koji omogućavaju sigurno čuvanje i pouzdano vraćanje baze u slučaju grešaka, nenamernog brisanja ili drugih nepredviđenih situacija. Kroz pravilno planirane procedure backup-a i oporavka, moguće je ne samo sačuvati podatke, već i osigurati kontinuitet poslovanja i minimalizovati rizik od trajnog gubitka informacija. Ovi mehanizmi omogućavaju brzo i efikasno vraćanje baze u prethodno stanje, što doprinosi stabilnosti sistema i sigurnosti podataka.

Pored tehničke zaštite, procesi backup-a i recovery-ja imaju i stratešku vrednost, jer obezbeđuju organizacijama mogućnost da pouzdano upravljaju informacijama, planiraju kontinuitet poslovanja i reaguju na kritične situacije sa sigurnošću i kontrolom. Stoga, implementacija pouzdanih metoda za backup i oporavak u PostgreSQL-u predstavlja neophodnu meru u svakom savremenom informacionom sistemu.

# 6. Literatura

[1] <https://www.geeksforgeeks.org/dbms/why-recovery-is-needed-in-dbms/>

[2] <https://www.sciencedirect.com/topics/computer-science/database-recovery>

[3] <https://www.geeksforgeeks.org/dbms/difference-between-backup-and-recovery/>

[4] <https://blog.algomaster.io/p/postgresql-internal-architecture>

[5] <https://www.postgresql.org/docs/current/app-pgbasebackup.html>

[6] <https://severalnines.com/blog/how-use-pgbackrest-backup-postgresql-and-timescaledb/>

[7] <https://www.adservio.fr/post/what-is-pgadmin>

# 