# Rozpoznanie bazy: rozmiary, obiekty, aktywność

**Poziom: podstawy** **Dotyczy: PostgreSQL 9.6–17** **Czas: ~15 min**

*Zestaw zapytań, od których zaczynamy pracę na nieznanej wcześniej instancji — co w niej siedzi, ile zajmuje i ile osób jest akurat podłączonych.*

# Fiszka

**TYLKO ODCZYT**

## Wylistowanie największych tabel i indeksów: pg_class

```
SELECT
    nspname || '.' || relname AS "Object Name", relkind AS "Object Type",
    pg_size_pretty(pg_relation_size(C.oid)) AS "size"
FROM pg_class C
LEFT JOIN pg_namespace N ON (N.oid = C.relnamespace)
WHERE nspname NOT IN ('pg_catalog', 'information_schema')
ORDER BY pg_relation_size(C.oid) DESC
LIMIT 20;
```

Sprawdzamy, co w bazie faktycznie zajmuje miejsce na dysku — pierwsze zapytanie na nowej, nieznanej instancji, zanim jeszcze zaczniemy szukać przyczyn wolnego działania. Kolumna `relkind` odróżnia zwykłą tabelę (`r`) od indeksu (`i`), widoku materializowanego (`m`) czy sekwencji (`S`). Odfiltrowanie `pg_catalog` i `information_schema` odcina katalog systemowy, żeby lista pokazywała wyłącznie obiekty naszych danych.

Przykład:

```
             Object Name              | Object Type |  size
--------------------------------------+-------------+---------
 public.zamowienia                    | r           | 2145 MB
```

```
 public.idx_zamowienia_klient_id       | i           | 412 MB
 public.zamowienia_pkey                | i           | 380 MB
 public.klienci                        | r           | 98 MB
 public.pozycje_zamowien               | r           | 76 MB
(5 rows)
```

**TYLKO ODCZYT**

## Sprawdzenie indeksów danej tabeli: pg_indexes

```
SELECT * FROM pg_indexes WHERE tablename = 'NazwaTabeli' AND schemaname =
'NazwaSchematu';
```

Sprawdzamy, jakie indeksy faktycznie istnieją na konkretnej tabeli i jaka jest ich pełna definicja — wygodniejsze do dalszego przetwarzania w skryptach niż metakomenda `\d`, bo zwraca gotowy tekst `CREATE INDEX` w kolumnie `indexdef`.

Przykład:

```
 schemaname |    tablename    |          indexname          | tablespace
|                              indexdef
------------+-----------------+-----------------------------+------------
+-------------------------------------------------------------------
 public     | zamowienia      | zamowienia_pkey             |            | CREATE UNIQUE
INDEX zamowienia_pkey ON public.zamowienia USING btree (id)
 public     | zamowienia      | idx_zamowienia_klient_id    |            | CREATE INDEX
idx_zamowienia_klient_id ON public.zamowienia USING btree (klient_id)
(2 rows)
```

**TYLKO ODCZYT**

## Sprawdzenie rozmiaru bazy danych: pg_database_size

```
SELECT pg_size_pretty(pg_database_size('NazwaBazy'));
```

Sprawdzamy łączny rozmiar bazy na dysku, wliczając wszystkie jej tabele, indeksy i pozostałe obiekty — pierwszy numer, o który pyta się przy planowaniu pojemności wolumenu albo przy szacowaniu czasu backupu opisanego na stronie pg_dump, pg_dumpall, pg_restore.

Przykład:

```
 pg_size_pretty
----------------
 4823 MB
(1 row)
```

Rozmiary wszystkich baz w klastrze naraz, bez podawania nazwy, zwraca `\l+` opisane na stronie psql: klient wiersza poleceń.

**TYLKO ODCZYT**

## Sprawdzenie ostatniej daty modyfikacji baz danych: pg_stat_file

```
SELECT (pg_stat_file('base/'||oid||'/PG_VERSION')).modification, datname
FROM pg_database;
```

> **Pojęcie: pg_stat_file.** Funkcja zwracająca metadane systemu plików (rozmiar, czas modyfikacji) dla wskazanej ścieżki wewnątrz `PGDATA` — tutaj wykorzystana jako pośredni sposób sprawdzenia, kiedy ostatnio dotknięty był katalog danej bazy, przez datę modyfikacji jej pliku `PG_VERSION`. To przybliżenie, nie precyzyjny znacznik ostatniej transakcji, ale wystarczające, żeby odróżnić bazy używane na bieżąco od takich, których nikt nie ruszał od lat.

Przykład:

```
       modification        |  datname
---------------------------+-----------
 2017-08-24 11:02:13+02    | template0
 2022-04-14 09:41:55+02    | template1
 2023-03-22 14:12:08+02    | archiwum
 2026-09-24 22:10:03+02    | produkcja
(4 rows)
```

**TYLKO ODCZYT**

## Sprawdzenie liczby aktywnych połączeń: pg_stat_activity

```
SELECT count(*) FROM pg_stat_activity;
```

Sprawdzamy, ile połączeń jest akurat otwartych do klastra, niezależnie od tego, czy coś w danym momencie wykonują — przydatne razem z limitem skonfigurowanym w `max_connections`, opisanym na stronie postgresql.conf i pg_hba.conf, żeby ocenić, jak blisko limitu jesteśmy.

Przykład:

```
 count
-------
    34
(1 row)
```

Porównanie z limitem w jednym zapytaniu:

```
SELECT count(*), current_setting('max_connections') FROM pg_stat_activity;
```

Przykład:

```
 count | current_setting
-------+-----------------
    34 | 100
(1 row)
```

# Z życia wzięte

**Z ŻYCIA WZIĘTE**

Przejmujemy po odejściu poprzedniego administratora bazę cennik na pg01, nie wiedząc, co w niej w ogóle siedzi. Zapytanie do `pg_class` pokazuje tabelę `zamowienia_archiwum` ważącą więcej niż cała reszta bazy razem wzięta. Okazuje się, że nikt nigdy nie czyścił jej z danych sprzed pięciu lat.

# Częste błędy

**Liczenie rozmiaru tabeli bez wliczenia jej indeksów.** `pg_relation_size` zwraca rozmiar samej tabeli albo samego indeksu — do łącznego rozmiaru obiektu wraz z jego indeksami i danymi TOAST służy `pg_total_relation_size`, dająca zwykle znacząco większą liczbę niż sama tabela.

**Wnioskowanie o realnej aktywności bazy wyłącznie z daty modyfikacji PG_VERSION.** Ta data zmienia się rzadko i nie odzwierciedla bieżących operacji `SELECT`/`INSERT` — do sprawdzenia faktycznej aktywności służy `pg_stat_activity` albo statystyki z `pg_stat_user_tables`, temat rozwinięty na stronie Sesje, długie zapytania, kasowanie kwerendy.

**Powiązane:** psql: klient wiersza poleceń · Sesje, długie zapytania, kasowanie kwerendy · Indeksy i plany wykonania w PostgreSQL · postgresql.conf i pg_hba.conf

**Źródła:** dokumentacja PostgreSQL, rozdział „System Catalogs" i „System Administration Functions"

---

Revision #44
Created 2026-09-24 23:08:05 UTC by Tomasz Konieczny
Updated 2026-10-26 14:12:23 UTC by Tomasz Konieczny