

Archiwizacja ciągła i przywracanie do punktu tuż przed awarią

W PostgreSQL działa mechanizm podobny do redo i archivelogów w Oracle. W PostgreSQL mowa o mechanizmie WAL – Write Ahead Log. Działa to tak :

1. Użytkownik zmienia jakieś dane
2. Informacja o zmianie trafia do plików WAL – w których przechowywane są informacje o zmianach na wypadek gdyby zmiany nie zostały utrwalone w plikach danych a nastąpiłoby np. odcięcie zasilania.
3. Zmiana danych następuje na poziomie bloków w buforze – od tej pory noszą miano brudnych bloków.
4. Kiedy następuje checkpoint brudne bloki są utrwalane na poziomie plików danych. Checkpoint następuje wtedy gdy upłynie czas określony w parametrze checkpoint_timeout, lub gdy skończy się miejsce w plikach WAL (domyślnie są to trzy pliki o rozmiarze 16MB każdy. Ich ilość określamy w parametrze checkpoint_segments).
5. Jeśli archiwizacja ciągła jest wyłączona (a jest wyłączona domyślnie) bieżące pliki WAL są kasowane z katalogu pg_xlog, a w ich miejsce powstają nowe o kolejnych numerach. Jeśli w wyniku np. długo wykonującego się backupu w katalogu pg_xlog powstanie więcej niż 3 pliki WAL taka ich ilość pozostanie i będzie rotowała, ale nadal w paczkach po tyle plików ile określimy w parametrze checkpoint_segments. Jeśli archiwizacja ciągła jest włączona to przed skasowaniem pliki WAL kopiowane są w miejsce archiwizacji które wskażesz w konfiguracji.

W punktach powyżej pojawia się pojęcie archiwizacji ciągłej. Co to takiego? Wyobraź sobie że zarządzasz dużym systemem którego baza danych oparta jest o PostgreSQL. Robisz backupy raz w tygodniu korzystając z pg_dump lub pg_dumpall. Czy to nam wystarczy? Nie zawsze. Jeśli zrobiłeś backup np. w niedzielę a w środę nastąpi awaria to z takiego rzutu odzyskasz stan bazy z niedzieli. Czy to się sprawdzi np. w banku? Dla klienta banku pewnie byłoby to korzystne, o ile backup był zrobiony tuż po wypłatach ;) Dla administratora i samego banku nieco mniej. W takiej sytuacji chcielibyśmy odzyskać stan bazy z momentu tuż przed awarią, a nie sprzed paru dni. Może też nastąpić taka sytuacja że w wyniku ludzkiego błędu będzie trzeba cofnąć stan bazy do wczoraj (a backup jest sprzed przykładowo 5 dni). W obu przypadkach mechanizm archiwizacji ciągłej będzie dla nas rozwiązaniem. Ok, ale jak to działa? Wyobraź sobie że masz kopię zapasową plików danych sprzed tygodnia, ale w ciągu tego tygodnia zbierałeś wszystkie pliki WAL jakie powstawały. Masz więc punkt wyjścia w postaci kopii plików danych i pliki WAL zawierające wszystkie instrukcje zmieniające cokolwiek w bazie od backupu do teraz. Mając taki zestaw możesz przywrócić pliki danych z kopii na pierwotne miejsce, a następnie powtórzyć wszystkie operacje z WAL lub ich część – do wskazanego punktu w czasie. Fajne? Pewnie, bo może nam uratować skórę. Jeśli nie włączysz archiwizacji ciągłej, przywracanie bazy w taki sposób nie będzie możliwe, ponieważ miałbyś backup sprzed jakiegoś czasu – np. kilku dni i pliki WAL zawierające informacje o zmianach z np. ostatniej godziny. Mielibyśmy kilka dni luki. Potrzebujemy całej historii operacji od backupu do teraz, by przywracać bazę do punktu w czasie. Aby włączyć archiwizację ciągłą i umożliwić sobie odtwarzanie baz do punktu w czasie musimy zmienić 3 parametry w pliku postgresql.conf

Edytujemy plik postgresql.conf:

```
bash-4.1$ whoami
postgres
bash-4.1$ nano /var/lib/pgsql/9.4/data/postgresql.conf
```

Odremowujemy parametr WAL_LEVEL i ustawiamy jego wartość na archive lub hot_standby. Jego domyślna wartość to minimal, która pozwala jedynie na odtworzenie bazy po awarii. Wartość archive powoduje dodawanie do WAL informacji które umożliwiają nam wykorzystanie archiwizacji ciągłej, a hot_standby pozwala na uruchamianie serwera w trybie hot_standby i puszczanie zapytań działających w trybie tylko do odczytu.

```
# -----
# WRITE AHEAD LOG
# -----
#
# - Settings -
wal_level = archive # minimal, archive, hot standby, or logical
# (change requires restart)
```

Odremowujemy teraz parametr ARCHIVE_MODE i ustawiamy jego wartość na ON. Umożliwia on uruchomienie archiwizacji.

```
# - Archiving -
archive_mode = on # allows archiving to be done
# (change requires restart)
```

Zarchiwizowane pliki WAL muszą gdzieś lądować, więc przygotowujemy sobie odpowiedni katalog. Ważne jest by jego właścicielem był użytkownik systemowy postgres. Zadbaj o to by katalog leżał na jakimś dysku ze sporą ilością miejsca, ponieważ zarchiwizowane pliki WAL sporo go zajmą.

Przygotowanie katalogu do którego będą wrzucane zarchiwizowane pliki WAL:

```
[root@vps-1077604-8206 home]# mkdir /home/pg_wal_archives
[root@vps-1077604-8206 home]# chown postgres /home/pg_wal_archives/
[root@vps-1077604-8206 home]#
```

Została jeszcze jedna rzecz. Pliki WAL same się nie skopiują, więc musimy podać komendę która to zrobi. Można by sobie zadać pytanie – a dlaczego nie wystarczy wskazać ścieżki do katalogu? Komenda którą podajemy to nie musi być polecenie cp. Dzięki takiej konstrukcji możemy tu wykorzystać rsync, ftp, albo jakikolwiek inny mechanizm.

Odszukujemy parametr ARCHIVE_COMMAND i podajemy komendę kopiującą oraz odremowujemy go:

```
# - Archiving -
# Archiving
archive_mode = on                # allows archiving to be done
                                # (change requires restart)
archive_command = 'cp %p /home/pg wal archives/%f' # command to use
                                # placeholders: %p = path of file to archive
```

Ponieważ plik postgresql.conf nie jest czytany „na żywo”, musimy przeładować konfigurację. Przeładowanie konfiguracji:

service postgresql-9.4 restart

```
[root@vps-1077604-8206 ~]# service postgresql-9.4 restart
Stopping postgresql-9.4 service: [ OK ]
Starting postgresql-9.4 service: [ OK ]
[root@vps-1077604-8206 ~]#
```

Zadaliśmy już o kopiowanie plików WAL, teraz potrzebujemy naszego punktu wyjścia jakim będzie kopia zapasowa. Nie może to być kopia stworzona z użyciem pg_dump ani pg_dumpall ponieważ są to kopie logiczne a nie fizyczne. Aby wykonać kopię zapasową online, musimy przejść do trybu backupu. Sprawia on że zmiany w bazie nie będą utrwalane w plikach danych, a jedynie w plikach WAL. Zostaną one utrwalone dopiero po zakończeniu trybu backupu. Chodzi o spójność danych, nie możemy doprowadzić do sytuacji że coś się pozmienia w tych plikach w trakcie kopiowania.

