



UNIVERZITET U NIŠU  
ELEKTRONSKI FAKULTET



**BACKUP/RESTORE POSTGRESQL BAZE PODATAKA**  
**-Seminarski rad-**

Student:  
Sara Savić, br.ind. 1758

Mentor:  
Prof. dr Aleksandar Stanimirović

Niš, maj 2024.

# SAŽETAK

Backup i Restore predstavljaju ključne elemente u kontekstu baza podataka. Poznavanje ovih tehnika pruža sigurnosnu 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-ovanja 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 arhiviranje i oporavak u određenoj tački vremena. Naposljetku, 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
2.BACKUP/RESTORE KOD RELACIONIH BAZA PODATAKA.....	5
2.1 Definicije i opis pojmova Backup/Restore kod relacionih baza podataka.....	5
3.BACKUP/RESTORE KOD POSTGRESQL BAZE PODATAKA .....	7
3.1 SQL Dump.....	7
3.1.1 Obnavljanje/Vraćanje (Restoring) Dump-a .....	10
3.1.2 pg_dumpall .....	15
3.1.3 Rad sa velikim bazama podataka .....	16
3.2 Backup na nivou fajl sistema .....	22
3.3 Kontinuirano arhiviranje i obnova do trenutka (PITR).....	25
3.3.1 Postavljanje WAL arhiviranja.....	26
3.3.2 Pravljenje osnovnog backup-a(base backup) .....	29
3.3.3 Pravljenje osnovnog backup-a(base backup) korišćenjem niskog nivoa API-ja .....	30
3.3.4. Oporavak u tački vremena-Point-In-Time Recovery .....	34
3.3.5. Vremenske linije .....	39
4. KREIRANJE BACKUP-A I RESTORE-A KORIŠĆENJEM pgAdmin 4 ALATA .....	42
5.ZAKLJUČAK.....	45
6.SPISAK KORIŠĆENE LITERATURE.....	46

## 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 ozbiljnim 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 zaštititi. 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 transakcionih fajlova, omogućavajući sveobuhvatno vraćanje podataka bez gubitka informacija. S druge strane, inkrementalna kopija je idealna za skoro obavljene transakcije 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 struktura 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 sigurnosni 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 konzistentno stanje. Proces obnove podataka uključuje obnavljanje najnovije kopije baze podataka i primenu transakcionih sigurnosnih 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.

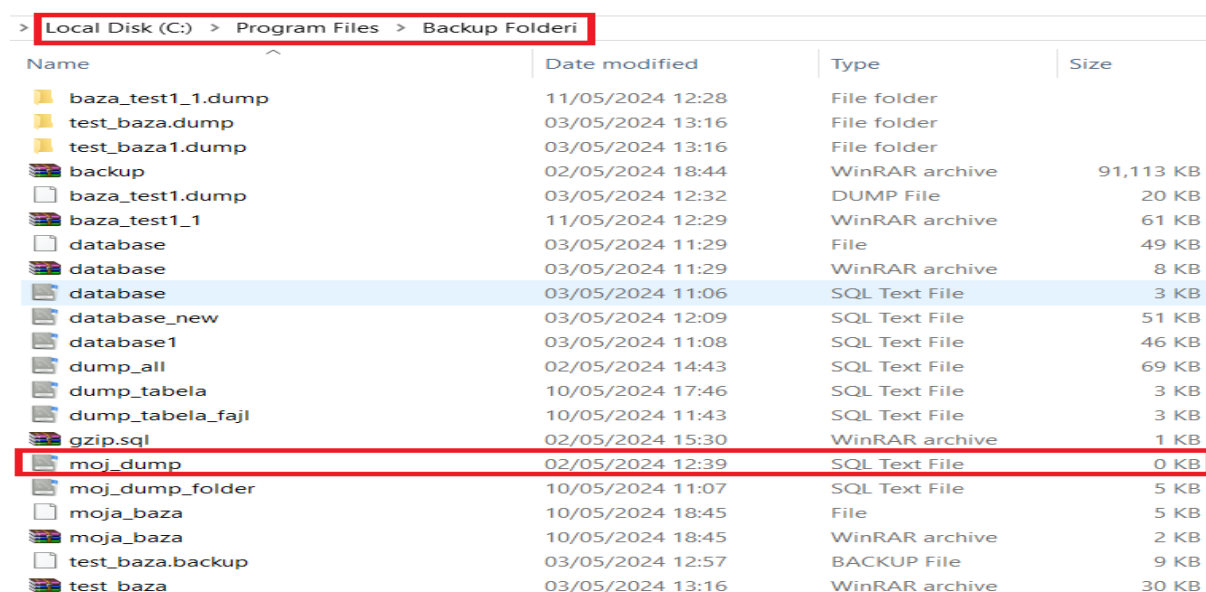
#### 3.1 SQL Dump

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:

```
C:\Program Files\PostgreSQL\16\bin>pg_dump moja_baza > "C:\Program Files\Backup Folderi\moj_dump.sql"
```

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".



> Local Disk (C:) > Program Files > Backup Folderi			
Name	Date modified	Type	Size
baza_test1_1.dump	11/05/2024 12:28	File folder	
test_baza.dump	03/05/2024 13:16	File folder	
test_baza1.dump	03/05/2024 13:16	File folder	
backup	02/05/2024 18:44	WinRAR archive	91,113 KB
baza_test1.dump	03/05/2024 12:32	DUMP File	20 KB
baza_test1_1	11/05/2024 12:29	WinRAR archive	61 KB
database	03/05/2024 11:29	File	49 KB
database	03/05/2024 11:29	WinRAR archive	8 KB
database	03/05/2024 11:06	SQL Text File	3 KB
database_new	03/05/2024 12:09	SQL Text File	51 KB
database1	03/05/2024 11:08	SQL Text File	46 KB
dump_all	02/05/2024 14:43	SQL Text File	69 KB
dump_tabela	10/05/2024 17:46	SQL Text File	3 KB
dump_tabela_fajl	10/05/2024 11:43	SQL Text File	3 KB
gzip.sql	02/05/2024 15:30	WinRAR archive	1 KB
moj_dump	02/05/2024 12:39	SQL Text File	0 KB
moj_dump_folder	10/05/2024 11:07	SQL Text File	5 KB
moja_baza	10/05/2024 18:45	File	5 KB
moja_baza	10/05/2024 18:45	WinRAR archive	2 KB
test_baza.backup	03/05/2024 12:57	BACKUP File	9 KB
test_baza	03/05/2024 13:16	WinRAR archive	30 KB

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.

```

1  -- PostgreSQL database dump
2  --
3  --
4  --
5  -- Dumped from database version 16.2
6  -- Dumped by pg_dump version 16.2
7  --
8  SET statement_timeout = 0;
9  SET lock_timeout = 0;
10 SET idle_in_transaction_session_timeout = 0;
11 SET client_encoding = 'UTF8';
12 SET standard_conforming_strings = on;
13 SELECT pg_catalog.set_config('search_path', '', false);
14 SET check_function_bodies = false;
15 SET xmloption = content;
16 SET client_min_messages = warning;
17 SET row_security = off;
18
19 SET default_tablespace = '';
20
21 SET default_table_access_method = heap;
22
23 --
24 -- Name: korisnici; Type: TABLE; Schema: public; Owner: postgres
25 --
26 CREATE TABLE public.korisnici (
27     korisnikid integer NOT NULL,
28     korisnickoime character varying(50) NOT NULL,
29     email character varying(100) NOT NULL,
30     sifra character varying(100) NOT NULL,
31     datumkreiranja timestamp without time zone DEFAULT CURRENT_TIMESTAMP
32 );
33
34 ALTER TABLE public.korisnici OWNER TO postgres;
35
36 --
37 -- Name: korisnici_korisnikid_seq; Type: SEQUENCE; Schema: public; Owner: postgres
38 --
39 CREATE SEQUENCE public.korisnici_korisnikid_seq
40     AS integer
41     START WITH 1
42
43
44
45
46 ALTER TABLE public.korisnici OWNER TO postgres;
47
48 --
49 -- Name: korisnici_korisnikid_seq; Type: SEQUENCE; Schema: public; Owner: postgres
50 --
51 CREATE SEQUENCE public.korisnici_korisnikid_seq
52     AS integer
53     START WITH 1
54     INCREMENT BY 1
55     NO MINVALUE
56     NO MAXVALUE
57     CACHE 1;
58
59 ALTER SEQUENCE public.korisnici_korisnikid_seq OWNER TO postgres;
60
61 --
62 -- Name: korisnici_korisnikid_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
63 --
64 ALTER SEQUENCE public.korisnici_korisnikid_seq OWNED BY public.korisnici.korisnikid;
65
66 --
67 -- Name: moja_tabela; Type: TABLE; Schema: public; Owner: postgres
68 --
69 CREATE TABLE public.moja_tabela (
70     id integer NOT NULL,
71     ime character varying(50),
72     prezime character varying(50)
73 );
74
75 ALTER TABLE public.moja_tabela OWNER TO postgres;
76
77 --
78 -- Name: moja_tabela_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
79 --
80 CREATE SEQUENCE public.moja_tabela_id_seq
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
25
```



pg\_dump svoj rezultat ispisuje na standardni 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.

```
C:\Program Files\PostgreSQL\16\bin>pg_dump -t moja_tabela moja_baza > "C:\Program Files\Backup Folderi\dump_tabela.sql"
```

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 sadrži SQL komande koje opisuju strukturu te tabele, uključujući definicije kolona, ograničenja, indeksa i drugih elementa koji se odnose samo na tu tabelu. Dodatno, sadrži i SQL komande za dodavanje podataka u tu tabelu. Sadržaj ovog fajla prikazan je na slici 5.

```
SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'WIN1252';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

SET default_tablespace = '';

SET default_table_access_method = heap;

--
-- Name: moja_tabela; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.moja_tabela (
    id integer NOT NULL,
    ime character varying(50),
    prezime character varying(50)
);

ALTER TABLE public.moja_tabela OWNER TO postgres;

--
-- Name: moja_tabela_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.moja_tabela_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER SEQUENCE public.moja_tabela_id_seq OWNER TO postgres;
```

```

ALTER SEQUENCE public.moja_tabela_id_seq OWNED BY public.moja_tabela.id;

--
-- Name: moja_tabela id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.moja_tabela ALTER COLUMN id SET DEFAULT nextval('public.moja_tabela_id_seq'::regclass);

--
-- Data for Name: moja_tabela; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY public.moja_tabela (id, ime, prezime) FROM stdin;
981 Ime981 Prezime981
982 Ime982 Prezime982
983 Ime983 Prezime983
984 Ime984 Prezime984
985 Ime985 Prezime985
986 Ime986 Prezime986
987 Ime987 Prezime987
988 Ime988 Prezime988
989 Ime989 Prezime989
990 Ime990 Prezime990
991 Ime991 Prezime991
992 Ime992 Prezime992
993 Ime993 Prezime993
994 Ime994 Prezime994
995 Ime995 Prezime995
996 Ime996 Prezime996
997 Ime997 Prezime997
998 Ime998 Prezime998
999 Ime999 Prezime999
1000 Ime1000 Prezime1000
\.

