**Porównanie wydajności złączeń i zagnieżdżeń w PostgreSQL i MySQL**

Bartosz Skóra

411144

Obraz zawierający logo, Czcionka, tekst, symbol

Opis wygenerowany automatycznie

**1.Wstęp**

Ćwiczenie wykonano w celu zbadania i porównania wydajności różnych systemów baz danych odnośnie zapytań łączących i zagnieżdżających rekordy w tabelach stratygraficzna, wypełnionej na podstawie Tabela stratygraficzna (według ICS, kwiecień 2023), oraz tabeli milion, zawierającej milion rekordów wypełnionych liczbami.

Analizę wykonano używając systemów:

* PostgreSQL
* MySQL

Środowisko jakie użyto do wykonywania pracy:

* DataGrip 2023.1.2(Windows 64x)

**2.Parametry systemu**

PC:

* CPU: AMD Ryzen 9 3900X 12-Core Processor 3.80 GHz
* GPU: AMD Radeon RX 5700 XT 8GB GDDR6
* RAM: G.Skill Trident Z ,DDR4, 32 GB, 3600MHz, CL17
* SSD: SSD Samsung 970 EVO Plus 1TB M.2 2280 PCI-E x4 Gen3 NVMe
* Motherboard: MSI MPG X570 GAMING PRO CARBON WIFI
* OS: Microsoft Windows 10 Home PL 64 bit OEM

IDE:

* DataGrip 2023.1.2(Windows x 64)

Tools:

* MySQL 8.0.33 for Windows x64
* PostgreSQL 15.3 for Windows x86-64

**3.Zapytania testowe**

**1ZL:**

Zapytanie 1 (1 ZL), którego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci zdenormalizowanej, przy czym do warunku złączenia dodano operację modulo, dopasowującą zakresy wartości złączanych kolumn:

*SELECT COUNT(\*) FROM geo.Milion*

*INNER JOIN geo.GeoTabela*

*ON (mod(geo.Milion.liczba,102)=(geo.GeoTabela.id\_pietro));*

**2ZL:**

Zapytanie 2 (2 ZL), którego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci znormalizowanej, reprezentowaną przez złączenia pięciu tabel:

*SELECT COUNT(\*) FROM geo.Milion*

*INNER JOIN geo.GeoPietro ON (mod(Milion.liczba,102)=geo.GeoPietro.id\_pietro)*

*NATURAL JOIN geo.GeoEpoka*

*NATURAL JOIN geo.GeoOkres*

*NATURAL JOIN geo.GeoEra*

*NATURAL JOIN geo.GeoEon;*

**3ZG:**

Zapytanie 3 (3 ZG), którego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci zdenormalizowanej, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane:

*SELECT COUNT(\*) FROM geo.Milion WHERE mod(geo.Milion.liczba,102)=*

*(SELECT id\_pietro FROM geo.GeoTabela WHERE mod(Milion.liczba,102)=(id\_pietro));*

**4ZG**

Zapytanie 4 (4 ZG), którego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci znormalizowanej, przy czym złączenie jest wyko-nywane poprzez zagnieżdżenie skorelowane, a zapytanie wewnętrzne jest złączeniem ta-bel poszczególnych jednostek geochronologicznych:

*SELECT COUNT(\*) FROM geo.Milion WHERE mod(geo.Milion.liczba,102) IN*

*(SELECT GeoPietro.id\_pietro FROM geo.GeoPietro*

*NATURAL JOIN geo.GeoEpoka*

*NATURAL JOIN geo.GeoOkres*

*NATURAL JOIN geo.GeoEra*

*NATURAL JOIN geo.GeoEon);*

**4.Wyniki testów**

Każdy test został wykonany dziesięciokrotnie, wyniki miedzy PostgreSQL i MySQL znacząco się różniły, w niektórych przypadkach nawet o parę sekund. Różnice między tabelami posiadającymi indeksy i tymi bez indeksów są znikome.