Tryb backupu:

psql -c „select pg_start_backup('etykieta_backupu)'”

```
bash-4.1$ psql -c "select pg_start_backup('backup_mapeta')"
could not change directory to "/root": Brak dostępu
pg_start_backup
-----
0/2C000028
(1 row)
bash-4.1$
```

Utworzymy sobie też katalog pod backup:

```
[root@vps-1077604-8206 backups]# pwd
/var/lib/pgsql/9.4/backups
[root@vps-1077604-8206 backups]# mkdir 20160404
[root@vps-1077604-8206 backups]# chown postgres 20160404
[root@vps-1077604-8206 backups]# ls -la
razem 12
drwx----- 3 postgres postgres 4096 04-04 17:25 .
drwx----- 4 postgres postgres 4096 03-22 11:41 ..
drwxr-xr-x  2 postgres dba      4096 04-04 17:25 20160404
[root@vps-1077604-8206 backups]#
```

W dalszej kolejności kopiujemy pliki. Możemy wykonać to dowolną komendą systemu operacyjnego.

```
bash-4.1$ tar cfp /var/lib/pgsql/9.4/backups/20160404/backup.tar /var/lib/pgsql/9.4/data
bash-4.1$ ls /var/lib/pgsql/9.4/backups/20160404/
backup.tar
```

Po zakończeniu kopiowania wychodzimy z trybu backupu:

psql -c „select pg_stop_backup()”

```
bash-4.1$ psql -c "select pg_stop_backup()"
could not change directory to "/root": Brak dostępu
UWAGA: pg_stop_backup kompletny, zarchiwizowano wszystkie wymagane segmenty WAL
pg_stop_backup
-----
 0/2C0000B8
(1 row)
```

Uruchomiłem tryb backupu i stworzyłem tabelkę do której wrzucam duży wolumen danych. Chodzi mi o to by troszkę plików WAL zostało dodanych do naszego nowego katalogu. Stworzyłem pustą tabelkę i uruchomiłem prosty skrypt w języku Plpg/SQL który do tej nowej tabelki dodaje 10 milionów wierszy.

```
create table pobackupie(
id integer primary key,
cos varchar
);

do
$$
begin
for x in 1..10000000 loop
insert into pobackupie values (x,'x po raz '||x);
end loop;
end
$$;

commit;
```

Zawartość katalogu pg_xlog i nowego katalogu na zarchwizowane pliki WAL w trakcie wykonywania powyższego skryptu:

```
bash-4.1$ ls /var/lib/pgsql/9.4/data/pg_xlog/
00000001000000000000002C 00000001000000000000002D 00000001000000000000002F 000000010000000000000031 archive_status
00000001000000000000002C.00000028.backup 00000001000000000000002E 000000010000000000000030 000000010000000000000032
bash-4.1$ ls /home/pg_wal_archives/
00000001000000000000002B 00000001000000000000002C 00000001000000000000002C.00000028.backup
bash-4.1$
```

Właściwości	Statystyki	Powiązania	Zależne
Właściwości			
Zadne właściwości nie są dostępne dla bieżącej selekcji			

Zwróć uwagę że tabelkę stworzyłem po wykonaniu backupu, a więc w kopii zapasowej nie ma o niej informacji! To ważne bo za chwilę będziemy odtwarzać bazę z użyciem wszystkich zarchiwizowanych plików WAL.

Po zakończeniu mielenia skryptu nasza tabelka zawiera sporo danych:

The screenshot shows a SQL editor window titled "Edytor SQL" and "Graficzny Konstruktor Zapytania". The main text area contains the following SQL script:

```
create table pobackupie(  
  id integer primary key,  
  cos varchar  
);  
  
do  
  $$  
begin  
  for x in 1..10000000 loop  
    insert into pobackupie values (x,'x po raz '||x);  
  end loop;  
end  
$$;  
  
commit;  
  
select * from pobackupie limit 100;
```

Below the editor is a "Okno wyjściowe" (Output Window) with tabs for "Dane wyjściowe", "Plan zapytania", "Komunikaty", and "Historia". The "Dane wyjściowe" tab is active, displaying a table with the following data:

	id integer	cos character varying
1	208511	x po raz 208511
2	208512	x po raz 208512
3	208513	x po raz 208513

Mój katalog ze zarchiwizowanymi plikami WAL też trochę się wypełnił. Powstało w nim 89 plików, każdy po 16 MB - razem 1424 MB, a więc prawie 1,5 GB!

```
bash-4.1$ ls /var/lib/pgsql/9.4/data/pg_xlog/
00000001000000000000002C.00000028.backup 000000010000000000000086 000000010000000000000089 archive_status
000000010000000000000084 000000010000000000000087 00000001000000000000008A
000000010000000000000085 000000010000000000000088 00000001000000000000008B
bash-4.1$ ls /home/pg_wal_archives/
00000001000000000000002B 00000001000000000000004A 00000001000000000000006A
00000001000000000000002C 00000001000000000000004B 00000001000000000000006B
00000001000000000000002C.00000028.backup 00000001000000000000004C 00000001000000000000006C
00000001000000000000002D 00000001000000000000004D 00000001000000000000006D
00000001000000000000002E 00000001000000000000004E 00000001000000000000006E
00000001000000000000002F 00000001000000000000004F 00000001000000000000006F
000000010000000000000030 000000010000000000000050 000000010000000000000070
000000010000000000000031 000000010000000000000051 000000010000000000000071
000000010000000000000032 000000010000000000000052 000000010000000000000072
000000010000000000000033 000000010000000000000053 000000010000000000000073
000000010000000000000034 000000010000000000000054 000000010000000000000074
000000010000000000000035 000000010000000000000055 000000010000000000000075
000000010000000000000036 000000010000000000000056 000000010000000000000076
000000010000000000000037 000000010000000000000057 000000010000000000000077
000000010000000000000038 000000010000000000000058 000000010000000000000078
000000010000000000000039 000000010000000000000059 000000010000000000000079
00000001000000000000003A 00000001000000000000005A 00000001000000000000007A
00000001000000000000003B 00000001000000000000005B 00000001000000000000007B
00000001000000000000003C 00000001000000000000005C 00000001000000000000007C
00000001000000000000003D 00000001000000000000005D 00000001000000000000007D
00000001000000000000003E 00000001000000000000005E 00000001000000000000007E
00000001000000000000003F 00000001000000000000005F 00000001000000000000007F
000000010000000000000040 000000010000000000000060 000000010000000000000080
000000010000000000000041 000000010000000000000061 000000010000000000000081
000000010000000000000042 000000010000000000000062 000000010000000000000082
000000010000000000000043 000000010000000000000063 000000010000000000000083
000000010000000000000044 000000010000000000000064 000000010000000000000084
000000010000000000000045 000000010000000000000065 000000010000000000000085
000000010000000000000046 000000010000000000000066 000000010000000000000086
000000010000000000000047 000000010000000000000067 000000010000000000000087
000000010000000000000048 000000010000000000000068 000000010000000000000088
000000010000000000000049 000000010000000000000069
bash-4.1$
```