```

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”.

```
C:\Program Files\PostgreSQL\16\bin>pg_dump -h localhost -p 5432 moja_baza > "C:\Program Files\Backup Folderi\moj_dump.sql"
```

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 učitao 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.

Query
Query History

1 DELETE FROM moja\_tabela WHERE id <= 980;  
2

Data Output
Messages
Notifications

DELETE 980  
  
Query returned successfully in 66 msec.

Query
Query History

1 INSERT INTO moja\_tabela (ime, prezime) VALUES  
2 ('', 'Nevalidno prezime'),  
3 ('Nevalidno ime', NULL),  
4 ('123', '456'),  
5 ('Nevalidno ime', 'Nevalidno prezime');

Data Output
Messages
Notifications

INSERT 0 4  
  
Query returned successfully in 66 msec.

Query
Query History

1 DROP TABLE moja\_tabela1;  
2

Data Output
Messages
Notifications

DROP TABLE  
  
Query returned successfully in 79 msec.

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.

Query		Query History	
1		SELECT * FROM moja_tabela;	
Data Output		Messages	Notifications
	id [PK] integer	ime character varying (50)	prezime character varying (50)
1	1001		Nevalidno prezime
2	1002	Nevalidno ime	[null]
3	1003	123	456
4	1004	Nevalidno ime	Nevalidno prezime
5	981	Ime981	Prezime981
6	982	Ime982	Prezime982
7	983	Ime983	Prezime983
8	984	Ime984	Prezime984
9	985	Ime985	Prezime985
10	986	Ime986	Prezime986
11	987	Ime987	Prezime987
12	988	Ime988	Prezime988
13	989	Ime989	Prezime989
14	990	Ime990	Prezime990
15	991	Ime991	Prezime991
16	992	Ime992	Prezime992
17	993	Ime993	Prezime993
18	994	Ime994	Prezime994

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

Ova oštećenja zahtevaju brzu reakciju kako bi se sprečili dalji gubici 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.

```
C:\Program Files\PostgreSQL\16\bin>psql test_baza1 < moj_dump.sql
SET
SET
SET
SET
SET
set_config
-----
(1 row)

SET
SET
SET
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
CREATE SEQUENCE
ALTER SEQUENCE
ALTER TABLE
ALTER TABLE
COPY 0
COPY 0
setval
-----
1
(1 row)

setval
-----
1
(1 row)

ALTER TABLE
ALTER TABLE
```

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.

Query

Query History

1

SELECT \* FROM moja\_tabela;

Data Output

Messages

Notifications

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 transakcija, 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.

```

C:\Program Files\PostgreSQL\16\bin>psql --set ON_ERROR_STOP=on -1 moja_baza < "C:\Program Files\Backup Folderi\database_new.sql"
SET
SET
SET
SET
SET
set_config
-----
(1 row)

SET
SET
SET
SET
SET
SET
SET
2024-05-03 12:10:48.553 CEST [11140] ERROR: relation "korisnici" already exists
2024-05-03 12:10:48.553 CEST [11140] STATEMENT: CREATE TABLE public.korisnici (
    korisnikid integer NOT NULL,
    korisnickoime character varying(50) NOT NULL,
    email character varying(100) NOT NULL,
    sifra character varying(100) NOT NULL,
    datumkreiranja timestamp without time zone DEFAULT CURRENT_TIMESTAMP
);
ERROR: relation "korisnici" already exists

```

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 konkretnom 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.

```

Administrator: Command Prompt
C:\Program Files\PostgreSQL\16\bin>createdb -f template0 nova_baza
C:\Program Files\PostgreSQL\16\bin>pg_dump baza_test1 | psql nova_baza
SET
SET
SET
SET
set_config
-----
(1 row)

SET
SET
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
CREATE TABLE
ALTER TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
CREATE SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
ALTER TABLE
ALTER TABLE
COPY 10
COPY 1000
COPY 1000
setval
-----
10
(1 row)

setval
-----
1000
(1 row)

```

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 sigurnosna kopija baze “baza\_test1”. Korišćenjem takozvane cevi (“|”), ta sigurnosna 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\_baza”).

### 3.1.2 pg\_dumpall

pg\_dump dump-uje samo jednu bazu podataka 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.

```
C:\Program Files\PostgreSQL\16\bin>pg_dumpall > "C:\Program Files\Backup Folderi\dump_all.sql"
```

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:

```
C:\Program Files\PostgreSQL\16\bin>psql -f "C:\Program Files\Backup Folderi\dump_all.sql" postgres
```

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<sup>1</sup>. 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 konzistentna, 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.

```
C:\Program Files\PostgreSQL\16\bin>pg_dumpall --schema-only > definitions.sql  
C:\Program Files\PostgreSQL\16\bin>pg_dumpall --roles-only > roles.sql  
C:\Program Files\PostgreSQL\16\bin>pg_dumpall --tablespaces-only > tablespaces.sql
```

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

---

<sup>1</sup> postgres- uobičajena baza podataka u PostgreSQL okruženju koja se automatski kreira prilikom instalacije okruženja.

### 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:

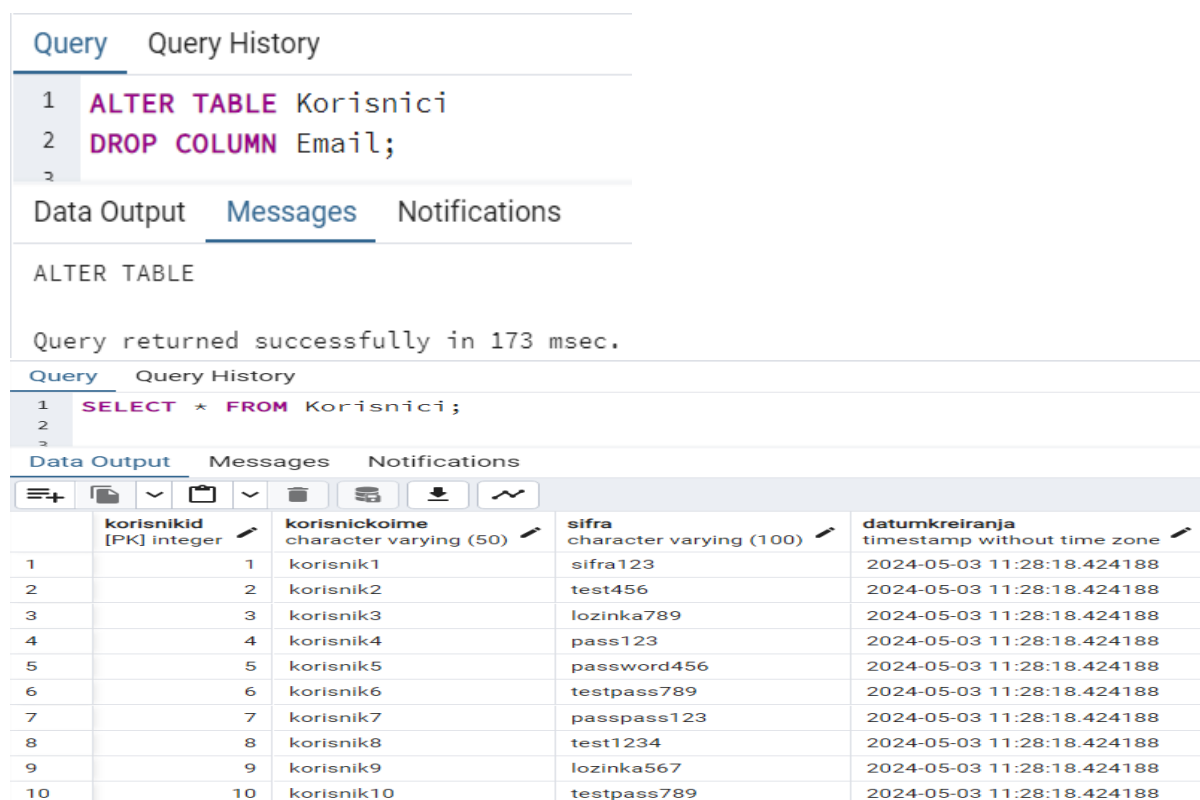
Moguće je izvršiti kompresiju dump-a korišćenjem odgovarajućeg programa [7].

```
C:\Program Files\PostgreSQL\16\bin>pg_dump -Z6 -d baza_test > "C:\Program Files\Backup Folderi\database.gz"
```

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.



The screenshot shows a database management interface. The top section displays a query window with the following SQL commands:

```
1 ALTER TABLE Korisnici
2 DROP COLUMN Email;
```

Below the query window, the 'Messages' tab shows a confirmation message: "Query returned successfully in 173 msec."

