pg_dump, pg_dumpall, pg_restore

Poziom: podstawy **Dotyczy: PostgreSQL 9.6–17** **Czas: ~25 min**

Backup logiczny bierze dane, nie pliki — wynikiem jest zestaw poleceń `CREATE`/`COPY`, które da się odtworzyć na innej wersji silnika, na innym systemie operacyjnym, a nawet tylko dla jednej tabeli z całej bazy.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
SQL	Structured Query Language: język zapytań do relacyjnych baz danych

Fiszka

ZMIENIA SYSTEM

Zrzut jednej tabeli: pg_dump -t

```
pg_dump -t NazwaSchematu.NazwaTabeli -h NazwaSerwera -U NazwaUsera NazwaBazy >
SciezkaDoPliku
```

Zrzucamy strukturę i dane dokładnie jednej tabeli, bez reszty bazy — wygodne, gdy potrzebujemy tylko punktowej kopii przed ryzykowną operacją na tej jednej tabeli.

Przykład:

```
pg_dump -t sprzedaz.zamowienia -h pg01 -U nazwa_usera produkcja >
/home/nazwa_usera/zamowienia_dump.sql
```

Polecenie przy sukcesie nic nie wypisuje na ekran — cały wynik trafia do wskazanego pliku.

pg_dump = zrzut bazy PostgreSQL do pliku

□□ TYLKO ODCZYT

Import zrzutu w formacie plain: psql

```
psql -h NazwaSerwera -U NazwaUsera NazwaBazy < SciezkaDoPliku
```

Odtwarzamy zrzut tekstowy zwykłym klientem `psql`, tak jakby jego treść wpisano ręcznie w sesji — dokładnie ten sam mechanizm, co metakomenda `\i` opisana na stronie psql: klient wiersza poleceń.

Przykład:

```
psql -h pg01 -U nazwa_usera produkcja < /home/nazwa_usera/zamowienia_dump.sql
```

Przykład:

```
CREATE TABLE
COPY 48213
ALTER TABLE
```

□□ **psql** = PostgreSQL SQL: klient bazy PostgreSQL

□□ OSTROŻNIE · NIEODWRACALNE

Wyczyszczenie tabeli przed importem nadpisującym: TRUNCATE

```
TRUNCATE TABLE NazwaTabeli;
```

Jeśli import ma nadpisać całą zawartość tabeli, a nie dołożyć wiersze do istniejących, opróżniamy tabelę przed importem — inaczej `COPY` ze zrzutu doda swoje wiersze obok już istniejących, co przy kluczach głównych bez `ON CONFLICT` skończy się błędem duplikatu klucza. Mechanizm `TRUNCATE` opisuje strona Podstawowe operacje na bazie i użytkownikach.

Przykład:

```
TRUNCATE TABLE
```

□□ TYLKO ODCZYT

Zrzut jednego schematu w formacie custom: pg_dump -n -Fc

```
pg_dump -U NazwaUsera -d NazwaBazy -n NazwaSchematu -Fc -f SciezkaDoPliku
```

Zrzucamy cały schemat, wraz z danymi wszystkich jego tabel, w formacie binarnym gotowym pod migrację do innej bazy — dokładnie to zapytanie stoi za migracją danych ze schematu do osobnej bazy docelowej.

Przykład:

```
pg_dump -U nazwa_usera -d produkcja -n sprzedaz -Fc -f my_schema.dump
```

pg_dump = zrzut bazy PostgreSQL do pliku

TYLKO ODCZYT

Odtworzenie zrzutu custom lokalnie: pg_restore

```
pg_restore -U NazwaUsera -d NazwaBazy -n NazwaSchematu SciezkaDoPliku
```

Odtwarzamy zrzut w bazie na tym samym serwerze, na którym uruchamiamy polecenie — baza docelowa musi już istnieć, `pg_restore` sam jej nie tworzy.

Przykład:

```
pg_restore -U nazwa_usera -d docelowa -n sprzedaz my_schema.dump
```

pg_restore = odtwórz zrzut bazy PostgreSQL zapisany w formacie custom albo katalogowym

TYLKO ODCZYT

Odtworzenie zrzutu custom na zdalnej bazie: pg_restore -h -p

```
pg_restore -U NazwaUsera -h NazwaSerwera -p NumerPortu -d NazwaBazy -n NazwaSchematu
SciezkaDoPliku
```

Ten sam mechanizm, ale połączenie idzie do innego serwera — dokładnie to, czego potrzebuje migracja danych z jednej instancji do drugiej, na przykład przy przenoszeniu środowiska testowego z serwera produkcyjnego.