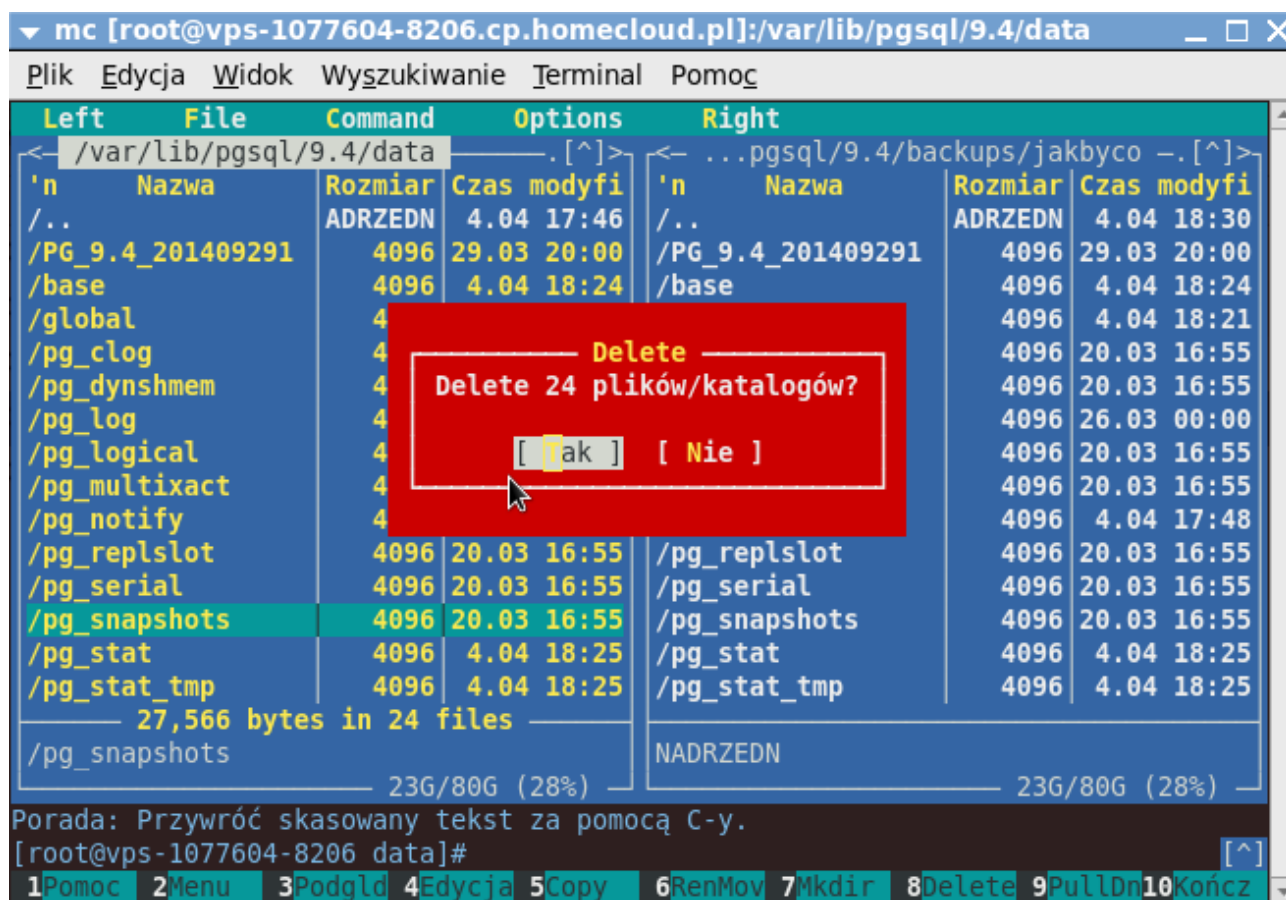
Zasymulujemy teraz awarię która będzie wymagała odtwarzania bazy do punktu tuż przed awarią. Zanim zaczniemy odtwarzać, warto zrobić kopię aktualnego stanu plików danych, nawet jeśli są one uszkodzone. Może się tak zdarzyć że w trakcie odtwarzania coś pójdzie nie tak i jeszcze bardziej napsujemy.

```

mc [root@vps-1077604-8206.cp.homecloud.pl]:/var/lib/pgsql/9.4/data
Plik  Edycja  Widok  Wyszukiwanie  Terminal  Pomoc
Left  File      Command  Options      Right
<- /var/lib/pgsql/9.4/data .[^]> <- ...pgsql/9.4/backups/jakbyco -.[^]>
'n   Nazwa      Rozmiar  Czas modyfi  'n   Nazwa      Rozmiar  Czas modyfi
/pg_replslot      4096    20.03 16:55  /..   ADRZEDN    4.04 18:30
/pg_serial         4096    20.03 16:55  /PG_9.4_201409291 4096 29.03 20:00
/pg_snapshots     4096    20.03 16:55  /base  4096  4.04 18:24
/pg_stat          4096    4.04 18:25  /global 4096  4.04 18:21
/pg_stat_tmp      4096    4.04 18:25  /pg_clog 4096 20.03 16:55
/pg_subtrans      4096    20.03 16:55  /pg_dynshmem 4096 20.03 16:55
/pg_tblspc        4096    29.03 20:00  /pg_log  4096 26.03 00:00
/pg_twophase      4096    20.03 16:55  /pg_logical 4096 20.03 16:55
/pg_xlog          4096    4.04 18:08  /pg_multixact 4096 20.03 16:55
PG_VERSION        4       20.03 16:55  /pg_notify 4096  4.04 17:48
pg_hba.conf       4489    22.03 11:39  /pg_replslot 4096 20.03 16:55
pg_ident.conf     1636    20.03 16:55  /pg_serial  4096 20.03 16:55
postgres-uto.conf  88      20.03 16:55  /pg_snapshots 4096 20.03 16:55
postgresql.conf   21290   4.04 17:22  /pg_stat    4096  4.04 18:25
postmaster.opts   59      4.04 17:48  /pg_stat_tmp 4096  4.04 18:25
postmaster.opts
23G/80G (29%) NADRZEDN 23G/80G (29%)
Porada: Uzupełnianie: M-Tab (lub Esc+Tab). Podwójne wciśnięcie wywołuje listę.
[root@vps-1077604-8206 data]#
1Pomoc 2Menu 3Podgląd 4Edycja 5Copy 6RenMov 7Mkdir 8Delete 9PullDn10Kończ

