Przewodnik administratora > Monitorowanie bazy danych

Niniejszy artykuł zawiera polecenia służące monitorowaniu bazy danych. Większość zapytań wykonać można z poziomu psql lub pgAdmin.

Wersja PostgreSQL

Polecenie zwraca wersję serwera PostgreSQL wraz z danymi serwera, na którym został zainstalowany

```
SELECT version();
```

Rozmiar bazy danych

Polecenie zwraca rozmiar bazy danych. Wykorzystanie funkcji *pg_size_pretty* zwiększa czytelność wyniku polecenia.

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

Zmiana nazwy bazy danych

W przypadku zajścia konieczności zmiany nazwy bazy danych należy wykonać takie polecenie (z poziomu połączenia do innej bazy):

```
ALTER DATABASE edokumenty_2 RENAME TO edokumenty);
```

Sprawdzenie ile pamięci RAM zajmuje Postgres

```
#ps -u postgres o pid=   | tr -d ' ' | sed 's#.*#/proc/&/smaps#' | xargs sudo grep ^Pss: | awk '{A+=$2} END{print A}'
```

Wynik podawany jest w kB, na podstawie analizy plików smaps

Sprawdzenie aktualnie wykonywanych zapytań

Od wersji postgresql 9.4:

```
SELECT (now() - pg_stat_activity.xact_start) AS age,
pg_stat_activity.datname, pg_stat_activity.pid,
pg_stat_activity.usename,
pg_stat_activity.query_start, pg_stat_activity.client_addr,
pg_stat_activity.client_port, pg_stat_activity.query
FROM pg_stat_activity
WHERE (pg_stat_activity.xact_start IS NOT NULL)
ORDER BY pg_stat_activity.xact_start;
```

Sprawdzenie "poziomu zaśmiecenia" tabel w bazie

```
SELECT psut.relname,
    to_char(psut.last_vacuum, 'YYYY-MM-DD HH24:MI') as last_vacuum,
    to_char(psut.last_autovacuum, 'YYYY-MM-DD HH24:MI') as last_autovacuum,
    to_char(pg_class.reltuples, '9G999G999G999') AS n_tup,
    to_char(psut.n_dead_tup, '9G999G999G999') AS dead_tup,
    to_char(CAST(current_setting('autovacuum_vacuum_threshold') AS bigint)
        + (CAST(current_setting('autovacuum_vacuum_scale_factor') AS numeric)
           * pg_class.reltuples), '9G999G999G999') AS av_threshold,
    CASE
        WHEN CAST(current_setting('autovacuum_vacuum_threshold') AS bigint)
            + (CAST(current_setting('autovacuum_vacuum_scale_factor') AS numeric)
               * pg_class.reltuples) < psut.n_dead_tup
        THEN '*'
```

```
        ELSE ''
    END AS expect_av
FROM pg_stat_user_tables psut
    JOIN pg_class on psut.relid = pg_class.oid
ORDER BY 1;
```

Sprawdzenie wielkości tabel

```
SELECT pgn.nspname, relname, pg_size_pretty(relpages::bigint * 8 * 1024) AS size
FROM pg_class pg, pg_namespace pgn
WHERE pg.relnamespace = pgn.oid AND pgn.nspname NOT IN ('information_schema', 'pg_catalog')
ORDER BY relpages DESC;
```

Rozszerzone informacje

```
SELECT pgn.nspname, relname, pg_size_pretty(relpages::bigint * 8 * 1024) AS size,
CASE WHEN relkind = 't' THEN (SELECT pgd.relname FROM pg_class pgd WHERE pgd.reltoastrelid = pg.oid) WHEN nspname = 'pg_to
FROM pg_class pg, pg_namespace pgn
WHERE pg.relnamespace = pgn.oid AND pgn.nspname NOT IN ('information_schema', 'pg_catalog')
ORDER BY relpages DESC;
```

Zamykanie połączeń do bazy danych

Jeżeli chcemy usunąć bazę danych, a istnieją połączenia do bazy, to należy je zamknąć. Dla PostgreSQL w wersji < 9.2

```
SELECT pg_terminate_backend(pg_stat_activity.procpid)
FROM pg_stat_activity
WHERE pg_stat_activity.datname = 'TARGET_DB'
 AND procpid <> pg_backend_pid();
```

Dla wersji >= 9.2

```
SELECT pg_terminate_backend(pg_stat_activity.pid)
FROM pg_stat_activity
WHERE pid <> pg_backend_pid()
AND pg_stat_activity.datname = 'TARGET_DB';
```

Zamykanie połączeń w przypadku długich zablokowanych zapytań

W przypadku gdy stworzymy duży raport a ten nie chce się wykonać

```
SELECT (now() - pg_stat_activity.xact_start) AS age, pg_stat_activity.pid,
pg_stat_activity.query_start, substring(pg_stat_activity.query from 0 for 100)
FROM pg_stat_activity
WHERE (pg_stat_activity.xact_start IS NOT NULL)
ORDER BY pg_stat_activity.xact_start;
```

Zamykamy procesy po PIDach. W komendzie zastępujemy PID

```
SELECT pg_cancel_backend(PID);
```

W ostateczności można skorzystać z polecenia

```
SELECT pg_terminate_backend(PID);
```

Szybkie tworzenie kopii roboczej bazy

W przypadku, gdy chcemy szybko zrobić kopię bazy danych bez robienia kopii bezpieczeństwa i odtwarzania jej na nową bazę można wykonać następujące polecenie:

```
CREATE DATABASE <baza_docelowa> WITH TEMPLATE <baza_zrodlowa> OWNER edokumenty;
```

Lub z konsoli (bash)

```
postgres#: createdb -O edokumenty -T <baza_zrodlowa> <baza_docelowa>
```

Aby ten sposób zadziałał, muszą być spełnione nastepujące warunki:

1. Baza docelowa nie może istnieć (należy ją wpierw usunąć, jeżeli istnieje).
2. Do bazy źródłowej nie może istnieć żadne otwarte połączenie. Jeżeli istnieje, to należy je zamknąć.

Operacje należy wykonywać z użytkownika będącym administratorem PostgreSQL-a.

Ilość rekordów w bazie

```
SELECT
SUM(pgClass.reltuples)::numeric AS totalRowCount
FROM
pg_class pgClass
LEFT JOIN
pg_namespace pgNamespace ON (pgNamespace.oid = pgClass.relnamespace)
WHERE
pgNamespace.nspname NOT IN ('pg_catalog', 'information_schema') AND
pgClass.relkind='r'
;
```

Sprawdzenie wielkości zużycia miejsca na dysku

```
ncdu <katalog> np. ncdu /
df -h
du -hs
```