Porównanie wydajności złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych w SQLServer i PostgreSQL.

Wiktoria Kania | 410509 | Bazy Danych

Wstęp

Świczenie ma na celu sprawozdanie i porównanie wydajności złączeń i zagnieżdżeń skorelowanych dla PostgreSQL i SQLServer. Do wykonania testów użyto wymiaru czasu na tabelach jednostek geologicznych, o dużej objętości danych. Tabele te zostały stworzone specjalnie w celach tego zadania w formie znormalizowanej jak i zdenormalizowanej.

Specyfikacja sprzętowa

Testy wykonano na komputerze o następujących parametrach technicznych:

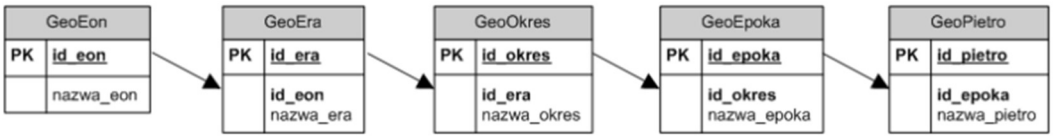
- CPU: AMD Ryzen 7 7735HS 3.20 GHz
- GPU: NVIDIA® GeForce RTX™ 4050 + AMD Radeon™ Graphics 680M
- RAM: 32 GB
- SSD: 512 GB
- SO: Windows 11 Home
- Oprogramowanie:
  - PostgreSQL 16.2
  - SQLServer 19.3

Tabela geochronologiczna

Baza składa się z tabeli geochronologicznej, gdzie główną jednostką jest eon, a następnie pomniejsze jednostki, kolejno:
Eon -> Era -> Okres -> Epoka -> Piętro

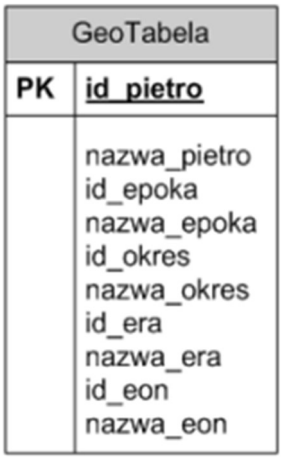
Tabela została stworzona w dwóch wariantach:

- znormalizowanym (schemat płatka śniegu)



  Stworzone zostało pięć tabel, w następstwie takim w jakim występują powyższe jednostki czasowe geologiczne.

- zdenormalizowanym (schemat gwiazdy):



Forma została otrzymana za pomocą złączenia naturalnego obejmującego wszystkie tabele ze schematu znormalizowanego:

```
CREATE TABLE GeoTabela AS (SELECT * FROM GeoPietro
NATURAL JOIN GeoEpoka
NATURAL JOIN GeoOkres
NATURAL JOIN GeoEra
NATURAL JOIN GeoEon );
```

*Dla TSQL (drugiego przypadku), konieczna była jednak modyfikacja zapytania, wprowadzając zamiast natural joina, bardziej rozbudowany inner join, ze względu na nieobsługiwalność.

Finalna tabela zawierała 77 rekordów, co wynikało z 77 pięter geologicznych, wziętych pod uwagę (od lochkowa do megalaju).

Pomocnicza tabela

Do zastosowania zapuytan testujących czas wykonywania operacji, potrzebna była tabela zawierająca liczby od 0 do 999999. Stworzona została za pomocą pomocniczej tabeli „Dziesięć”, symulującej układ dziesiętny. Sześć kolumn tabeli „Milion zostało wypełnione po kolei każdą cyfrą z tej tabeli.

Testy wydajności

Wszystkie testy złączeń zostały przeprowadzone w pierwszej kolejności bez nałożonych indeksów na kolumny danych, jedynymi indeksami jakie w tym przypadku występowały, były klucze główne w poszczególnych tabelach.

Drugi etap został rozpoczęty od indeksowania wszystkich kolumn biorących udział w złączeniu, a następnie zrealizowano te same zapytania co w pierwszym etapie.

