**Wikiprint Book**

**Title: Optymalizacja zapytań PostgreSQL**

**Subject: eDokumenty - elektroniczny system obiegu dokumentów, workflow i CRM - DeployerGuide/OptimisingQueries**

**Version: 7**

**Date: 07/29/25 09:54:40**

## Table of Contents

*Optymalizacja zapytań PostgreSQL* 3
*Testowanie zapytań* 3
*Przykłady* 3
*Przykład nr 1* 3
*Instrukcje dla optymalizatora* 3

# Optymalizacja zapytań PostgreSQL

Zasady:

1. Unikaj podzapytań. Zamiast tego wybieraj dane za pomocą JOIN-ów
1. Najmniejsza tabela pierwsza (lub taka na której warunek jest najbardziej efektywny)
1. Dla zapytań wykonywanych na wielu tabelach na których przeprowadzono optymalizację

## Testowanie zapytań

Dla 20 jednoczesnych klientów, używając 4 wątków, przez maksymalnie 300 sec wykonuj zapytanie zawarte w pliku:

```
edokumenty$ pgbench -c20 -T300 -j4 -f tests/query.sql edokumenty -p5432
```

## Przykłady

### Przykład nr 1

```
-- Przed
SELECT t1.id, a, b, c,
(SELECT d FROM table2 t2 WHERE t2.id = t1.id) AS d
 FROM table t1;

-- Po
SELECT t1.id, a, b, c, t2.d
 FROM table t1
 INNER JOIN t2 ON t1.id = t2.id
```

## Instrukcje dla optymalizatora

Przykładowe instrukcje dla optymalizatora które można zadeklarować np. przed wykonaniem zapytania. (GUC)

```
-- Zwiększ zasoby procesora dla zapytania
SET cpu_table_cost = 0.15;
--
SET enable_nestloop = FALSE;
--
SET enable mergejoin = FALSE;
```

Jesli jesteśmy pewni że nasze zapytanie zostało już najlepiej zoptymalizowane pod kątek kolejności JOIN, to można zadeklarować: nie przestawiaj ani nie sprawdzaj kolejności tabel:

```
SET join_collapse_limit = 1;
```