The bottom section shows the 'Query History' tab with the following SQL command:

```
1 SELECT * FROM Korisnici;
```

Below the query history, the 'Data Output' tab displays a table with the following data:

	korisnikid [PK] integer	korisnickoime character varying (50)	sifra character varying (100)	datumkreiranja timestamp without time zone
1	1	korisnik1	sifra123	2024-05-03 11:28:18.424188
2	2	korisnik2	test456	2024-05-03 11:28:18.424188
3	3	korisnik3	lozinka789	2024-05-03 11:28:18.424188
4	4	korisnik4	pass123	2024-05-03 11:28:18.424188
5	5	korisnik5	password456	2024-05-03 11:28:18.424188
6	6	korisnik6	testpass789	2024-05-03 11:28:18.424188
7	7	korisnik7	passpass123	2024-05-03 11:28:18.424188
8	8	korisnik8	test1234	2024-05-03 11:28:18.424188
9	9	korisnik9	lozinka567	2024-05-03 11:28:18.424188
10	10	korisnik10	testpass789	2024-05-03 11:28:18.424188

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).

```
C:\Program Files\PostgreSQL\16\bin>"C:\Program Files\7-Zip\7z.exe" e "C:\Program Files\Backup Folderi\database.gz" -o"C:\Program Files\Backup Folderi\"

7-Zip 22.01 (x64) : Copyright (c) 1999-2022 Igor Pavlov : 2022-07-15

Scanning the drive for archives:
1 file, 7972 bytes (8 KiB)

Extracting archive: C:\Program Files\Backup Folderi\database.gz
--
Path = C:\Program Files\Backup Folderi\database.gz
Type = gzip
Headers Size = 10

Everything is Ok

Size:          49270
Compressed: 7972
```

```
C:\Program Files\PostgreSQL\16\bin>psql -U postgres -d baza_test1 -f "C:\Program Files\Backup Folderi\database"
SET
SET
SET
SET
SET
  set_config
-----
(1 row)

SET
SET
SET
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
CREATE SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
ALTER TABLE
ALTER TABLE
ALTER TABLE
COPY 10
COPY 1000
COPY 1000
  setval
-----
      10
(1 row)

  setval
-----
    1000
(1 row)
```

Query		Query History			
1		SELECT * FROM Korisnici;			
Data Output		Messages Notifications			
	korisnikid [PK] integer	korisnickoime character varying (50)	email character varying (100)	sifra character varying (100)	datumkreiranja timestamp without time zone
1	1	korisnik1	korisnik1@example.com	sifra123	2024-05-03 11:28:18.424188
2	2	korisnik2	korisnik2@example.com	test456	2024-05-03 11:28:18.424188
3	3	korisnik3	korisnik3@example.com	lozinka789	2024-05-03 11:28:18.424188
4	4	korisnik4	korisnik4@example.com	pass123	2024-05-03 11:28:18.424188
5	5	korisnik5	korisnik5@example.com	password456	2024-05-03 11:28:18.424188
6	6	korisnik6	korisnik6@example.com	testpass789	2024-05-03 11:28:18.424188
7	7	korisnik7	korisnik7@example.com	passpass123	2024-05-03 11:28:18.424188
8	8	korisnik8	korisnik8@example.com	test1234	2024-05-03 11:28:18.424188
9	9	korisnik9	korisnik9@example.com	lozinka567	2024-05-03 11:28:18.424188
10	10	korisnik10	korisnik10@example.com	testpass789	2024-05-03 11:28:18.424188

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

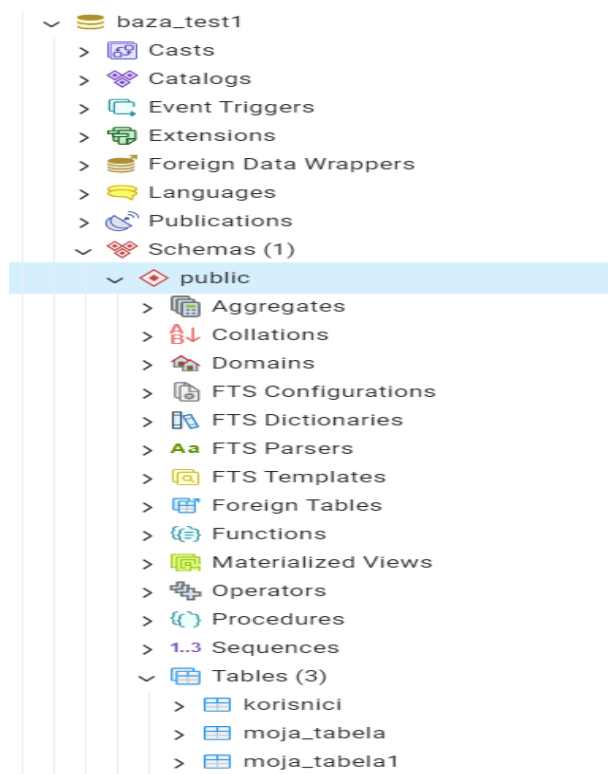
Još jedna opcija, koja rešava problem velikih izlaznih fajlova generisanih od strane pg\_dump-a, jeste korišćenje takozvanog “prilagođenog formata” (*eng.custom format*), formata koji je podrazumevano arhiviran (bez dodatnih koraka) i pruža značajnu fleksibilnost, posebno prilikom uvoza [7].

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

```
C:\Program Files\PostgreSQL\16\bin>pg_dump -Fc -d baza_test1 > "C:\Program Files\Backup Folderi\baza_test1.dump"
```

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”

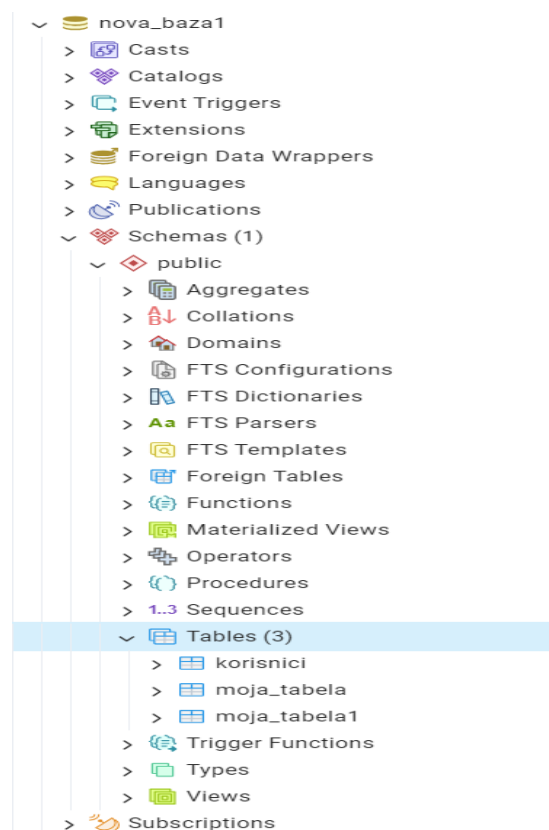
Ovaj format datoteke koristi se sa pg\_restore komandom kako bi se ponovo izgradila baza podataka. To omogućava pg\_restore-u da bude selektivan u vezi sa tim šta se obnavlja, ili čak da preuredi stavke pre obnavljanja. Arhivirani formati datoteka namenjeni su prenosu podataka između različitih arhitektura. 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]

```
C:\Program Files\PostgreSQL\16\bin>createdb -T template0 nova_baza1  
C:\Program Files\PostgreSQL\16\bin>pg_restore -j 4 -d nova_baza1 "C:\Program Files\Backup Folder\1\baza_test1.dump"
```

Slika 21-Prikaz obnove baze "baza\_test1" korišćenjem opcije -j. Opcija -j 4 označava da će se proces obnavljanja izvršiti 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. Prilikom kreiranja arhive u ovakvom formatu, svaka tabela i "izbačeni blob"<sup>2</sup>(*eng.dumped blob*) će biti smešteni u posebnu datoteku unutar novog direktorijuma.Osim toga, biće generisana i tabela sadržaja

<sup>2</sup> "Izbačeni blob" se u ovom kontekstu odnosi na binarne podatke koji su ekstrahovani iz baze podataka i smešteni u posebne datoteke unutar direktorijuma koji se koristi za arhiviranje.

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 kompatibilan sa direktorijumskim formatom: važeća arhiva u direktorijumskom formatu nastaje kada se ekstrahuje 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.

```
C:\Program Files\PostgreSQL\16\bin>pg_dump -Fd -t "C:\Program Files\Backup Folder\ baza_test1_1.dump" baza_test1
C:\Program Files\PostgreSQL\16\bin>pg_dump -Ft -f "C:\Program Files\Backup Folder\ baza_test1_1.tar" baza_test1
```

Slika 23-Kreiranje direktorijumske i tar datoteke

This PC > Local Disk (C:) > Program Files > Backup Folder\

Name	Date modified	Type	Size
baza_test1_1.dump	11/05/2024 12:28	File folder	
test_baza.dump	03/05/2024 13:16	File folder	
test_baza1.dump	03/05/2024 13:16	File folder	
backup	02/05/2024 18:44	WinRAR archive	91,113 KB
baza_test1.dump	03/05/2024 12:32	DUMP File	20 KB
baza_test1_1	11/05/2024 12:29	WinRAR archive	61 KB
database	03/05/2024 11:29	File	49 KB
database	03/05/2024 11:29	WinRAR archive	8 KB
database	03/05/2024 11:06	SQL Text File	3 KB
database_new	03/05/2024 12:09	SQL Text File	51 KB
database1	03/05/2024 11:08	SQL Text File	46 KB
dump_all	02/05/2024 14:43	SQL Text File	69 KB
dump_tabela	10/05/2024 17:46	SQL Text File	3 KB
dump_tabela_fajl	10/05/2024 11:43	SQL Text File	3 KB
gzip.sql	02/05/2024 15:30	WinRAR archive	1 KB
moj_dump	02/05/2024 12:39	SQL Text File	0 KB
moj_dump_folder	10/05/2024 11:07	SQL Text File	5 KB
moja_baza	10/05/2024 18:45	File	5 KB
moja_baza	10/05/2024 18:45	WinRAR archive	2 KB
test_baza.backup	03/05/2024 12:57	BACKUP File	9 KB
test_baza	03/05/2024 13:16	WinRAR archive	30 KB

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

4799.dat	11/05/2024 12:28	WinRAR archive	1 KB
4801.dat	11/05/2024 12:28	WinRAR archive	6 KB
4802.dat	11/05/2024 12:28	WinRAR archive	6 KB
toc.dat	11/05/2024 12:28	DAT File	7 KB

4799.dat	722	722	DAT File	11/05/2024 12:...
4801.dat	21,684	21,684	DAT File	11/05/2024 12:...
4802.dat	21,684	21,684	DAT File	11/05/2024 12:...
restore.sql	6,245	6,245	SQL Text File	11/05/2024 12:...
toc.dat	7,107	7,107	DAT File	11/05/2024 12:...