Przykład:

```
pg_restore -U nazwa_usera -h pg02 -p 5432 -d docelowa -n sprzedaz my_schema.dump
```

□□ **pg_restore** = odtwórz zrzut bazy PostgreSQL zapisany w formacie custom albo katalogowym

□□ TYLKO ODCZYT

Zrzut samej struktury, bez danych: pg_dump --schema-only

```
pg_dump --schema-only -U NazwaUsera -d NazwaBazy -Fc -f SciezkaDoPliku
```

Zrzucamy wyłącznie definicje obiektów — `CREATE TABLE`, indeksy, ograniczenia — bez ani jednego wiersza danych. Przydatne przy przygotowaniu pustej kopii bazy pod środowisko testowe albo przy dokumentowaniu struktury schematu.

Przykład:

```
pg_dump: last built-in OID is 16383
```

□□ **pg_dump** = zrzut bazy PostgreSQL do pliku

□□ TYLKO ODCZYT

Zrzut samych danych, bez struktury: pg_dump --data-only

```
pg_dump --data-only -U NazwaUsera -d NazwaBazy -Fc -f SciezkaDoPliku
```

Odwrotność poprzedniego — zakłada, że struktura docelowa już istnieje, i przenosi wyłącznie zawartość tabel. Częsty wybór przy odświeżaniu danych w środowisku testowym, którego struktura schematu jest już aktualna.

□□ **pg_dump** = zrzut bazy PostgreSQL do pliku

□□ TYLKO ODCZYT

Zrzut równoległy: pg_dump -Fd -j

```
pg_dump -U NazwaUsera -d NazwaBazy -Fd -j NumerPortu -f SciezkaDoPliku
```

> **Pojęcie: dump równoległy.** Format `directory` pozwala `pg_dump` zrzucać wiele tabel jednocześnie, oddzielnymi procesami — flaga `-j` (jobs) ustala, ile procesów roboczych ma pracować naraz. Na bazie z wieloma dużymi, niezależnymi tabelami skraca to czas backupu proporcjonalnie do liczby rdzeni, jakie oddamy zadaniu — nie działa dla

formatów `plain` ani `custom`, tylko dla `directory`.

Przykład dla czterech procesów:

```
pg_dump -U nazwa_usera -d produkcja -Fd -j 4 -f /backup/produkcja_dump/
```

pg_dump = zrzut bazy PostgreSQL do pliku

TYLKO ODCZYT

Odtworzenie równoległe: pg_restore -j

```
pg_restore -U NazwaUsera -d NazwaBazy -j NumerPortu SciezkaDoPliku
```

Analogicznie do dumpa, `pg_restore` odtwarza wiele tabel jednocześnie, jeśli źródłem jest zrzut w formacie `directory` — dla formatu `custom` równoległość działa tylko częściowo, bo mimo że plik jest jeden, `pg_restore` potrafi rozdzielić między procesy ładowanie danych i budowę indeksów.

Przykład:

```
pg_restore -U nazwa_usera -d docelowa -j 4 /backup/produkcja_dump/
```

pg_restore = odtwórz zrzut bazy PostgreSQL zapisany w formacie custom albo katalogowym

ZMIENIA SYSTEM

Zrzut wszystkich ról klastra: pg_dumpall --roles-only

```
pg_dumpall --roles-only -U NazwaUsera > SciezkaDoPliku
```

`pg_dump` pojedynczej bazy nie zawiera definicji ról ani ich haseł — role są wspólne dla całego klastra, nie własnością jednej bazy. Przy odtwarzaniu bazy na nowym klastrze, w którym te same role jeszcze nie istnieją, ten zrzut wykonujemy przed odtworzeniem samych danych.

Przykład:

```
pg_dumpall --roles-only -U postgres > /backup/role.sql
```

◫◫ ZMIENIA SYSTEM

Zrzut wszystkich tablespace'ów klastra: pg_dumpall --tablespaces-only

```
pg_dumpall --tablespaces-only -U NazwaUsera > SciezkaDoPliku
```

Analogicznie do ról, definicje tablespace'ów — czyli wskazań na katalogi na dysku spoza `PGDATA` — należą do całego klastra, nie do pojedynczej bazy, i trzeba je odtworzyć osobno, zanim odtworzymy bazy, które z tych tablespace'ów korzystają.