```


Przed odtwarzaniem usuwamy wszystkie pliki danych z katalogu \$PGDATA, będą one zastępowane tymi z kopii zapasowej.



```

mc [root@vps-1077604-8206.cp.homecloud.pl]:/var/lib/pgsql/9.4/data
Plik Edycja Widok Wyszukiwanie Terminal Pomoc

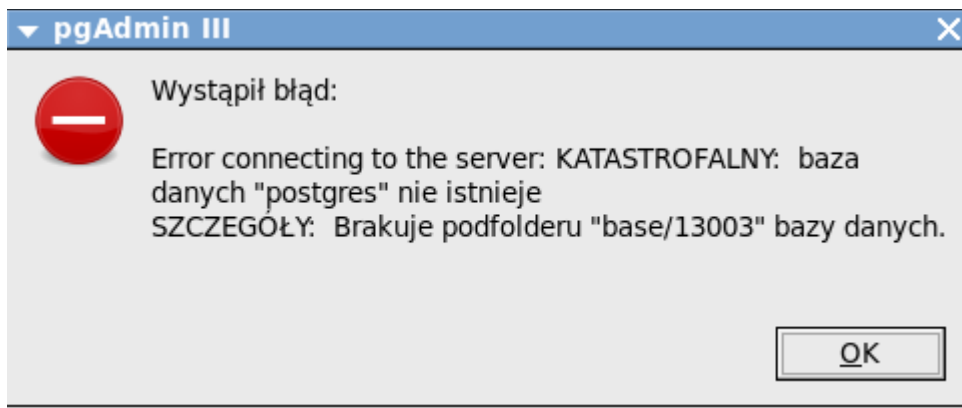
Left File Command Options Right
<- /var/lib/pgsql/9.4/data .[^]> <- ...pgsql/9.4/backups/jakbyco -.[^]>
'n Nazwa Rozmiar Czas modyfi 'n Nazwa Rozmiar Czas modyfi
/.. ADRZEDN 4.04 17:46 /.. ADRZEDN 4.04 18:30
/Pg_9.4_201409291 4096 29.03 20:00
/base 4096 4.04 18:24
/global 4096 4.04 18:21
/pg_clog 4096 20.03 16:55
/pg_dynshmem 4096 20.03 16:55
/pg_log 4096 26.03 00:00
/pg_logical 4096 20.03 16:55
/pg_multixact 4096 20.03 16:55
/pg_notify 4096 4.04 17:48
/pg_replslot 4096 20.03 16:55
/pg_serial 4096 20.03 16:55
/pg_snapshots 4096 20.03 16:55
/pg_stat 4096 4.04 18:25
/pg_stat_tmp 4096 4.04 18:25

NADRZEDN NADRZEDN
23G/80G (29%) 23G/80G (29%)

Porada: F13 (lub Shift-F3) wywołuje przeglądarkę w trybie surowym.
[root@vps-1077604-8206 data]#
1Pomoc 2Menu 3Podgląd 4Edycja 5Copy 6RenMov 7Mkdir 8Delete 9PullDn 10Kończ

```

Przypominam że klaster jest cały czas uruchomiony. Próba dostępu do jakiegokolwiek bazy kończy się w tej chwili tak:

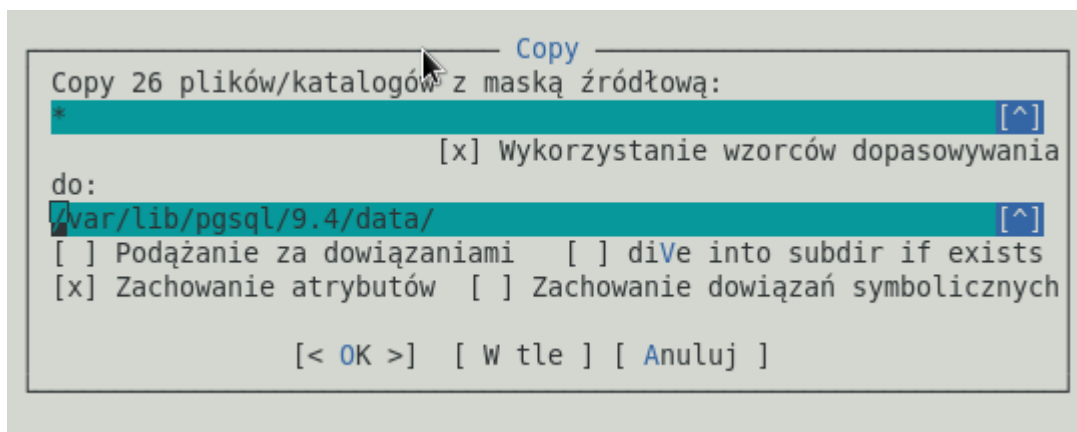


Zatrzymujemy serwer:

