Konfiguracja bazy danych

Hubert Albanowsski, Filip Opac

02 lip 2024

Contents

CHAPTER 1

Konfiguracja bazy danych

1.1 Lokalizacja i struktura katalogów

1.1.1 Lokalizacja:

1) **Katalog danych:**
    - `/var/lib/postgresql/<wersja>/main` na na systemach Debian/Ubuntu
    - `/var/lib/pgsql/<wersja>/data` na systemach Red Hat/CentOS.
    - Zawiera wszystkie dane, pliki konfiguracyjne, logi i pliki kontrolne.
2) **Katalog Konfiguracyjny:**
    - Pliki konfiguracyjne zwykle znajdują się w katalogu danych, choć zdarza się, że mogą znajdować się w innym katalogu np `/etc/postgresql/<wersja>/main`
3) **Katalog logów:**
    - Domyślnie `/var/log/postgresql` na Debianie/Ubuntu oraz `/var/lib/pgsql/<wersja>/data/pg_log` na Red Har/CentOS
    - zawiera logi PostgreSQL

1.1.2 Struktura katalogów:

- `base/`: Zawiera dane użytkownika dla każdej bazy danych.
- `global/`: Przechowuje dane globalne, np. tabele systemowe.
- `pg_xlog/` lub pg_wal/ (od wersji 10): Zawiera dzienniki Write-Ahead Log (WAL).
- `pg_clog/` lub pg_xact/: Przechowuje dane dotyczące transakcji.
- `pg_tblspc/`: Linki symboliczne do tabel przestrzeni.

- `pg_multixact/`: Dane dotyczące wielokrotnych transakcji.
- `pg_subtrans/`: Dane dotyczące podrzędnych transakcji.
- `pg_stat/`: Dane statystyczne.
- `pg_snapshots/`: Przechowuje dane dotyczące snapshotów.

1.2 Tabele - rozmiar, planowanie i monitorowanie

1.2.1 Rozmiar

Rozmiar tabel jest kluczowym aspektem zarządzania bazą danych, ponieważ bezpośrednio wpływa na wydajność systemu. Duże tabele mogą prowadzić do dłuższych czasów odpowiedzi na zapytania, zwiększonego zużycia pamięci oraz większych obciążeń wejścia/wyjścia.

Monitorowanie rozmiaru tabel PostgreSQL dostarcza kilka narzędzi i zapytań SQL, które umożliwiają monitorowanie rozmiaru tabel:

- **Rozmiar pojedynczej tabeli:**

```
SELECT pg_size_pretty(pg_total_relation_size('nazwa_tabeli')) AS size;
```

To zapytanie zwraca przyjazny dla użytkownika rozmiar całej tabeli, wliczając w to dane, indeksy oraz wszelkie towarzyszące pliki.

- **Rozmiar wszystkich tabel w bazie danych:**

```
SELECT relname AS "Table", pg_size_pretty(pg_total_relation_size(relid)) AS "Size"
FROM pg_catalog.pg_statio_user_tables
ORDER BY pg_total_relation_size(relid) DESC;
```

To zapytanie zwraca listę wszystkich tabel w bazie danych wraz z ich rozmiarami, posortowaną według rozmiaru.

- **Rozmiar indeksów tabeli:**

```
SELECT indexrelname AS "Index", pg_size_pretty(pg_relation_size(indexrelid)) AS "Size"
FROM pg_stat_user_indexes
WHERE relname = 'nazwa_tabeli';
```

To zapytanie zwraca listę indeksów dla określonej tabeli wraz z ich rozmiarami.

1.2.2 Planowanie

Planowanie tabel w PostgreSQL obejmuje kilka kluczowych aspektów, takich jak normalizacja, denormalizacja, indeksowanie oraz partycjonowanie.

1) **Normalizacja** - Normalizacja to proces organizowania danych w bazie danych w taki sposób, aby zminimalizować redundancję i zapewnić integralność danych. Składa się z kilku form normalnych (NF), z których każda eliminuje różne rodzaje redundancji:
    - **Pierwsza forma normalna (1NF)**: Każda komórka tabeli zawiera pojedynczą wartość, a każde pole jest atomowe.
    - **Druga forma normalna (2NF)**: Spełnia wymagania 1NF i zapewnia, że wszystkie kolumny niekluczowe są w pełni zależne od klucza głównego.

- **Trzecia forma normalna (3NF)**: Spełnia wymagania 2NF i zapewnia, że nie ma zależności przejściowych między kolumnami niekluczowymi.

2) **Denormalizacja** - Denormalizacja to proces łączenia tabel, aby poprawić wydajność zapytań kosztem zwiększenia redundancji danych. Stosuje się ją wtedy, gdy zapytania często wymagają złączeń wielu tabel, co może wpływać na wydajność.

