Michał Szczypka

Projekt

1. Testy wydajności

W testach skupiłem się na porównaniu wydajności złączeń oraz zapytań. Testy wykonałem na jednym komputerze.

2. Konfiguracja sprzętowa i programowa

Wszystkie testy omówione były wykonane na komputerze o parametrach:

- Procesor Inter® Core™ i5-4690K CPU 3.50GHz 4 Rdzenie
- RAM 16 gb
- SSD 256gb
- S.O Windows 10
- Jako systemy zarządzania bazami danych wybrano oprogramowanie:
- MySQL 8.0 CE
- PostgreSQL, wersja 14.3-1
- Testy wykonałem 5-krotnie na moim komputerze, dla każdego systemu zarządzania bazą danych

3. Kryteria testów

W teście wykonałem szereg zapytań. Procedurę przeprowadziłem w dwóch etapach:

- Pierwszy etap obejmował zapytania bez nałożonych indeksów na kolumny danych (jedynymi indeksowanymi danymi były dane w kolumnach będących kluczami głównymi tabel)
- W drugim etapie nałożyłem indeksy na wszystkie kolumny.

Celem zasadniczym tego testu była ocena wpływu normalizacji na zapytania złożone.

Zaproponowałem cztery zapytania:

- 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

```
SELECT COUNT(*) FROM Milion INNER JOIN GeoTabela ON
(mod(Milion.liczba,68)=(GeoTabela.id_pietro));
```

- 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 Milion INNER JOIN GeoPietro ON (mod(Milion.liczba,68)=GeoPietro.id_pietro) NATURAL JOIN GeoEpoka NATURAL JOIN GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon;

- 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 Milion WHERE mod(Milion.liczba,68)= (SELECT id_pietro FROM GeoTabela WHERE mod(Milion.liczba,68)=(id_pietro));

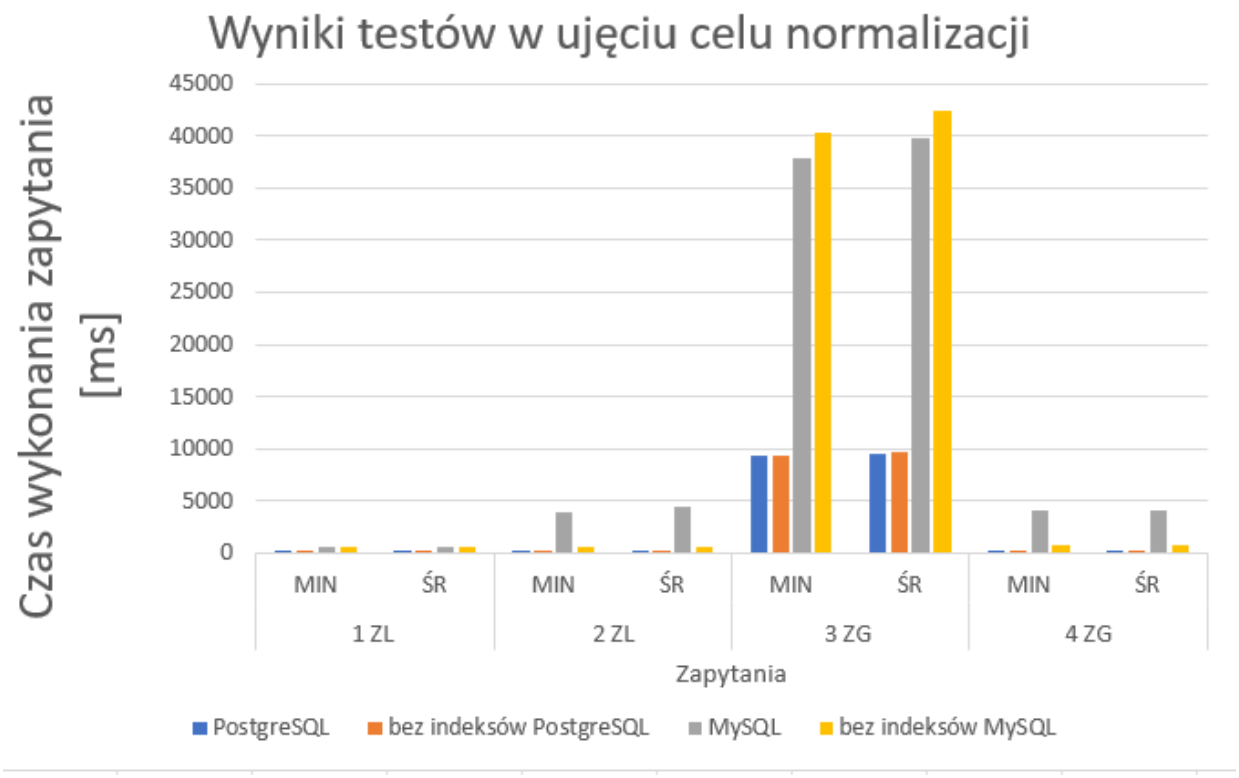
- Zapytanie 4 (4 ZG), którego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci znormalizowanej, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane, a zapytanie wewnętrzne jest złączeniem tabel poszczególnych jednostek geochronologicznych:

SELECT COUNT(*) FROM Milion WHERE mod(Milion.liczba,68)= (SELECT GeoPietro.id_pietro FROM GeoPietro NATURAL JOIN GeoEpoka NATURAL JOIN GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon;

4.Wyniki testów

Dla każdego systemu zarządzania bazą danych testy wykonałem 5 razy. Wyciągnąłem wartości minimalne oraz średnie z pięciu prób. W przypadku systemu MySQL wyniki po za zapytaniem 3 były bardzo zbliżone. Oscylują one blisko 650 milisekund bez indeksów, z indeksami różniąca ta była pomiędzy trwaniem poszczególnych zapytań jest już większa. Zapytanie czwarte i drugie są do siebie bardzo zbliżone, jednakże zapytanie trzecie i pierwsze posiadają skrajne przedziały trwania tych procesów [594,39836] milisekund.

	1 ZL		2 ZL		3 ZG			4 ZG
BEZ INDEKSÓW	**MIN**	**ŚR**	**MIN**	**ŚR**	**MIN**	**ŚR**	**MIN**	**ŚR**
MySQL	610,9	628,75	656	668	40250	42394,8	687	703,25
postgres	155	165,75	286	300	9388	9696	163	167,25
Z INDEKSAMI								
MySQL	594	609,5	4000	4359,5	37875	39836	4078	4129
postgres	143	158,25	229	239,25	9300	9511,25	150	164,25
dane podane są ms								



5.Wnioski

Z wykresu oraz z tabelki możemy wyciągnąć następujące wnioski:

- Postać zdenormalizona w PostgreSQL jest znacznie wydajniejsza, natomiast w MySQL to właśnie postać znormalizowana wykonuje zapytania znacznie krócej.
- Dla systemów zarządzania bazą danych MySQL i PostgreSQL, największym czasem wykonania zarówno w postaci znormalizowanej jak i zdenormalizowanej stanowiło zapytanie nr 3.
- Najkrótszym czasem zapytania w oby postaciach cechowało się zapytanie nr 1.
- Zagnieżdzenia skorelowane są wolniejsze w wykonaniu niż złączenia.
- PostgreSQL w większości przypadków przetwarza szybciej zapytania niż MySQL