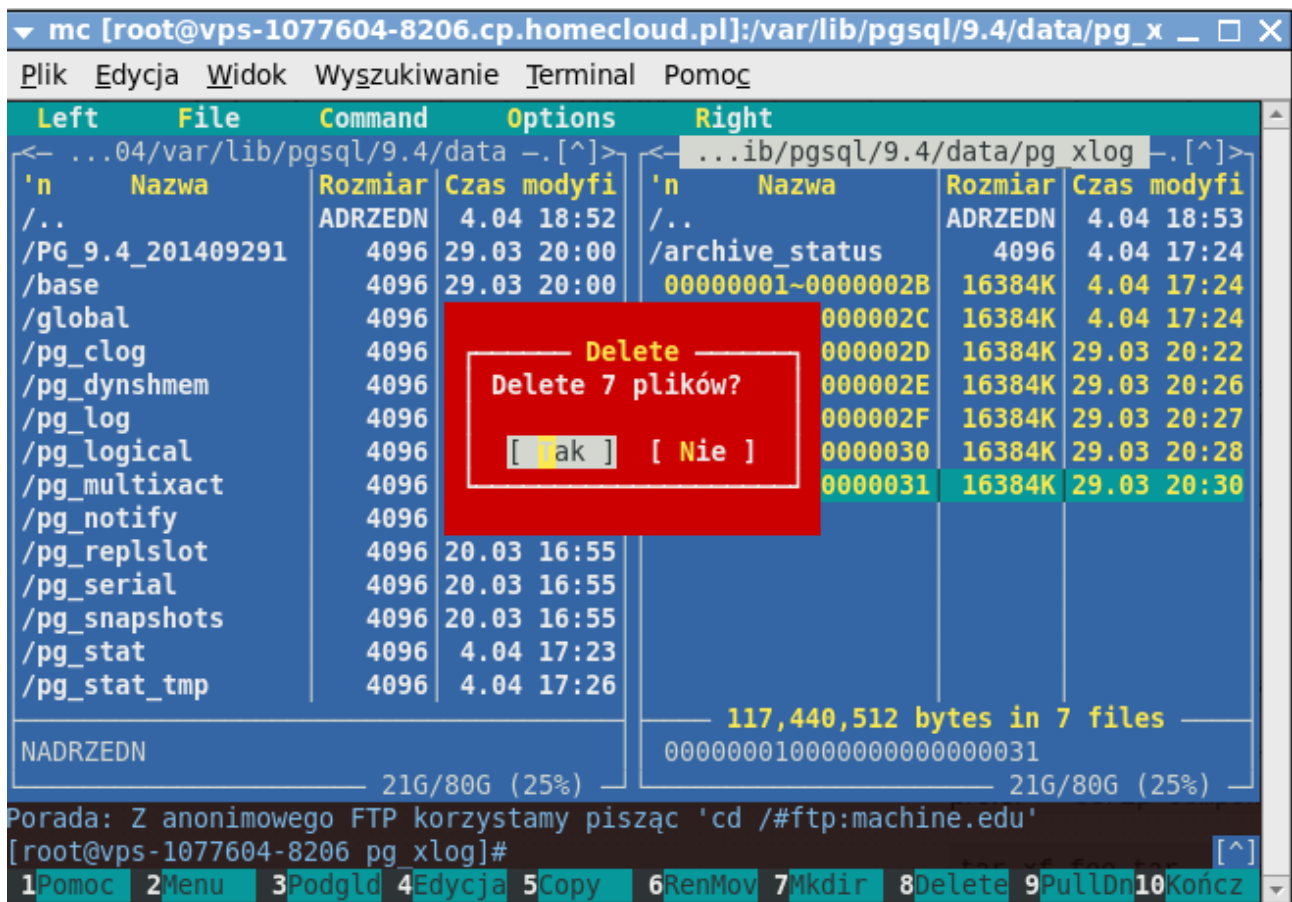
service postgresql-9.4 stop

```
[root@vps-1077604-8206 ~]# whoami
root
[root@vps-1077604-8206 ~]# service postgresql-9.4 stop
Stopping postgresql-9.4 service: [ OK ]
[root@vps-1077604-8206 ~]#
```

Kopiuujemy pliki z backupu :

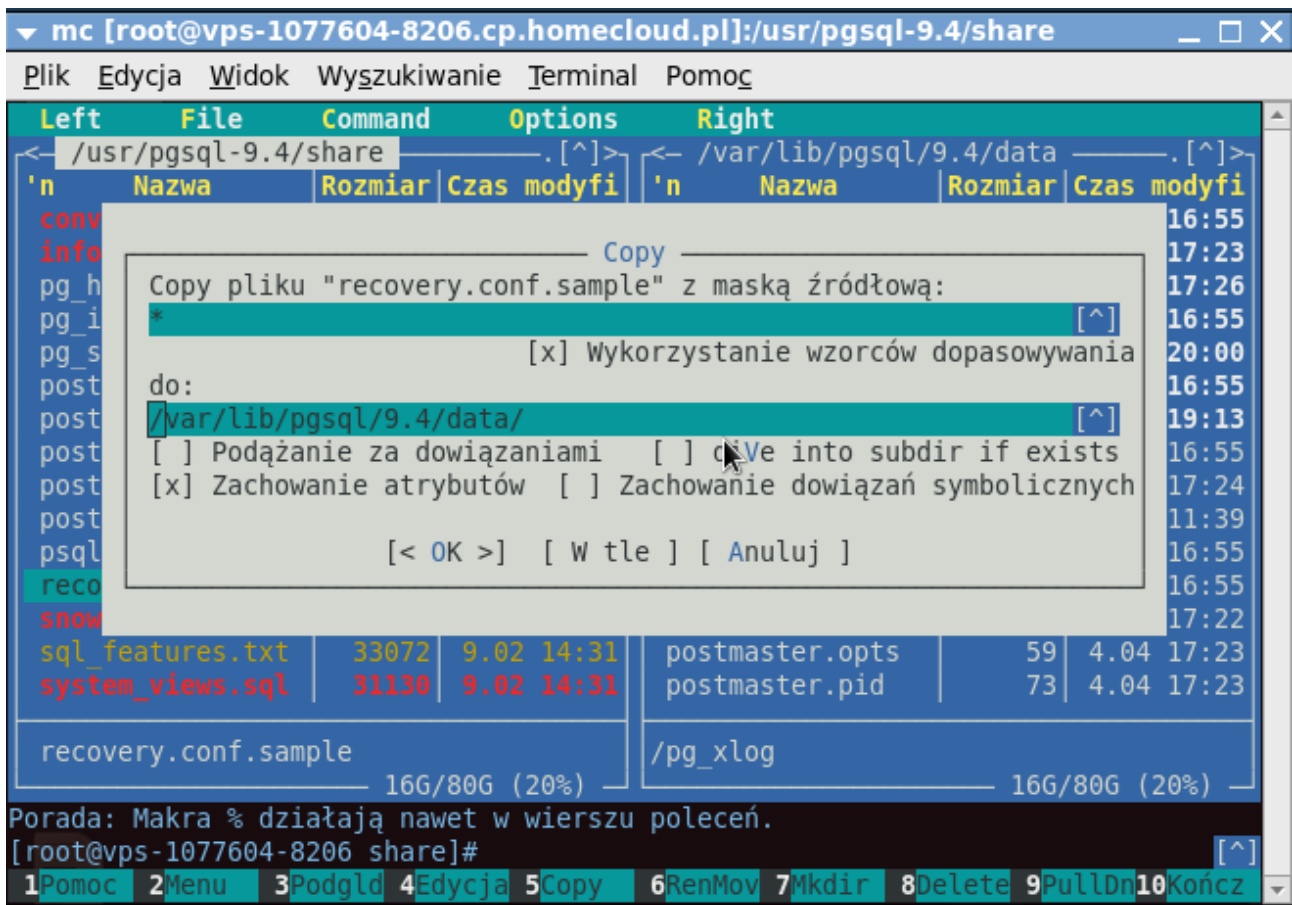


Wywalamy zawartość pg_xlog bo i tak nie koreluje z aktualnym stanem:



Do odzyskiwania będzie nam potrzebny plik recovery.conf. Jego obecność w katalogu \$PGDATA spowoduje że klaster przy uruchamianiu przejdzie w tryb odtwarzania. Poza odtwarzaniem nie powinno go tam oczywiście być. W nim zapisane jest skąd ma zaczytać pliki WAL i jak ma je odtwarzać. Skąd go jednak wziąć? W katalogu /usr/pgsql-9.4/share mamy wiele przykładowych plików konfiguracyjnych, w tym właśnie recovery.conf. Jego nazwa będzie jednak zawierała końcówkę „.sample” którą będziemy musieli usunąć.