3) **Indeksowanie** - Indeksy są kluczowe dla optymalizacji wydajności zapytań. Pozwalają one na szybsze wyszukiwanie, sortowanie i filtrowanie danych. W PostgreSQL dostępnych jest kilka typów indeksów:

   - **B-tree**: Najczęściej używany typ indeksu, odpowiedni dla większości operacji wyszukiwania.
   - **Hash**: Szybszy dla operacji równościowych, ale mniej wszechstronny niż B-tree.
   - **GIN (Generalized Inverted Index)**: Używany do indeksowania danych wielowartościowych, takich jak tabele JSONB i dokumenty pełnotekstowe.
   - **GiST (Generalized Search Tree)**: Używany dla bardziej złożonych typów danych, takich jak geometria.

Przykład tworzenia indeksu B-tree:

```
CREATE INDEX idx_nazwa_kolumny ON nazwa_tabeli(nazwa_kolumny);
```

4) **Partycjonowanie** - Partycjonowanie polega na podzieleniu tabeli na mniejsze, bardziej zarządzalne części zwane partycjami. Może to znacznie poprawić wydajność zapytań, szczególnie w przypadku dużych tabel. PostgreSQL obsługuje kilka typów partycjonowania:

   - **Partycjonowanie zakresowe (Range Partitioning)**: Dzieli dane na zakresy wartości.
   - **Partycjonowanie listy (List Partitioning)**: Dzieli dane na podstawie wartości z określonej listy.
   - **Partycjonowanie haszowe (Hash Partitioning)**: Dzieli dane na podstawie wartości haszowej.

Przykład tworzenia partycji zakresowej:

```
CREATE TABLE orders (
order_id serial PRIMARY KEY,
order_date date NOT NULL,
customer_id int NOT NULL,
amount numeric
) PARTITION BY RANGE (order_date);
CREATE TABLE orders_2022 PARTITION OF orders
FOR VALUES FROM ('2022-01-01') TO ('2023-01-01');
CREATE TABLE orders_2023 PARTITION OF orders
FOR VALUES FROM ('2023-01-01') TO ('2024-01-01');
```

1.2.3 Monitorowanie

Dodatkowo możlive jest monitorowanie wydajności tabel za pomocą:

- **pg_stat_user_table:**

```
SELECT relname, seq_scan, seq_tup_read, idx_scan, idx_tup_fetch, n_tup_ins, n_tup_upd, n_
↪tup_del
FROM pg_stat_user_tables;
```

- **pg_stat_activity:**

```
SELECT pid, usename, datname, state, query_start, query
FROM pg_stat_activity;
```

1.3 Podstawowe parametry konfiguracyjne

1.3.1 Plik postgresql.conf

Plik *postgresql.conf* zawiera ustawienia dotyczące wydajności, logowania, sieci i wielu innych aspektów.

Kluczowe ustawienia:

1) **Słuchanie połączeń:**

```
listen_addresses = 'localhost'  # Adresy IP, na których PostgreSQL będzie nasłuchiwać␣
↪połączeń
port = 5432                     # Port, na którym PostgreSQL będzie nasłuchiwać połączeń
```

2) **Pamięć i wydajność:**

```
shared_buffers = 128MB           # Ilość pamięci RAM przeznaczona na buforowanie danych
work_mem = 4MB                   # Ilość pamięci RAM na operacje sortowania i agregacji␣
↪na użytkownika
maintenance_work_mem = 64MB      # Ilość pamięci RAM na operacje utrzymaniowe (np.␣
↪VACUUM, CREATE INDEX)
```

3) **Autovacuum:**

```
autovacuum = on                  # Automatyczne czyszczenie i analiza tabel
autovacuum_naptime = 1min        # Częstotliwość uruchamiania procesu autovacuum
```

1.3.2 Plik pg_hba.conf

Plik **pg_hba.conf** odpowiada za kontrolę dostępu do bazy danych PostgreSQL.

Przykład konfiguracji:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# Zezwól lokalnym użytkownikom na połączenie
local   all             all                                     md5

# Zezwól zdalnym użytkownikom z sieci 192.168.1.0/24 na połączenie
host    all             all             192.168.1.0/24          md5
```

1.3.3 Plik pg_ident.conf

Plik **pg_ident.conf** pozwala mapować systemowych użytkowników do użytkowników PostgreSQL.

Przykład konfiguracji:

```
# MAPNAME       SYSTEM-USERNAME         PG-USERNAME

mymap           johndoe                 john
mymap           janedoe                 jane
```

W pliku **pg_hba.conf** można użyć tej mapy:

```
host    all             all             127.0.0.1/32            ident map=mymap
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search