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:

```
C:\Program Files\PostgreSQL\16\bin>createdb -T template0 baza_oporavak1

C:\Program Files\PostgreSQL\16\bin>
C:\Program Files\PostgreSQL\16\bin>pg_restore -d baza_oporavak1 -v "C:\Program Files\Backup Folderi\baza_test1_1.dump"
pg_restore: connecting to database for restore
pg_restore: creating TABLE "public.korisnici"
pg_restore: creating SEQUENCE "public.korisnici_korisnikid_seq"
pg_restore: creating SEQUENCE OWNED BY "public.korisnici_korisnikid_seq"
pg_restore: creating TABLE "public.moja_tabela"
pg_restore: creating TABLE "public.moja_tabela1"
pg_restore: creating SEQUENCE "public.moja_tabela1_id_seq"
pg_restore: creating SEQUENCE OWNED BY "public.moja_tabela1_id_seq"
pg_restore: creating SEQUENCE "public.moja_tabela_id_seq"
pg_restore: creating SEQUENCE OWNED BY "public.moja_tabela_id_seq"
pg_restore: creating DEFAULT "public.korisnici_korisnikid"
pg_restore: creating DEFAULT "public.moja_tabela id"
pg_restore: creating DEFAULT "public.moja_tabela1 id"
pg_restore: processing data for table "public.korisnici"
pg_restore: processing data for table "public.moja_tabela"
pg_restore: processing data for table "public.moja_tabela1"
pg_restore: executing SEQUENCE SET korisnici_korisnikid_seq
pg_restore: executing SEQUENCE SET moja_tabela1_id_seq
pg_restore: executing SEQUENCE SET moja_tabela_id_seq
pg_restore: creating CONSTRAINT "public.korisnici_korisnici_email_key"
pg_restore: creating CONSTRAINT "public.korisnici_korisnici_pkey"
pg_restore: creating CONSTRAINT "public.moja_tabela1_moja_tabela1_pkey"
pg_restore: creating CONSTRAINT "public.moja_tabela_moja_tabela_pkey"
```

```
C:\Program Files\PostgreSQL\16\bin>createdb -T template0 baza_oporavak2

C:\Program Files\PostgreSQL\16\bin>
C:\Program Files\PostgreSQL\16\bin>pg_restore -d baza_oporavak2 -v "C:\Program Files\Backup Folderi\baza_test1_1.tar"      1 file(s) copied.

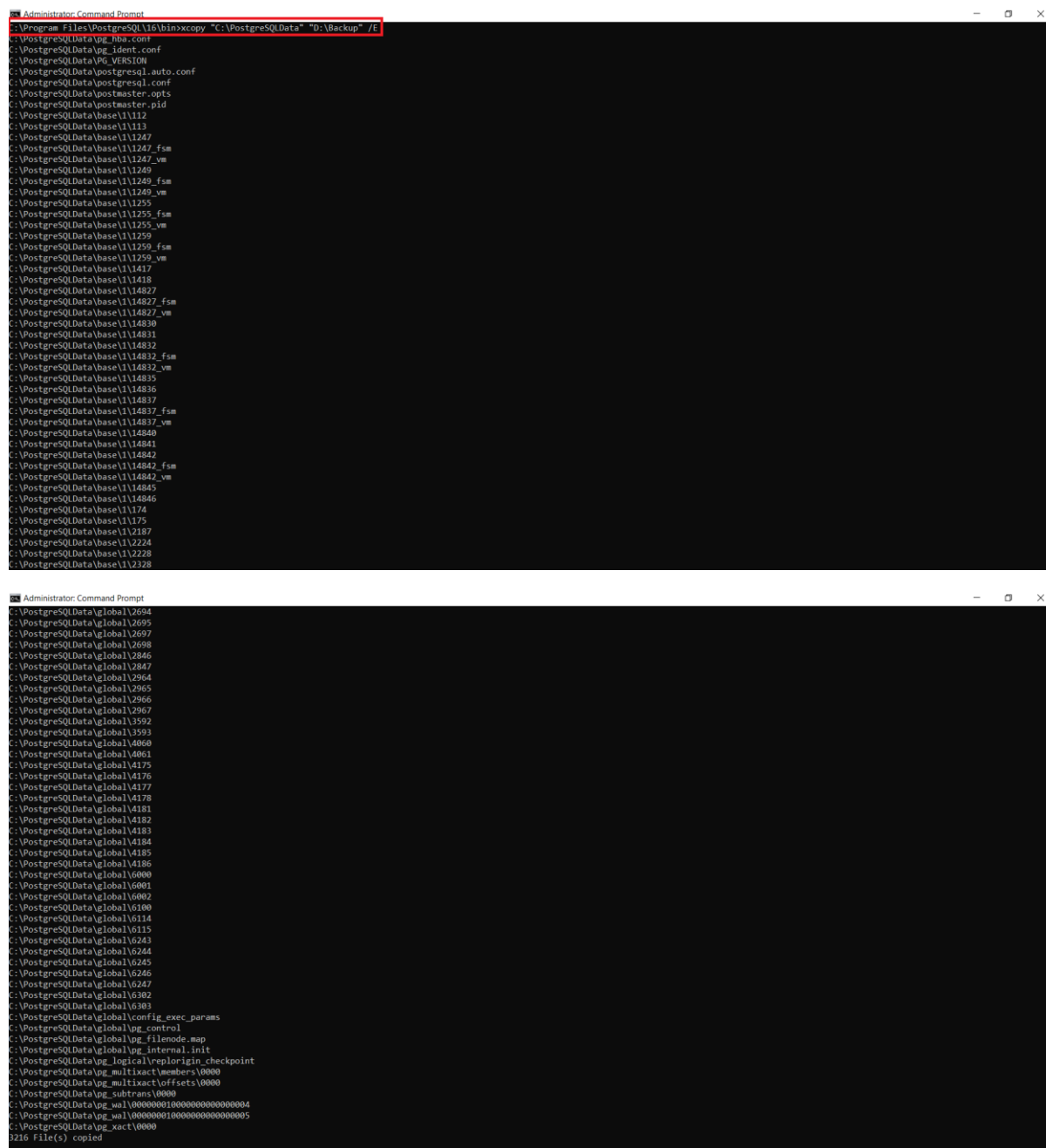
pg_restore: connecting to database for restore
pg_restore: creating TABLE "public.korisnici"
pg_restore: creating SEQUENCE "public.korisnici_korisnikid_seq"
pg_restore: creating SEQUENCE OWNED BY "public.korisnici_korisnikid_seq"
pg_restore: creating TABLE "public.moja_tabela"
pg_restore: creating TABLE "public.moja_tabela1"
pg_restore: creating SEQUENCE "public.moja_tabela1_id_seq"
pg_restore: creating SEQUENCE OWNED BY "public.moja_tabela1_id_seq"
pg_restore: creating SEQUENCE "public.moja_tabela_id_seq"
pg_restore: creating SEQUENCE OWNED BY "public.moja_tabela_id_seq"
pg_restore: creating DEFAULT "public.korisnici_korisnikid"
pg_restore: creating DEFAULT "public.moja_tabela id"
pg_restore: creating DEFAULT "public.moja_tabela1 id"
pg_restore: processing data for table "public.korisnici"
pg_restore: processing data for table "public.moja_tabela"
pg_restore: processing data for table "public.moja_tabela1"
pg_restore: executing SEQUENCE SET korisnici_korisnikid_seq
pg_restore: executing SEQUENCE SET moja_tabela1_id_seq
pg_restore: executing SEQUENCE SET moja_tabela_id_seq
pg_restore: creating CONSTRAINT "public.korisnici_korisnici_email_key"
pg_restore: creating CONSTRAINT "public.korisnici_korisnici_pkey"
pg_restore: creating CONSTRAINT "public.moja_tabela1_moja_tabela1_pkey"
pg_restore: creating CONSTRAINT "public.moja_tabela_moja_tabela_pkey"
```

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 sigurnosne 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.

Name	Date modified	Type	Size
base	03/05/2024 13:26	File folder	
global	06/05/2024 17:28	File folder	
pg_commit_ts	08/04/2024 17:05	File folder	
pg_dynshmem	08/04/2024 17:05	File folder	
pg_logical	06/05/2024 17:29	File folder	
pg_multixact	03/05/2024 13:26	File folder	
pg_notify	08/04/2024 17:05	File folder	
pg_repslot	08/04/2024 17:05	File folder	
pg_serial	08/04/2024 17:05	File folder	
pg_snapshots	08/04/2024 17:05	File folder	
pg_stat	06/05/2024 17:29	File folder	
pg_stat_tmp	08/04/2024 17:05	File folder	
pg_subtrans	03/05/2024 13:26	File folder	
pg_tblspc	08/04/2024 17:05	File folder	
pg_twophase	08/04/2024 17:05	File folder	
pg_wal	03/05/2024 13:26	File folder	
pg_xact	03/05/2024 13:26	File folder	
pg_hba	08/04/2024 17:05	CONF File	6 KB
pg_ident	08/04/2024 17:05	CONF File	3 KB
PG_VERSION	08/04/2024 17:05	File	1 KB
postgresql.auto	08/04/2024 17:05	CONF File	1 KB
postgresql	06/05/2024 16:51	CONF File	30 KB
postmaster.opts	06/05/2024 16:55	OPTS File	1 KB

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 sigurnosnu 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 sigurnosne kopije fajl sistema moguće samo za kompletnu sigurnosnu kopiju i obnovu čitavog klastera baze podataka.

Alternativni pristup sigurnosnoj 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 sigurnosno 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<sup>3</sup>. Ipak, ovo ne predstavlja problem, samo zahteva da WAL log datoteka bude uključena u sigurnosnu kopiju. Takođe, pre pravljenja snimka (*eng.snapshot*), potrebno je izvršiti CHECKPOINT<sup>4</sup> 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 sigurnosno 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 kontinuiranog arhiviranja baze podataka (biće opisano u potpoglavlju 3.3) jer su takve kopije otporne na promene u fajl sistemu tokom procesa sigurnosnog kopiranja. Za to je potrebno omogućiti kontinuirano arhiviranje samo tokom procesa sigurnosnog kopiranja, a obnavljanje se vrši korišćenjem kontinuiranog oporavka iz arhive.

Još jedna opcija jeste korišćenje *rsync* alata za obnavljanje sigurnosne 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 sigurnosne kopije fajl sistema sa minimalnim prekidom rada. [8]

---

<sup>3</sup> WAL log je skraćenica od Write-Ahead Logging koji predstavlja standardnu metodu za osiguravanje integriteta podataka.