Kopiujemy przykładowy plik recovery.conf.sample do katalogu \$PGDATA i zmieniamy jego nazwę na recovery.conf.



Kopiujemy zawartość pliku recovery.conf.sample do pliku recovery.conf:

```
cat recovery.conf.sample > recovery.conf
```

```
bash-4.1$ cat recovery.conf.sample > recovery.conf
bash-4.1$ ls -la
razem 168
drwx----- 20 postgres postgres 4096 04-04 19:18 .
drwx----- 4 postgres postgres 4096 04-04 17:46 ..
-rw----- 1 postgres postgres 204 04-04 17:24 backup_label
drwx----- 8 postgres postgres 4096 03-29 20:00 base
drwx----- 2 postgres postgres 4096 04-04 17:24 global
drwx----- 3 postgres postgres 4096 03-29 20:00 PG_9.4_201409291
drwx----- 2 postgres postgres 4096 03-20 16:55 pg_clog
drwx----- 2 postgres postgres 4096 03-20 16:55 pg_dynshmem
-rw----- 1 postgres postgres 4489 03-22 11:39 pg_hba.conf
-rw----- 1 postgres postgres 1636 03-20 16:55 pg_ident.conf
drwx----- 2 postgres postgres 4096 03-26 00:00 pg_log
drwx----- 4 postgres postgres 4096 03-20 16:55 pg_logical
drwx----- 4 postgres postgres 4096 03-20 16:55 pg_multixact
drwx----- 2 postgres postgres 4096 04-04 17:23 pg_notify
drwx----- 2 postgres postgres 4096 03-20 16:55 pg_replslot
drwx----- 2 postgres postgres 4096 03-20 16:55 pg_serial
drwx----- 2 postgres postgres 4096 03-20 16:55 pg_snapshots
drwx----- 2 postgres postgres 4096 04-04 17:23 pg_stat
drwx----- 2 postgres postgres 4096 04-04 17:26 pg_stat_tmp
drwx----- 2 postgres postgres 4096 03-20 16:55 pg_subtrans
drwx----- 2 postgres postgres 4096 03-29 20:00 pg_tblspc
drwx----- 2 postgres postgres 4096 03-20 16:55 pg_twophase
-rw----- 1 postgres postgres 4 03-20 16:55 PG_VERSION
drwx----- 2 postgres postgres 20480 04-04 19:13 pg_xlog
-rw----- 1 postgres postgres 88 03-20 16:55 postgresql.auto.conf
-rw----- 1 postgres postgres 21290 04-04 17:22 postgresql.conf
-rw----- 1 postgres postgres 59 04-04 17:23 postmaster.opts
-rw----- 1 postgres postgres 73 04-04 17:23 postmaster.pid
-rw-r--r-- 1 postgres postgres 5587 04-04 19:18 recovery.conf
-rw-r--r-- 1 root root 5587 02-09 14:31 recovery.conf.sample
bash-4.1$
```

Edytujemy ten plik:

```
-rw-r--r-- 1 postgres postgres 5587 04-04 19:18 recovery.conf
-rw-r--r-- 1 root root 5587 02-09 14:31 recovery.conf.sample
bash-4.1$ nano recovery.conf
```


odnajdujemy w nim linijkę z restore_command, odkomentujemy i piszemy tak:

```
'cp /home/pg_wal_archives/%f %p'
```

```
#
# NOTE that the basename of %p will be different from %f; do not
# expect them to be interchangeable.
#
restore_command = 'cp /home/pg_wal_archives/%f %p' # e.g. 'cp /home/pg_wal_archives/2016-04-04-17:23:00.106 %p'
```

Konstrukcja tej instrukcji jest ładząco podobna do instrukcji archive_command – owszem i działa w drugą stronę :)

Z użyciem programu tail włączamy sobie podgląd logów :

```
[root@vps-1077604-8206 data]# tail -f /var/lib/pgsql/9.4/data/pg_log/postgresql-Mon.log
< 2016-04-04 02:33:31.780 CEST >DZIENNIK: niekompletny pakiet uruchomieniowy
< 2016-04-04 17:23:00.106 CEST >DZIENNIK: odebrano żądanie szybkiego zamknięcia
< 2016-04-04 17:23:00.136 CEST >DZIENNIK: przerywanie wszelkich aktywnych transakcji
< 2016-04-04 17:23:00.142 CEST >DZIENNIK: zamknięto program wywołujący autoodkurzanie
< 2016-04-04 17:23:00.226 CEST >DZIENNIK: zamykanie log
< 2016-04-04 17:23:00.538 CEST >DZIENNIK: system bazy danych jest zamknięty
< 2016-04-04 17:23:01.412 CEST >DZIENNIK: system bazy danych został zamknięty 2016-04-04 17:23:00 CEST
< 2016-04-04 17:23:01.510 CEST >DZIENNIK: MultiXact member wraparound protections are now enabled
< 2016-04-04 17:23:01.513 CEST >DZIENNIK: uruchomiono program wywołujący autoodkurzanie
< 2016-04-04 17:23:01.513 CEST >DZIENNIK: system bazy danych jest gotowy do przyjmowania połączeń
[root@vps-1077604-8206 data]#
```