Przykład:

```
pg_dumpall --tablespaces-only -U postgres > /backup/tablespaces.sql
```

Z życia wzięte

◫◫ Z ŻYCIA WZIĘTE

Przed ryzykowną migracją panelu Oli robimy `pg_dump` samej tabeli `zamowienia` z bazy produkcyjnej na pg01, zamiast całej bazy. Import na środowisku testowym kończy się błędem duplikatu klucza, bo zapomnieliśmy wcześniej zrobić `TRUNCATE` tabeli docelowej. Po wyczyszczeniu tabeli drugi import przechodzi bez problemu.

Backup logiczny a backup fizyczny

Pojęcie: backup logiczny. Zrzut danych w postaci poleceń SQL albo ustrukturyzowanego formatu pośredniego, wykonany przez odpytanie bazy, a nie przez skopiowanie plików z dysku — w odróżnieniu od backupu fizycznego, jakim zajmuje się pgBackRest: podstawowe komendy, backup logiczny nie wymaga zatrzymania instancji ani identycznej wersji silnika przy odtwarzaniu. Ogólny podział backupów rozwija strona Backup pełny, różnicowy, przyrostowy, logi — `pg_dump` w tym podziale jest zawsze backupem pełnym pojedynczej bazy albo jej fragmentu, nigdy przyrostowym.

`pg_dump` zrzuca jedną bazę danych, `pg_dumpall` zrzuca cały klaster wraz z rolami i tablespace'ami, których `pg_dump` celowo nie dotyka, bo to obiekty wspólne dla całego klastra, opisane na stronie Architektura PostgreSQL: klaster, baza, schemat, procesy, a nie własność jednej bazy. `pg_restore` odtwarza zrzut zapisany w jednym z formatów binarnych `pg_dump` — zrzutu w formacie tekstowym `pg_restore` nie obsługuje, ten odtwarza się zwykłym `psql`.

Formaty zrzutu

Format	Flaga	Charakterystyka	Odtwarzanie
plain	`-Fp` (domyślny)	zwykły tekst SQL, czytelny i edytowalny w edytorze	`psql < plik.sql`
custom	`-Fc`	format binarny, skompresowany, pozwala na selektywne odtworzenie wybranych obiektów	`pg_restore`
directory	`-Fd`	katalog z osobnym plikiem na każdą tabelę, jedyny format wspierający równoległy dump i restore jednocześnie	`pg_restore`
tar	`-Ft`	archiwum tar, binarne jak custom, ale bez kompresji i bez możliwości zmiany kolejności obiektów przy odtwarzaniu	`pg_restore`

Format `plain` sprawdza się przy małych bazach i przy zrzutach, które chcemy przejrzeć albo zmodyfikować ręcznie przed odtworzeniem. Format `custom` jest domyślnym wyborem do backupu produkcyjnego — kompresja oszczędza miejsce, a `pg_restore` pozwala odtworzyć tylko wybrane tabele ze zrzutu całej bazy bez ponownego dumpowania. Format `directory` wybieramy, gdy zależy nam na czasie — to jedyny format, w którym zarówno zrzut, jak i odtworzenie mogą działać wieloma procesami naraz.

Częste błędy

Próba odtworzenia zrzutu custom klientem psql. Format binarny `-Fc` nie jest tekstem SQL — `psql < plik.dump` kończy się nieczytelnym błędem, bo próbuje zinterpretować binarną zawartość jako polecenia SQL. Formaty `custom`, `directory` i `tar` odtwarzamy wyłącznie `pg_restore`.

Pominięcie pg_dumpall --roles-only przy migracji na nowy klaster. Odtworzenie samej bazy na świeżym klastrze bez wcześniejszego odtworzenia ról kończy się błędami `role "NazwaUsera" does not exist` przy próbie przywrócenia uprawnień zapisanych w zrzucie.

Powiązane: Backup pełny, różnicowy, przyrostowy, logi · pgBackRest: podstawowe komendy · Architektura PostgreSQL: klaster, baza, schemat, procesy · Podstawowe operacje na bazie i użytkownikach

Źródła: dokumentacja PostgreSQL, rozdział „pg_dump", „pg_dumpall" i „pg_restore"

Revision #43
Created 2026-09-24 23:08:10 UTC by Tomasz Konieczny
Updated 2026-10-26 14:12:35 UTC by Tomasz Konieczny