<sup>4</sup> CHECKPOINT-Tačka u sekvenci WAL loga na kojoj su svi podaci ažurirani tako da održavaju informacije iz loga.



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 “hvataći” trenutno stanje sistema datoteka prilikom arhiviranja.
- 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 “tope 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 sačuvan u bazi podataka) i mogu biti bezbedno “reciklirane”. [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<sup>5</sup>, mogu se zapisivati na tape drive-u<sup>6</sup> (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 funkcija.

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 arhiviranje 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 arhiviranje, 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.

---

<sup>5</sup> 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.

<sup>6</sup> Tape drive je uređaj koji čuva podatke računara na magnetnoj traci, posebno u svrhu backup-ovanja.

```
#wal_level = replica          # minimal, replica, or logical
                              # (change requires restart)
#fsync = on                   # flush data to disk for crash safety
                              # (turning this off can cause
                              # unrecoverable data corruption)
#synchronous_commit = on     # synchronization level;
                              # off, local, remote_write, remote_apply, or on
#wal_sync_method = fsync      # the default is the first option
                              # supported by the operating system:
                              #   open_datasync
                              #   fdatasync (default on Linux and FreeBSD)
                              #   fsync
                              #   fsync_writethrough
                              #   open_sync

archive_mode = on             # enables archiving; off, on, or always
                              # (change requires restart)
#archive_library = ''         # library to use to archive a WAL file
                              # (empty string indicates archive_command should
                              # be used)
#archive_command = 'copy "%p" "D:\Arhivica\%f"' # command to use to archive a WAL file
                              # placeholders: %p = path of file to archive
                              #           %f = file name only
                              # e.g. 'test ! -f /mnt/server/archivedir/%f && cp %p /mnt/server/archivedir/%f'
#archive_timeout = 0          # force a WAL file switch after this
                              # number of seconds; 0 disables
```

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

Na slici 30 prikazano je kako se postavljaju parametri u konfiguracionim fajlu 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).

This PC > DATA (D:) > Arhivica

Name	Date modified	Type	Size
000000010000000000000002B	13/05/2024 10:56	File	16,384 KB
000000010000000000000002C	13/05/2024 10:59	File	16,384 KB
000000010000000000000002D	13/05/2024 11:00	File	16,384 KB
000000010000000000000002E	13/05/2024 11:00	File	16,384 KB
000000010000000000000002F	13/05/2024 11:00	File	16,384 KB
0000000100000000000000004	08/05/2024 11:23	File	16,384 KB
0000000100000000000000005	08/05/2024 11:23	File	16,384 KB
0000000100000000000000030	13/05/2024 11:00	File	16,384 KB
0000000100000000000000031	13/05/2024 11:00	File	16,384 KB
0000000100000000000000032	13/05/2024 11:00	File	16,384 KB

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 '000000010000000000000002B', 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 zaštititi 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 arhiviranja 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 sigurnosna 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 sigurnosne 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, povećaćće 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 podataka 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 arhiviranje kreira se osnovni backup. Najjednostavniji način za kreiranje osnovnog backup-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 segmentnoj 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]

```
C:\Program Files\PostgreSQL\16\bin>pg_basebackup -D "D:\Basebackup"
2024-05-08 12:10:39.225 CEST [18556] LOG:  checkpoint starting: force wait
2024-05-08 12:10:39.226 CEST [17004] WARNING: archive_mode enabled, yet archiving is not configured
2024-05-08 12:10:39.331 CEST [18556] LOG:  checkpoint complete: wrote 0 buffers (0.0%); 0 WAL file(s) added, 0 removed,
0 recycled; write=0.001 s, sync=0.001 s, total=0.106 s; sync files=0, longest=0.000 s, average=0.000 s; distance=16383 k
B, estimate=16383 kB; lsn=0/60000060, redo lsn=0/60000028
```

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.

This PC > DATA (D:) > Basebackup

Name	Date modified	Type	Size
base	13/05/2024 11:19	File folder	
global	13/05/2024 11:19	File folder	
log	13/05/2024 11:19	File folder	
pg_commit_ts	13/05/2024 11:19	File folder	
pg_dynshmem	13/05/2024 11:19	File folder	
pg_logical	13/05/2024 11:19	File folder	
pg_multixact	13/05/2024 11:19	File folder	
pg_notify	13/05/2024 11:19	File folder	
pg_replslot	13/05/2024 11:19	File folder	
pg_serial	13/05/2024 11:19	File folder	
pg_snapshots	13/05/2024 11:19	File folder	
pg_stat	13/05/2024 11:19	File folder	
pg_stat_tmp	13/05/2024 11:19	File folder	
pg_subtrans	13/05/2024 11:19	File folder	
pg_tblspc	13/05/2024 11:19	File folder	
pg_twophase	13/05/2024 11:19	File folder	
pg_wal	13/05/2024 11:19	File folder	
pg_xact	13/05/2024 11:19	File folder	
backup_label	13/05/2024 11:19	File	1 KB
backup_manifest	13/05/2024 11:19	File	760 KB
pg_hba	13/05/2024 11:19	CONF File	6 KB
pg_ident	13/05/2024 11:19	CONF File	3 KB
PG_VERSION	13/05/2024 11:19	File	1 KB
postgresql.auto	13/05/2024 11:19	CONF File	1 KB
postgresql	13/05/2024 11:19	CONF File	30 KB

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

This PC > Local Disk (C:) > PostgreSQLData > pg\_wal

Name	Date modified	Type	Size
archive_status	13/05/2024 11:24	File folder	
000000010000000000000034.00000028.backup	13/05/2024 11:19	BACKUP File	1 KB
000000010000000000000035	13/05/2024 11:00	File	16,384 KB
000000010000000000000036	13/05/2024 11:00	File	16,384 KB
000000010000000000000037	13/05/2024 11:00	File	16,384 KB
000000010000000000000038	13/05/2024 11:19	File	16,384 KB
000000010000000000000039	13/05/2024 11:19	File	16,384 KB

```

START WAL LOCATION: 0/34000028 (file 000000010000000000000034)
STOP WAL LOCATION: 0/34000138 (file 000000010000000000000034)
CHECKPOINT LOCATION: 0/34000060
BACKUP METHOD: streamed
BACKUP FROM: primary
START TIME: 2024-05-13 11:19:14 CEST
LABEL: pg_basebackup base backup
START TIMELINE: 1
STOP TIME: 2024-05-13 11:19:21 CEST
STOP TIMELINE: 1

```

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:

```
postgres=# SELECT pg_backup_start(label => 'MojBackup', fast => false);
2024-05-18 18:49:28.303 CEST [8380] LOG:  checkpoint starting: force wait
2024-05-18 18:49:28.331 CEST [8380] LOG:  checkpoint complete: wrote 2 buffers (0.0%); 0 WAL file(s) added, 0 removed, 1
recycled; write=0.003 s, sync=0.003 s, total=0.028 s; sync files=2, longest=0.002 s, average=0.002 s; distance=16384 kB
, estimate=16384 kB; lsn=0/80000060, redo lsn=0/80000028
pg_backup_start
-----
0/80000028
(1 row)

postgres=# 1 file(s) copied.
```

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 checkpoint-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 elementarnih polja koja su prikazana na slici 36.

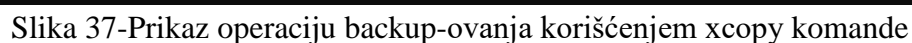
```
START WAL LOCATION: 0/80000028 (file 000000010000000000000080)
CHECKPOINT LOCATION: 0/80000060
BACKUP METHOD: streamed
BACKUP FROM: primary
START TIME: 2024-05-18 18:49:28 CEST
LABEL: MojBackup
START TIMELINE: 1
```

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<sup>7</sup> 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

<sup>7</sup> LSN-Broj redosleda zapisivanja, pokazivač na lokaciju u WAL logu.

3. 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.



- ```
postgres=# SELECT * FROM pg_backup_stop(wait_for_archive => true);
 1 file(s) copied.
 1 file(s) copied.
NOTICE: all required WAL segments have been archived
lsn | labelfile | spcmaphile
-----+-----+-----
0/80000138 | START WAL LOCATION: 0/80000028 (file 000000010000000000000080)+
| CHECKPOINT LOCATION: 0/80000060 +
| BACKUP METHOD: streamed +
| BACKUP FROM: primary +
| START TIME: 2024-05-18 18:49:28 CEST +
| LABEL: MojBackup +
| START TIMELINE: 1 +
(1 row)
```

32



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<sup>8</sup> 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).

```
START WAL LOCATION: 0/80000028 (file 00000001000000000000000080)
STOP WAL LOCATION: 0/80000138 (file 00000001000000000000000080)
CHECKPOINT LOCATION: 0/80000060
BACKUP METHOD: streamed
BACKUP FROM: primary
START TIME: 2024-05-18 18:49:28 CEST
LABEL: MojBackup
START TIMELINE: 1
STOP TIME: 2024-05-18 18:51:47 CEST
STOP TIMELINE: 1
```

Slika 39-Prikaz sadržaja datoteke istorije vremenske linije

- 5) 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.

5. 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 skript 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

---

<sup>8</sup> XLOG-skraćenica za transaction log, odnosno zapisnik koji čuva informacije o promenama izvršenim nad podacima u bazi tokom transakcija.

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]

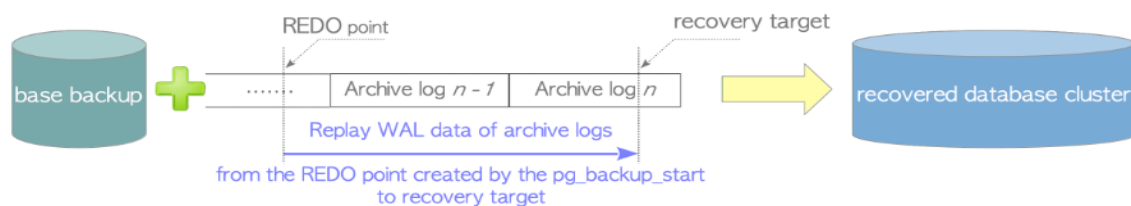
Sve datoteke ili direktorijumi koji počinju sa `pgsql_tmp` mogu biti izostavljeni iz rezervne kopije, jer se ove datoteke automatski uklanjaju prilikom pokretanja postmaster-a, a direktorijumi će biti ponovo kreirani po potrebi. Datoteke `pg_internal.init` mogu biti izostavljane iz rezervne kopije kada se pronade 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, PostgreSQL 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<sup>9</sup> tačke koju stvara osnovni backup, do tačke koju je potrebno oporaviti. Tačka oporavka, u ovom slučaju, naziva se cilj oporavka (*eng. recovery target*).