Uruchamiamy usługę, a PostgreSQL widząc istnienie pliku recovery.conf rozpoczyna odtwarzanie:

```
postgresql-Fri.log postgresql-Mon.log postgresql-Sat.log postgresql-Sun.log postgresql-Thu.log postgresql-Tue.log
[root@vps-1077604-8206 data]# tail -f /var/lib/pgsql/9.4/data/pg_log/postgresql-Mon.log
< 2016-04-04 02:33:31.780 CEST >DZIENNIK: niekompletny pakiet uruchomieniowy
< 2016-04-04 17:23:00.106 CEST >DZIENNIK: odebrano żądanie szybkiego zamknięcia
< 2016-04-04 17:23:00.142 CEST >DZIENNIK: zamknięto program wywołujący autoodkurzanie
< 2016-04-04 17:23:00.226 CEST >DZIENNIK: zamykanie master.pid
< 2016-04-04 17:23:00.538 CEST >DZIENNIK: system bazy danych jest zamknięty
< 2016-04-04 17:23:01.412 CEST >DZIENNIK: system bazy danych został zamknięty 2016-04-04 17:23:00 CEST
< 2016-04-04 17:23:01.510 CEST >DZIENNIK: MultiXact member wraparound protections are now enabled
< 2016-04-04 17:23:01.513 CEST >DZIENNIK: uruchomiono program wywołujący autoodkurzanie
< 2016-04-04 17:23:01.513 CEST >DZIENNIK: system bazy danych jest gotowy do przyjmowania połączeń
< 2016-04-04 19:29:43.784 CEST >DZIENNIK: działanie systemu bazy danych zostało przerwane; ostatnie znane podniesienie
< 2016-04-04 19:29:43.784 CEST >DZIENNIK: utworzenie brakującego folderu WAL "pg_xlog/archive_status"
< 2016-04-04 19:29:43.880 CEST >DZIENNIK: rozpoczęto odzyskiwanie archiwum
< 2016-04-04 19:29:43.899 CEST >DZIENNIK: odtworzono plik dziennika "00000010000000000000002C" z archiwum
< 2016-04-04 19:29:43.917 CEST >DZIENNIK: ponowienie uruchamia się w 0/2C000090
< 2016-04-04 19:29:43.918 CEST >DZIENNIK: stan spójnego odzyskania osiągnięty w 0/2C0000B8
< 2016-04-04 19:29:43.936 CEST >DZIENNIK: odtworzono plik dziennika "00000010000000000000002D" z archiwum
< 2016-04-04 19:29:44.190 CEST >DZIENNIK: odtworzono plik dziennika "00000010000000000000002E" z archiwum
< 2016-04-04 19:29:44.450 CEST >DZIENNIK: odtworzono plik dziennika "00000010000000000000002F" z archiwum
< 2016-04-04 19:29:44.712 CEST >DZIENNIK: odtworzono plik dziennika "000000100000000000000030" z archiwum
< 2016-04-04 19:29:44.996 CEST >DZIENNIK: odtworzono plik dziennika "000000100000000000000031" z archiwum
< 2016-04-04 19:29:45.265 CEST >DZIENNIK: odtworzono plik dziennika "000000100000000000000032" z archiwum
< 2016-04-04 19:29:45.523 CEST >DZIENNIK: odtworzono plik dziennika "000000100000000000000033" z archiwum
< 2016-04-04 19:29:45.779 CEST >DZIENNIK: odtworzono plik dziennika "000000100000000000000034" z archiwum
< 2016-04-04 19:29:46.050 CEST >DZIENNIK: odtworzono plik dziennika "000000100000000000000035" z archiwum
< 2016-04-04 19:29:46.309 CEST >DZIENNIK: odtworzono plik dziennika "000000100000000000000036" z archiwum
< 2016-04-04 19:29:46.583 CEST >DZIENNIK: odtworzono plik dziennika "000000100000000000000037" z archiwum
```

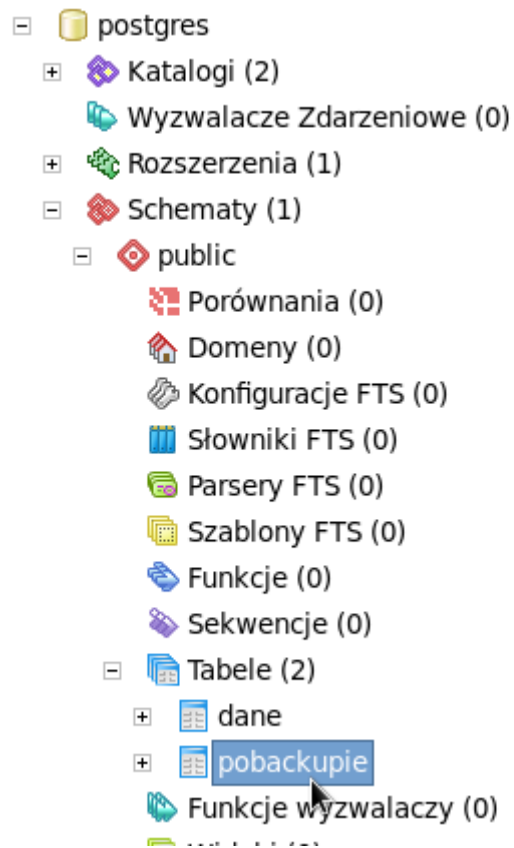
Zwróć szczególną uwagę na to, że mimo że teoretycznie usługa została uruchomiona jako taka, to odtwarzanie cały czas trwa i nadal nie będzie można się podłączyć do bazy.

```
< 20:[root@vps-1077604-8206 data]# service postgresql-9.4 start
< 20:Starting postgresql-9.4 service:
< 20:[root@vps-1077604-8206 data]#
< 2016-04-04 19:30:19.816 CEST >DZIENNIK: odtworzono plik dziennika "0000000100000000" z archiwum
< 2016-04-04 19:30:21.017 CEST >DZIENNIK: odtworzono plik dziennika "0000000100000000A0" z archiwum
< 2016-04-04 19:30:21.773 CEST >DZIENNIK: odtworzono plik dziennika "0000000100000000A1" z archiwum
< 2016-04-04 19:30:23.673 CEST >DZIENNIK: odtworzono plik dziennika "0000000100000000A2" z archiwum
< 2016-04-04 19:30:24.163 CEST >DZIENNIK: odtworzono plik dziennika "0000000100000000A3" z archiwum
< 2016-04-04 19:30:25.571 CEST >DZIENNIK: odtworzono plik dziennika "0000000100000000A4" z archiwum
< 2016-04-04 19:30:26.999 CEST >DZIENNIK: odtworzono plik dziennika "0000000100000000A5" z archiwum
< 2016-04-04 19:30:28.017 CEST >DZIENNIK: odtworzono plik dziennika "0000000100000000A6" z archiwum
< 2016-04-04 19:30:29.035 CEST >DZIENNIK: odtworzono plik dziennika "0000000100000000A7" z archiwum
< 2016-04-04 19:30:30.053 CEST >DZIENNIK: odtworzono plik dziennika "0000000100000000A8" z archiwum
```

Gdy zakończy odtwarzanie:

```
< 2016-04-04 19:36:19.878 CEST >DZIENNIK: odtworzono plik dziennika "000000010000000100000033" z archiwum
< 2016-04-04 19:36:20.080 CEST >DZIENNIK: odtworzono plik dziennika "000000010000000100000034" z archiwum
< 2016-04-04 19:36:20.446 CEST >DZIENNIK: odtworzono plik dziennika "000000010000000100000035" z archiwum
< 2016-04-04 19:36:21.080 CEST >DZIENNIK: odtworzono plik dziennika "000000010000000100000036" z archiwum
< 2016-04-04 19:36:21.762 CEST >DZIENNIK: odtworzono plik dziennika "000000010000000100000037" z archiwum
< 2016-04-04 19:36:22.699 CEST >DZIENNIK: stan spójnego odzyskania osiągnięty w 1/3786B100
cp: nie można wykonać stat na `/home/pg_wal_archives/000000010000000100000038': Nie ma takiego pliku ani katalogu
< 2016-04-04 19:36:22.903 CEST >DZIENNIK: ponowienie wykonane w 1/37FFFE30
< 2016-04-04 19:36:22.903 CEST >DZIENNIK: czas ostatniej zakończonej transakcji według dziennika 2016-04-04 17:50:01.6
< 2016-04-04 19:36:22.927 CEST >DZIENNIK: odtworzono plik dziennika "000000010000000100000037" z archiwum
cp: nie można wykonać stat na `/home/pg_wal_archives/00000002.history': Nie ma takiego pliku ani katalogu
< 2016-04-04 19:36:22.933 CEST >DZIENNIK: wybrany nowy ID linii czasowej: 2
cp: nie można wykonać stat na `/home/pg_wal_archives/00000001.history': Nie ma takiego pliku ani katalogu
< 2016-04-04 19:36:24.404 CEST >DZIENNIK: wykonane odtworzenie archiwum
< 2016-04-04 19:36:41.260 CEST >DZIENNIK: MultiXact member wraparound protections are now enabled
< 2016-04-04 19:36:41.267 CEST >DZIENNIK: system bazy danych jest gotowy do przyjmowania połączeń
< 2016-04-04 19:36:41.267 CEST >DZIENNIK: uruchomiono program wywołujący autoodkurzanie
```