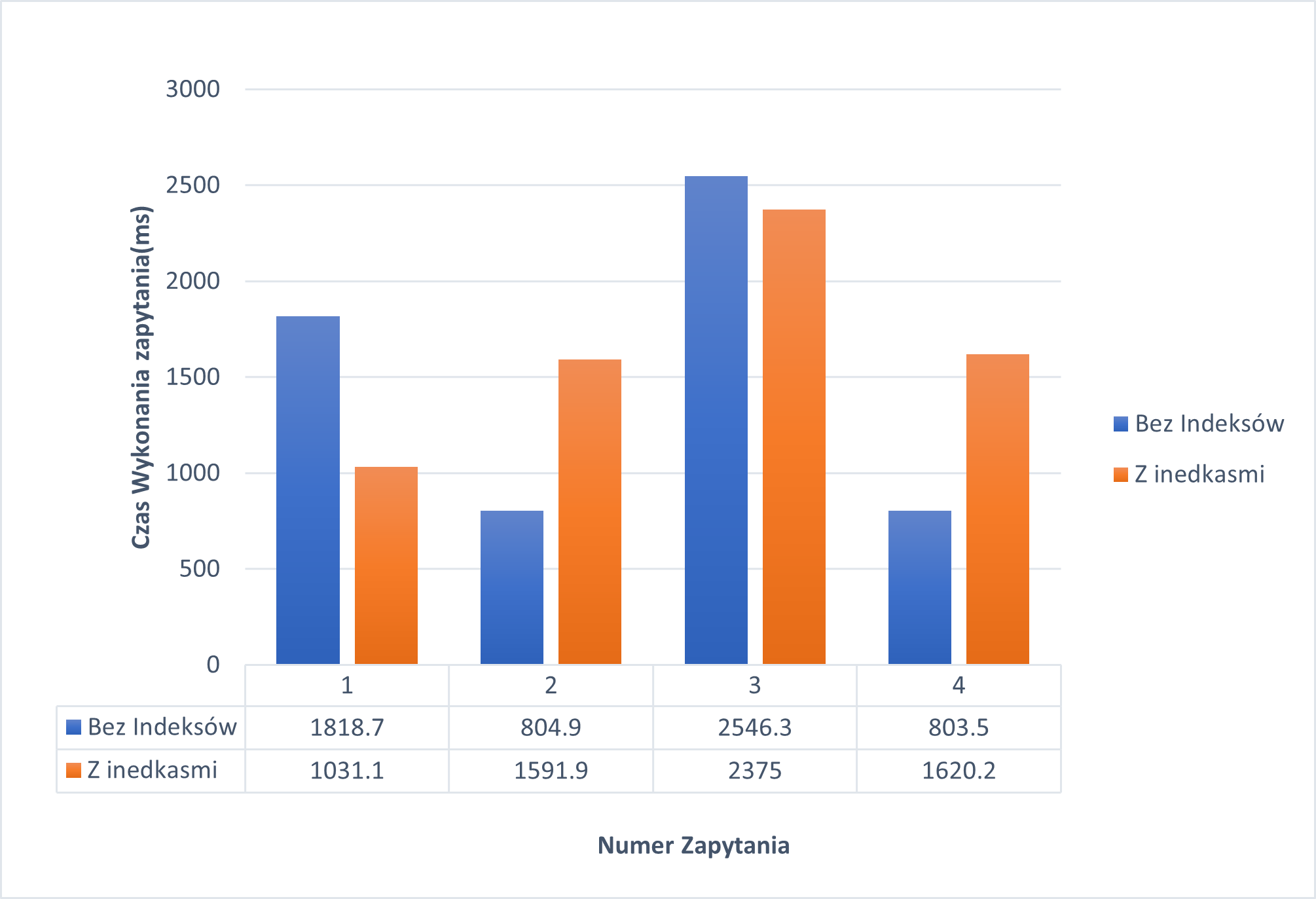
Tabela porównująca czasy wykonania testów(ms)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1ZL | 1ZL | 2ZL | 2ZL | 3ZG | 3ZG | 4ZG | 4ZG |
| BEZ INDEKSÓW | ŚR | MIN | ŚR | MIN | ŚR | MIN | ŚR | MIN |
| PostgreSQL | 162 | 98 | 589.7 | 584 | 11495.7 | 11366 | 170.1 | 168 |
| MySQL | 1818.6 | 1803 | 804.9 | 794 | 2546.3 | 2513 | 803.5 | 780 |
| Z INDEKSAMI |  |  |  |  |  |  |  |  |
| PostgreSQL | 169.9 | 167 | 390 | 382 | 11518.6 | 11423 | 172.4 | 170 |
| MySQL | 1031.1 | 984 | 1591.9 | 1567 | 2375 | 2340 | 1620.2 | 1600 |

Obraz zawierający tekst, zrzut ekranu, numer, diagram

Opis wygenerowany automatycznie

Wykres 1: Porównanie czasów wykonania testów w PostgreSQL



Wykres 2: Porównanie czasów wykonania testów w MySQL

**5.Wnioski**

Porównywając powyższe wyniki można stwierdzić że w PostgreSQL wpływ indeksowania był znikomy, jedynie w zapytaniu 2ZL można zobaczyć różnice. Natomiast w MySQL indeksowanie znacząco wpłynęło na czas wykonywania testów dla testów z tabelą zdenormalizowaną (1ZL,3ZG) skróciło czas wykonywania a dla testów z tabelą znormalizowaną (2ZL,4ZG) wydłużyło. Porównując te dwa systemy PostgreSQL znacznie lepiej radził sobie z testami 1ZL, 2ZL, 4ZG natomiast MySQL wypadł lepiej dla testu 3ZG.