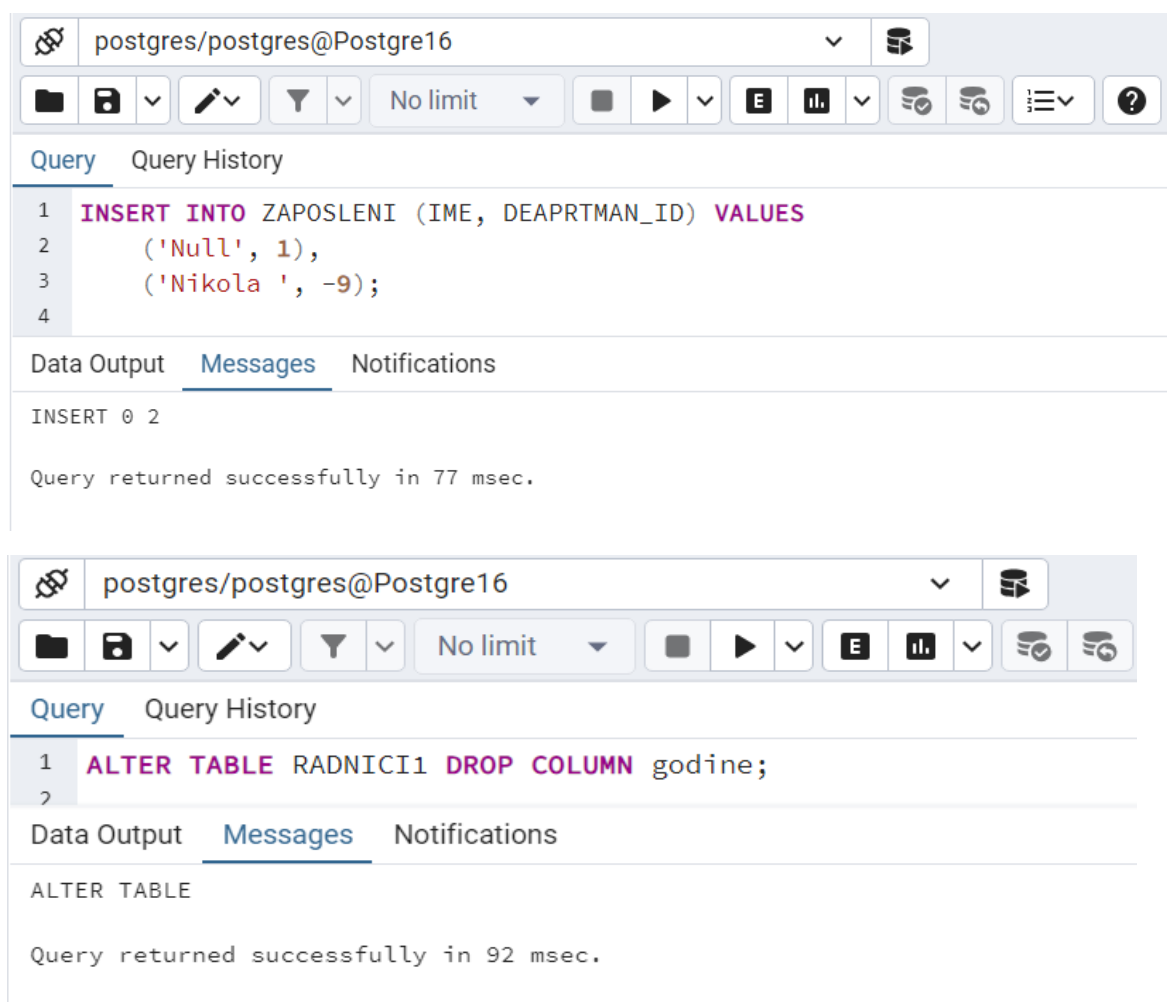


<sup>9</sup> REDO tačka je lokacija XLOG zapisa koji je zabeležen u trenutku kada je pokrenut poslednji checkpoint.

#### Slika 40-Prikaz osnovnog koncepta PITR-a<sup>10</sup>

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 vremenke 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`).

<sup>10</sup> Izvor slike- <https://www.interdb.jp/pg/pgsql10/02.html>

postgres/postgres@Postgre16

Query Query History

```
1 SELECT * FROM ZAPOSLENI;
```

Data Output Messages Notifications

|    | zaposleni_id<br>[PK] integer | ime<br>character varying (100) | deaprtman_id<br>integer |
|----|------------------------------|--------------------------------|-------------------------|
| 1  | 2                            | Nikola Brankovic               | 2                       |
| 2  | 3                            | Ivan Misic                     | 5                       |
| 3  | 4                            | Lara Krstic                    | 3                       |
| 4  | 6                            | Petar Zivic                    | 4                       |
| 5  | 7                            | Jovan Jovanovic                | 5                       |
| 6  | 8                            | Luna Jaksic                    | 4                       |
| 7  | 12                           | Jovan Milic                    | 1                       |
| 8  | 13                           | Jovan Illic                    | 6                       |
| 9  | 14                           | Masa Masic                     | 8                       |
| 10 | 15                           | Null                           | 1                       |
| 11 | 16                           | Nikola                         | -9                      |

postgres/postgres@Postgre16

Query Query History

```
1 SELECT * FROM RADNICI1;
```

Data Output Messages Notifications

|   | id<br>[PK] integer | ime<br>text | prezime<br>text | adresa<br>character | plata<br>real |
|---|--------------------|-------------|-----------------|---------------------|---------------|
| 1 | 1                  | Pavle       | Illic           | Nis                 | 50000         |
| 2 | 2                  | Nina        | Nesic           | Novi Sad            | 40000         |
| 3 | 3                  | Milos       | Misic           | Beograd             | 15000         |
| 4 | 4                  | Milos       | Ivkic           | Beograd             | 30000         |

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 povрати 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.

```

C:\Program Files\PostgreSQL\16\bin>pg_ctl stop -D C:\PostgreSQLData
2024-05-18 18:05:55.043 CEST [14124] LOG:  received fast shutdown request
waiting for server to shut down...2024-05-18 18:05:55.046 CEST [14124] LOG:  aborting any active transactions
2024-05-18 18:05:55.054 CEST [14124] LOG:  background worker "logical replication launcher" (PID 11472) exited with exit code 1
2024-05-18 18:05:55.055 CEST [11956] LOG:  shutting down
2024-05-18 18:05:55.202 CEST [11956] LOG:  checkpoint starting: shutdown immediate
2024-05-18 18:05:55.221 CEST [11956] LOG:  checkpoint complete: wrote 0 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.001 s, sync=0.001 s, total=0.023 s; sync files=0, longest=0.000 s, average=0.000 s; distance=16356 kB, estimate=16406 kB; lsn=0/85000028, redo lsn=0/85000028
1 file(s) copied.
2024-05-18 18:05:55.474 CEST [14124] LOG:  database system is shut down.
done
server stopped

```

Slika 43-Korak 1:Zaustavljanje glavnog servera baze podataka

```

#restore_command = 'cp "D:\Arhivica\%f" %p' # command to use to restore an archived WAL file
# placeholders: %p = path of file to restore
#               %f = file name only
# e.g. 'cp /mnt/server/archivedir/%f %p'
#archive_cleanup_command = '' # command to execute at every restartpoint
#recovery_end_command = '' # command to execute at completion of recovery

# - Recovery Target -
# Set these only when performing a targeted recovery.

#recovery_target = '' # 'immediate' to end recovery as soon as a
#                     # consistent state is reached
# (change requires restart)
#recovery_target_name = '' # the named restore point to which recovery will proceed
# (change requires restart)
#recovery_target_time = '' # the time stamp up to which recovery will proceed
# (change requires restart)
#recovery_target_xid = '' # the transaction ID up to which recovery will proceed
# (change requires restart)
#recovery_target_lsn = '' # the WAL LSN up to which recovery will proceed
# (change requires restart)
#recovery_target_inclusive = on # Specifies whether to stop:
# just after the specified recovery target (on)
# just before the recovery target (off)
# (change requires restart)
#recovery_target_timeline = 'latest' # 'current', 'latest', or timeline ID
# (change requires restart)
#recovery_target_action = 'pause' # 'pause', 'promote', 'shutdown'
# (change requires restart)

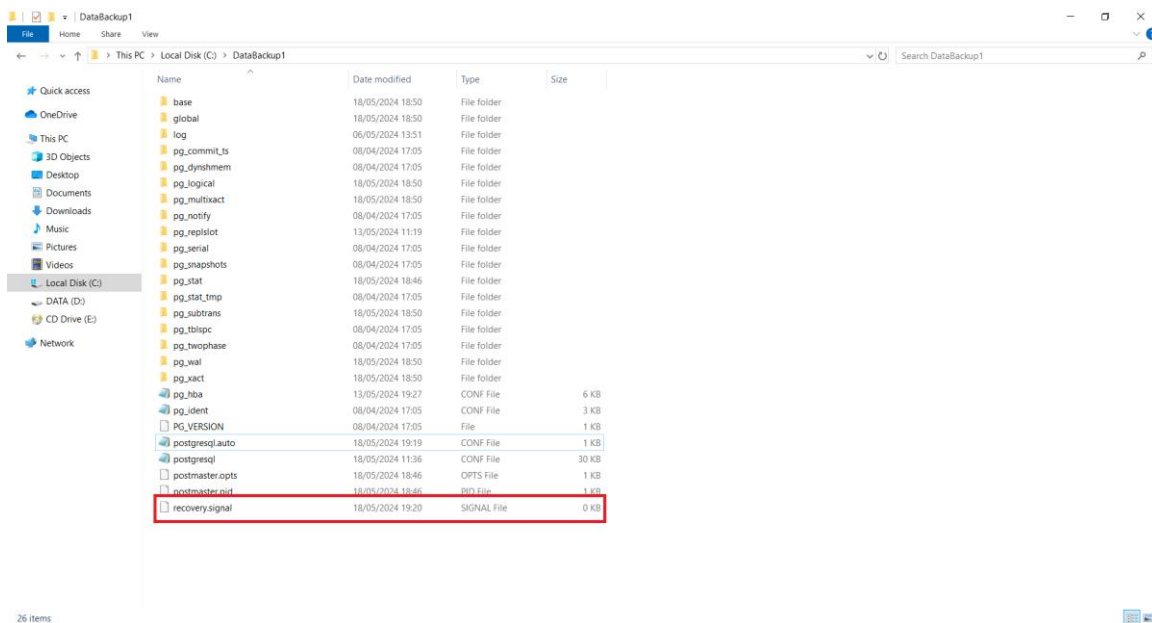
```

