

PostgreSQL dla programistów PHP — praktyczne porady

 Jan Kowalski  26.09.2026  297 wyświetleń  3 komentarzy

 Dostępne w językach:

 PL

 EN

[🏠 Start](#) / [Blog](#) / [Bazy Danych](#)

/ PostgreSQL dla programistów PHP — praktyczne porady

[← Wróć do bloga](#)



Pobierz PDF

Dlaczego PostgreSQL?

PostgreSQL to nie tylko „darmowy MySQL”. To zaawansowany system baz danych z funkcjami, których MySQL nie ma:

- pełne wsparcie dla **JSON/JSONB** z indeksami GIN
- **tablice** jako typ kolumny
- **Full-text search** bez osobnych tabel
- **CTE** (WITH queries) dla złożonych zapytań
- rozszerzenia: `pgcrypto` , `uuid-oss` , `pgvector`

Konfiguracja w Symfony

```
DATABASE_URL="postgresql://user:secret@localhost:5432/dbname"
```

Doctrine automatycznie dobierze odpowiedni sterownik i dialekt SQL.

Migracje

```
# generuje plik migracji na podstawie różnicy encji
php bin/console doctrine:migrations:diff
```

```
# wykonuje oczekujące migracje
php bin/console doctrine:migrations:migrate
```

```
# status migracji
php bin/console doctrine:migrations:status
```

Zasada: nigdy nie edytuj ręcznie tabel na produkcji. Zawsze przez migracje.

Indeksy – klucz do wydajności

W Doctrine dodajesz indeksy przez atrybuty na encji:

```
# [ORM\Entity]
#[ORM\Table(name: 'blog_articles')]
#[ORM\Index(fields: ['status'], name: 'idx_status')]
#[ORM\Index(fields: ['publishedAt'], name: 'idx_publishedAt')]
class BlogArticle
```

```
{  
    // ...  
}
```

Bezpośrednio w SQL:

```
-- indeks na kolumnie status  
CREATE INDEX idx_articles_status ON blog_articles(status)  
  
-- indeks złożony (status + data, malejąco)  
CREATE INDEX idx_articles_feed ON blog_articles(status, published_at)  
WHERE status = 'published';
```

EXPLAIN ANALYZE

Najważniejsze narzędzie do diagnozowania wolnych zapytań:

```
EXPLAIN ANALYZE  
SELECT a.*, u.name  
FROM blog_articles a  
JOIN users u ON u.id = a.author_id  
WHERE a.status = 'published'  
ORDER BY a.published_at DESC  
LIMIT 10;
```

Szukaj w wyniku:

- `Seq Scan` — pełny skan tabeli (zły znak przy dużych tabelach)
 - `Index Scan` — używa indeksu (dobrze)
 - `cost=` — szacunkowy koszt
 - `actual time=` — realny czas w ms
-

JSONB – elastyczne dane

Kiedy nie wiesz z góry, jakie pola będzie miał rekord:

```
ALTER TABLE blog_articles ADD COLUMN meta JSONB DEFAULT '[]';

-- zapisz dowolne dane
UPDATE blog_articles
SET meta = '{"reading_time": 5, "featured": true, "tags": ["PHP"]}'
WHERE id = 1;

-- zapytaj po wartości JSON
SELECT title FROM blog_articles
WHERE meta->>'featured' = 'true';

-- indeks GIN na całym JSON
CREATE INDEX idx_articles_meta ON blog_articles USING GIN (meta);
```

Porównanie PostgreSQL vs MySQL

Funkcja	PostgreSQL	MySQL
JSON z indeksami	✓ JSONB + GIN	⚠ ograniczone
Tablice	✓	✗
Full-text search	✓ wbudowany	⚠ słabszy
Transakcje DDL	✓	✗
pgvector (AI)	✓	✗
Popularność	rosnąca	malejąca

🔖 Tagi:

#PHP

#Symfony

#PostgreSQL

Udostępnij:

 Facebook

 X

 LinkedIn

 Kopiuj link

Komentarze (3)

P

Piotr Wiśniewski 21.10.2026, 14:27

JSONB z indeksami GIN to strzał w dziesiątkę dla elastycznych konfiguracji.

J

Jan Kowalski 21.10.2026, 14:27

Dokładnie! Widziałem aplikacje z 2000ms na zapytaniu, po dodaniu indeksu – 3ms.

A

Anna Nowak 21.10.2026, 14:27

Sekcja EXPLAIN ANALYZE powinna być w każdym kursie SQL. Naprawdę otwiera oczy.

[Zaloguj się](#), aby dodać komentarz.

FactoryCode

Projekt demonstracyjny zbudowany na Symfony 8, PHP 8.4 i PostgreSQL 16.
Tłumaczenia zarządzane przez panel admina z AI.

NAWIGACJA

 Start

 Blog

 Projekty

 Kontakt

 Polityka prywatności

KONTAKT

✉ kontakt@factorycode.pl

🐙 [GitHub](#)

© 2026 FactoryCode. Wszelkie prawa
zastrzeżone.

Zbudowane z [Symfony 8](#) & [Claude](#)
[AI](#)