Odtwarzanie zostało zakończone, podłączam się więc do bazy i sprawdzam czy istnieje moja tabelka:



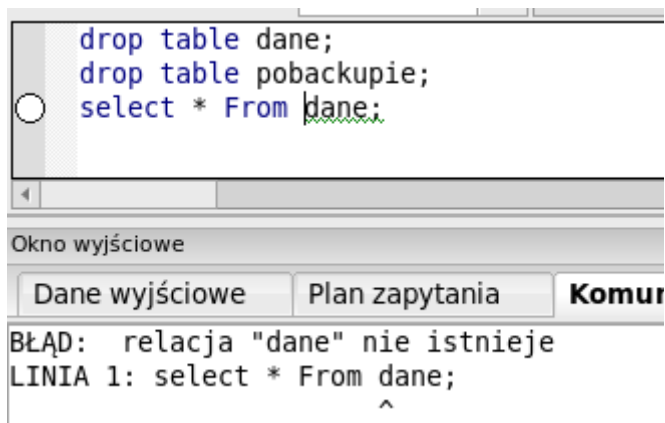
Przy okazji zauważ jedną bardzo ważną rzecz:

```
mc [root@vps-1077604-8206.cp.homecloud.pl]:/var/lib/pgsql/9.4/data
Plik  Edycja  Widok  Wyszukiwanie  Terminal  Pomoc
Left  File      Command  Options      Right
'~    ~          ~        ~            ~
'n    Nazwa      Rozmiar  Czas modyfi  'n    Nazwa      Rozmiar  Czas modyfi
/..   ADRZEDN    22.03 11:43  /pg_stat_tmp      4096  4.04 20:21
/..autoinstaller  4096  29.03 04:22  /pg_subtrans      4096  20.03 16:55
/..elinks         4096  20.03 17:45  /pg_tblspc        4096  29.03 20:00
/..mc             4096  4.04 20:22  /pg_twophase      4096  20.03 16:55
/..pki           4096  12.03 15:31  /pg_xlog           20480 4.04 20:16
/..ssh           4096  31.03.2015  PG_VERSION         4 20.03 16:55
/..subversion    4096  31.03.2015  backup_label.old   204  4.04 17:24
/parallels       4096  4.04 04:03  pg_hba.conf        4489 22.03 11:39
.bash_history    20524 4.04 19:59  pg_ident.conf      1636 20.03 16:55
.bash_logout     18 20.05.2009  postgres~uto.conf  88 20.03 16:55
.bash_profile    176 20.05.2009  postgresql.conf   21290 4.04 17:22
.bashrc          331  3.09.2015  postmaster.opts    59 4.04 19:34
.bashrc.save     0 3.08.2015  postmaster.pid     73 4.04 19:34
.bashrc.save.1  432  3.08.2015  recovery~f.sample  5587 9.02 14:31
.cshrc           100 23.09.2004  recovery.done      5616 4.04 19:27
NADRZEDN
20G/80G (25%)
Porada: Makra % działają nawet w wierszu poleceń.
[root@vps-1077604-8206 data]#
1Pomoc 2Menu 3Podgląd 4Edycja 5Copy 6RenMov 7Mkdir 8Delete 9PullDn 10Kończ
```

Po zakończeniu odtwarzania nazwa pliku recovery.conf została zmieniona na recovery.done
Dzieje się tak po to, by po restarcie serwer nowu nie wszedł w tryb odtwarzania.

Przywracanie do punktu w czasie:

Procedura wygląda niemal identycznie, z tą różnicą że do pliku `recovery.conf` dodajemy jeszcze jeden parametr określający punkt w czasie do którego przywracamy. Aby się upewnić odnośnie działania tej procedury skasuję dwie tabele, a następnie przywrócę bazę do stanu przed ich skasowaniem.

A screenshot of a PostgreSQL command window. The window title is "Okno wyjściowe". It contains three tabs: "Dane wyjściowe", "Plan zapytania", and "Komunikat". The "Komunikat" tab is active and displays an error message: "BŁĄD: relacja 'dane' nie istnieje" followed by "LINIA 1: select * From dane;" with a caret pointing to the word "dane". Above the error, the SQL commands entered are: "drop table dane;", "drop table pobackupie;", and "select * From dane;".

```
drop table dane;
drop table pobackupie;
select * From dane;
```

Okno wyjściowe

Dane wyjściowe Plan zapytania **Komunikat**

BŁĄD: relacja "dane" nie istnieje
LINIA 1: select * From dane;
 ^

Zmiana i odkomentowanie parametru określającego punkt w czasie:

```
#
#recovery_target_name = ''          # e.g. 'daily backup 2011-01-26'
#
recovery_target_time = '2016-04-04 20:05:00 EST' # e.g. '2004-07-14 22:39:00 EST'
#
#recovery_target_xid = ''
#
#recovery_target_inclusive = true
```

Wskazałem czas tuż przed skasowaniem tabel. Otwieram bazę jak zwykle, a po zakończonym procesie odtwarzania sprawdzam czy na pewno mam swoje skasowane tabele:

- [-] 📁 jsystems.pl (jsystems.pl:5432)
 - [-] 📁 Bazy danych (5)
 - 🚫 druga
 - 🚫 inna
 - 🚫 kopia
 - [-] 📁 postgres
 - + 📁 Katalogi (2)
 - 🔗 Wyzwalacze Zdarzeniowe (0)
 - + 📁 Rozszerzenia (1)
 - [-] 📁 Schematy (1)
 - [-] 📁 public
 - 🔗 Porównania (0)
 - 🏠 Domeny (0)
 - 🔗 Konfiguracje FTS (0)
 - 📖 Słowniki FTS (0)
 - 📄 Parsery FTS (0)
 - 📄 Szablony FTS (0)
 - 🔗 Funkcje (0)
 - 📄 Sekwencje (0)
 - [-] 📁 Tabele (2)
 - + 📄 dane
 - + 📄 pobackupie