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.

```

C:\Program Files\PostgreSQL\16\bin>type nul > "C:\DataBackup1\recovery.signal"

```



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 direktorijumu backup-a klastera baze podataka), kako je i prikazano na slici 45.

```

C:\Program Files\PostgreSQL\16\bin>pg_ctl start
pg_ctl: another server might be running; trying to start anyway
waiting for server to start...2024-05-18 19:21:02.414 CEST [10272] LOG: starting PostgreSQL 16.2, compiled by Visual C++ build 1937, 64-bit
2024-05-18 19:21:02.422 CEST [10272] LOG: listening on IPv6 address "::1", port 5433
2024-05-18 19:21:02.422 CEST [10272] LOG: listening on IPv4 address "127.0.0.1", port 5433
2024-05-18 19:21:02.495 CEST [10716] LOG: database system was interrupted; last known up at 2024-05-18 18:49:28 CEST
... 'cp' is not recognized as an internal or external command,
operable program or batch file.
2024-05-18 19:21:05.843 CEST [10716] LOG: starting archive recovery
2024-05-18 19:21:05.844 CEST [10716] LOG: database system was not properly shut down; automatic recovery in progress
2024-05-18 19:21:05.860 CEST [10716] LOG: redo starts at 0/800000028
2024-05-18 19:21:05.870 CEST [10716] LOG: invalid record length at 0/800000110: expected at least 24, got 0
2024-05-18 19:21:05.873 CEST [10716] LOG: consistent recovery state reached at 0/800000110
2024-05-18 19:21:05.874 CEST [10272] LOG: database system is ready to accept read-only connections
'cp' is not recognized as an internal or external command,
operable program or batch file.
2024-05-18 19:21:05.900 CEST [10716] LOG: redo done at 0/8000000D8 system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.04 s
'cp' is not recognized as an internal or external command,
operable program or batch file.
done
server started

C:\Program Files\PostgreSQL\16\bin>'cp' is not recognized as an internal or external command,
operable program or batch file.
2024-05-18 19:21:05.992 CEST [10716] LOG: selected new timeline ID: 2
'cp' is not recognized as an internal or external command,
operable program or batch file.
2024-05-18 19:21:06.116 CEST [10716] LOG: archive recovery complete
2024-05-18 19:21:06.125 CEST [11104] LOG: checkpoint starting: end-of-recovery immediate wait
2024-05-18 19:21:06.146 CEST [11104] LOG: checkpoint complete: wrote 2 buffers (0.0%); 0 WAL file(s) added, 0 removed, 1 recycled; write=0.002 s, sync=0.003 s, total=0.026 s; sync files=2, longest=0.002 s, average=0.002 s; distance=0 kB, estimate=0 kB; lun=0/800000110, redo lun=0/800000110
2024-05-18 19:21:06.178 CEST [10272] LOG: database system is ready to accept connections

```

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).

Query

Query History

1 SELECT \* FROM ZAPOSLENI ;

Data Output

Messages

Notifications

<

Query

Query History

1

SELECT \* FROM RADNICI1;

Data Output

Messages

Notifications

|   | id<br>[PK] integer | ime<br>text | prezime<br>text | adresa<br>character | plata<br>real | godine<br>integer |
|---|--------------------|-------------|-----------------|---------------------|---------------|-------------------|
| 1 | 1                  | Pavle       | Ilic            | Nis                 | 50000         | 32                |
| 2 | 2                  | Nina        | Nesic           | Novi Sad            | 40000         | 26                |
| 3 | 3                  | Milos       | Misic           | Beograd             | 15000         | 25                |
| 4 | 4                  | Milos       | Ivkic           | Beograd             | 30000         | 30                |

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.

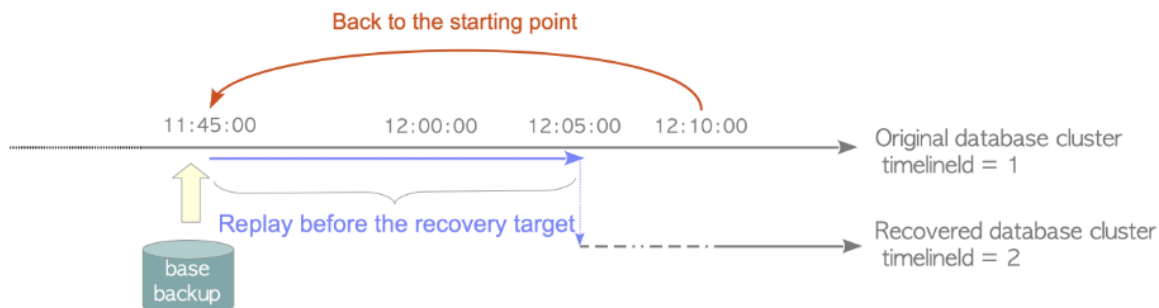
Oporavak baze podataka obuhvatiće sve dostupne segmente, vraćajući bazu podataka na određenu tačku u vremenu, ukoliko je ta tačka navedena u konfiguracionom fajlu. U slučaju da tačka nije navedena, oporavak će se odvijati do trenutne tačke vremena ili što je moguće bliže toj tački. Kada se definiše tačka 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.

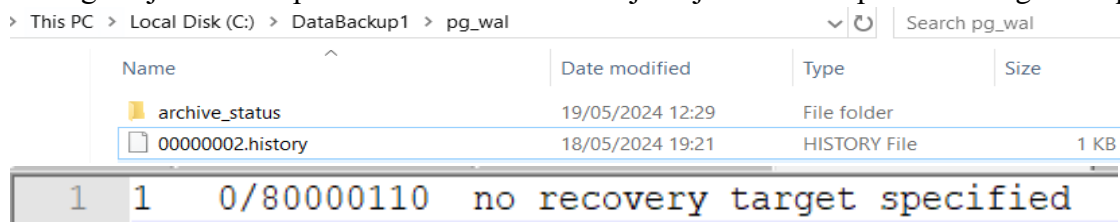
Kako bi rešio ovaj problem, PostgreSQL koristi koncept vremenskih linija (*eng. timelines*). Svaki put kada se završi proces arhivskog oporavka, kreira se nova vremenska linija koja označava seriju WAL zapisa generisanih nakon tog oporavka. Identifikacioni broj vremenske linije je deo imena datoteka 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<sup>11</sup>

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 `0000000100000000000000000009` i `0000000100000000000000000000A`, redom, novooporavljeni klaster baze podataka će dobiti `timelineid 2`, i PostgreSQL kreira WAL segment `0000000200000000000000000000A`.

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.



<sup>11</sup> Izvor slike-<https://www.interdb.jp/pg/pgsql10/03.html>



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

```
#recovery_target_timeline = '2' # 'current', 'latest', or timeline ID
# (change requires restart)
```

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

```
postgres=# insert into departmani (naziv) values('null');
INSERT 0 1
postgres=# select * from departmani;
 departman_id |      naziv
-----+-----
          1 | HR
          2 | Marketing
          3 | Finansije
          4 | Informacione tehnologije
          5 | Pravna sluzba
          6 | Operacije
          7 | Tehnicka sluzba
          8 | null
(8 rows)
```

Slika 50-Prikaz simulacije oštećenja tabele DEPARTMANI

```
C:\Program Files\PostgreSQL\16\bin>pg_ctl start -D C:\DataBackup\
waiting for server to start...2024-05-19 12:50:53.454 CEST [14920] LOG: starting PostgreSQL 16.2, compiled by Visual C++ build 1937, 64-bit
2024-05-19 12:50:53.459 CEST [14920] LOG: listening on IPv6 address "::1", port 5433
2024-05-19 12:50:53.459 CEST [14920] LOG: listening on IPv4 address "127.0.0.1", port 5433
2024-05-19 12:50:53.515 CEST [5004] LOG: database system was interrupted while in recovery at log time 2024-05-19 12:21:22 CEST
2024-05-19 12:50:53.515 CEST [5004] HINT: If this has occurred more than once some data might be corrupted and you might need to choose an earlier recovery target.
...The system cannot find the file specified.
2024-05-19 12:50:57.657 CEST [5004] LOG: starting archive recovery
The system cannot find the file specified.
The system cannot find the file specified.
2024-05-19 12:50:57.725 CEST [5004] LOG: consistent recovery state reached at 0/80000530
2024-05-19 12:50:57.725 CEST [14920] LOG: database system is ready to accept read-only connections
2024-05-19 12:50:57.725 CEST [5004] LOG: invalid record length at 0/80000530: expected at least 24, got 0
2024-05-19 12:50:57.725 CEST [5004] LOG: redo is not required
The system cannot find the file specified.
done
server started

C:\Program Files\PostgreSQL\16\bin>The system cannot find the file specified.
2024-05-19 12:50:57.794 CEST [5004] LOG: selected new timeline ID: 3
The system cannot find the file specified.
2024-05-19 12:50:57.913 CEST [5004] LOG: archive recovery complete
2024-05-19 12:50:57.920 CEST [3456] LOG: checkpoint starting: end-of-recovery immediate wait
2024-05-19 12:50:57.937 CEST [3456] LOG: checkpoint complete: wrote 2 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.002 s, sync=0.003 s, total=0.021 s; sync files=2, longest=0.002 s, average=0.002 s; distance=0 kb, estimate=0 kb; lsn=0/80000530, redo lsn=0/80000530
2024-05-19 12:50:57.967 CEST [14920] LOG: database system is ready to accept connections
```

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.

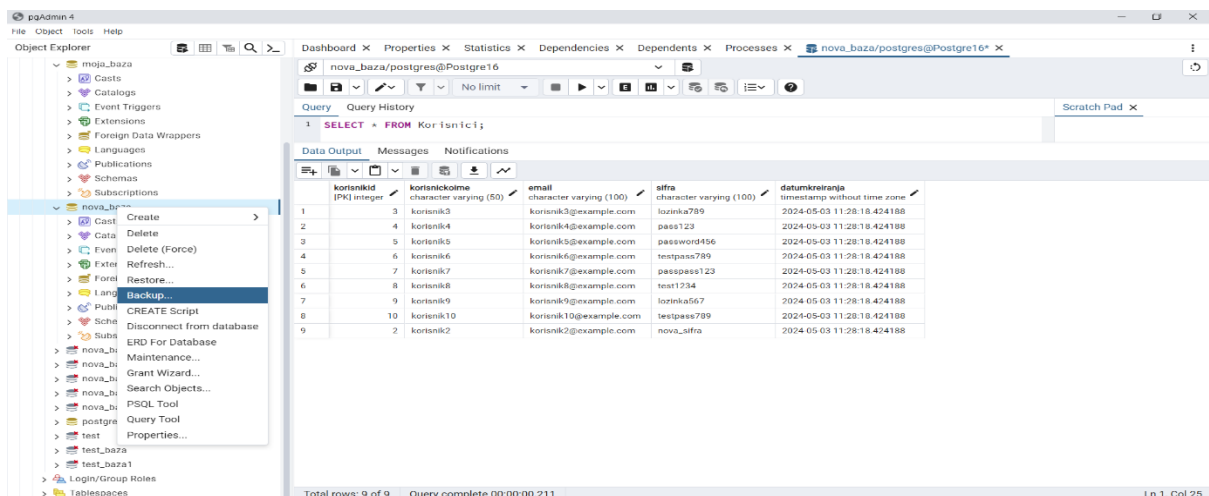
```
postgres=# select * from departmani;
 departman_id |      naziv
-----+-----
          1 | HR
          2 | Marketing
          3 | Finansije
          4 | Informacione tehnologije
          5 | Pravna sluzba
          6 | Operacije
          7 | Tehnicka sluzba
(7 rows)
```

Slika 52-Uspešan oporavak tabele DEPARTMANI

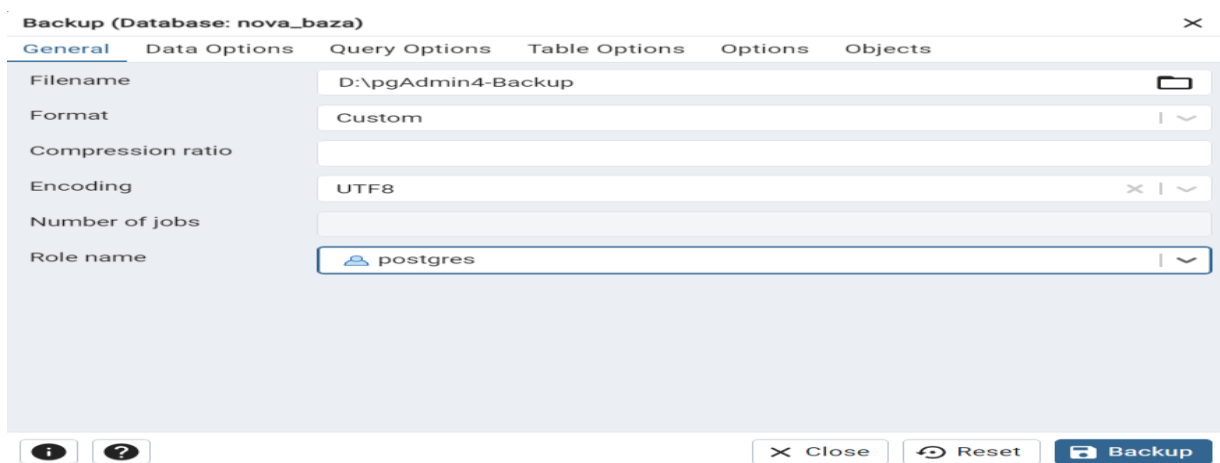
## 4. KREIRANJE BACKUP-A I RESTORE-A KORIŠĆENJEM pgAdmin 4 ALATA

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 naposljetku 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.

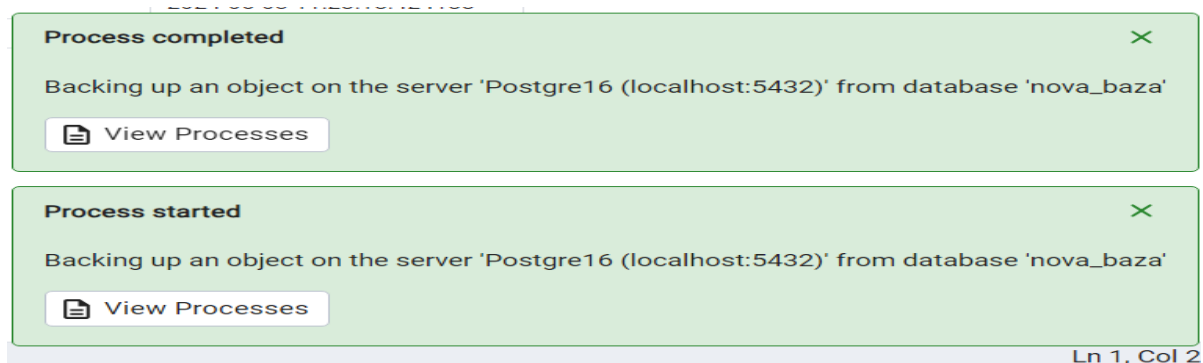


Slika 53–Prikaz početnog koraka za kreiranje backup verzije podataka korišćenjem pgAdmin 4 alata

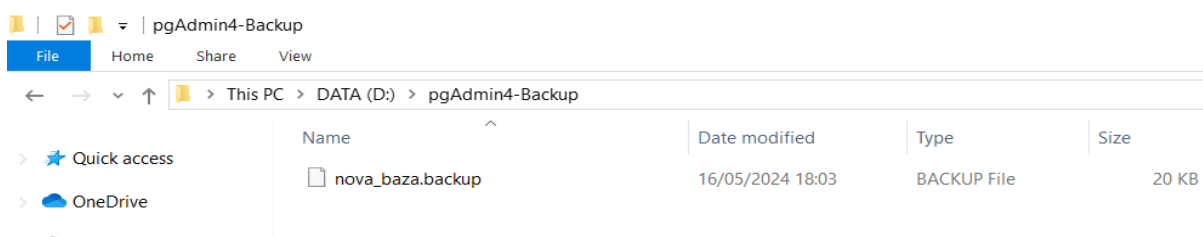


Slika 54 –Prikaz forme za unos informacija o backup verziji prilikom njenog kreiranja

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

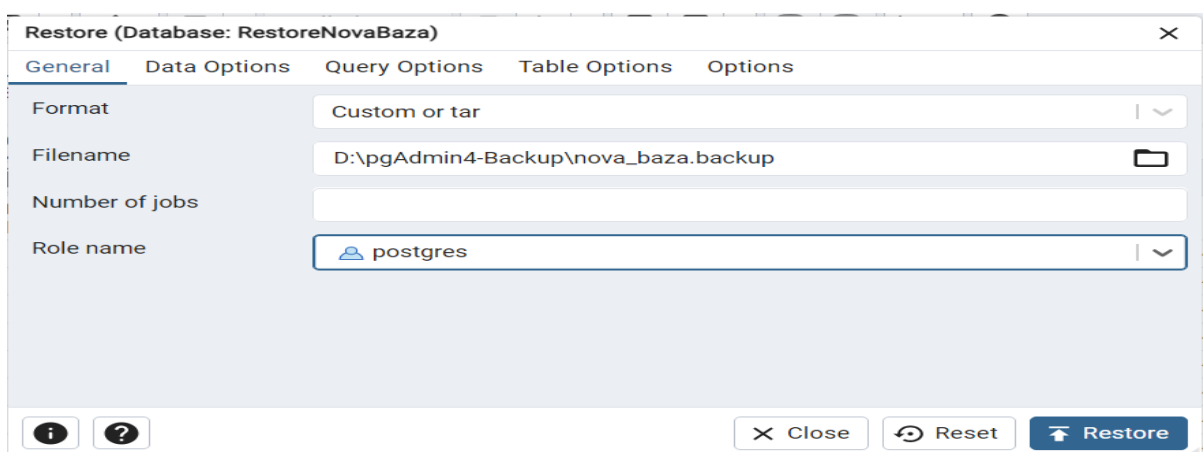


Slika 55-Prikaz poruka koje obavještavaju administratora o ishodu procesa



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 ekstrahovani 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 sigurnosnih 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 sigurnosnih 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=IwZXh0bgNhZW0CMTAAR0nzdBBSZVdHXK2LJLrRD1oO4LvDu01g4hT7FGTYa9DSzNkYsAfIoFYkA4Y\\_aem\\_AQiq3dwl2J7lgMUjA5\\_VWHnEoRZgyxmUP6X2cOzKZ1D0YxCYg5MdPHRBAV2hS6EF1XuWRfs2ygEpWNkL9aTcWY9w](https://www.g2.com/articles/database-backup?fbclid=IwZXh0bgNhZW0CMTAAR0nzdBBSZVdHXK2LJLrRD1oO4LvDu01g4hT7FGTYa9DSzNkYsAfIoFYkA4Y_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=IwZXh0bgNhZW0CMTAAR0UEHNE2Pr54fBh6mY7S-UsEx8pQjBdCRWf-ETlivjwga5yt3Vtfr9Z580\\_aem\\_AXXTS3GrS1kWalf0IavDZy2aBUUicrcI3A45sBeGN8iZdqUiLcGB7hx0ZXIatuHlrGb2GVfoEbFsyeo3OPAUnZK2](https://sqlbackupandftp.com/blog/how-to-backup-and-restore-postgresql-database/?fbclid=IwZXh0bgNhZW0CMTAAR0UEHNE2Pr54fBh6mY7S-UsEx8pQjBdCRWf-ETlivjwga5yt3Vtfr9Z580_aem_AXXTS3GrS1kWalf0IavDZy2aBUUicrcI3A45sBeGN8iZdqUiLcGB7hx0ZXIatuHlrGb2GVfoEbFsyeo3OPAUnZK2)

[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/>