ZAPYTANIE 1:

Celem pierwszego zapytania było 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 Milion
INNER JOIN GeoTabela ON (mod(Milion.liczba,77)=(GeoTabela.IdPietro));
```

*Przeprowadzenie operacji modulo spowodowało drobne zmiany dla tego i następnych zapytań w SQLServerze, przez drobne różnice w składniach rozwiązań. Złożoność zapytań dla obu nie zmieniła się jednak i nie powinna wpływać na ostateczne wyniki czasowe.

ZAPYTANIE 2:

Celem miało być 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,77)=GeoPietro.IdPietro)
NATURAL JOIN GeoEpoka
NATURAL JOIN GeoOkres
NATURAL JOIN GeoEra
NATURAL JOIN GeoEon;
```

ZAPYTANIE 3 :

Dla złączenia 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,77)=(SELECT IdPietro FROM GeoTabela
WHERE mod(Milion.liczba,77)=(IdPietro));
```

ZAPYTANIE 4:

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:

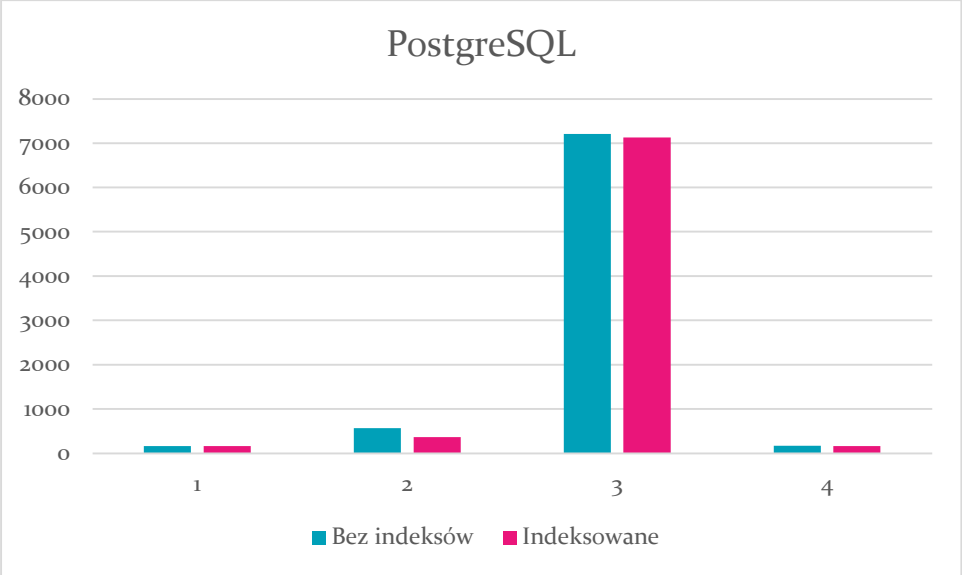
```
SELECT COUNT(*) FROM Milion
WHERE mod(Milion.liczba,77)=(SELECT GeoPietro.IdPietro FROM GeoPietro
NATURAL JOIN GeoEpoka
NATURAL JOIN GeoOkres
NATURAL JOIN GeoEra
NATURAL JOIN GeoEon;
```

Wyniki testów

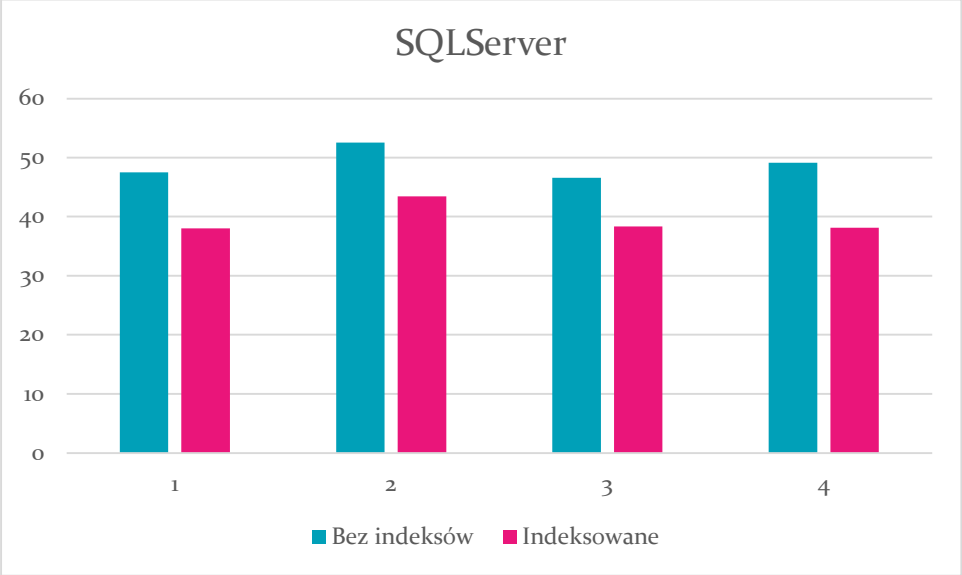
Każdy test został przeprowadzony po dwanaście razy dla każdego z zapytań. Wyniki skrajne zostały pominięte w obliczeniach. Średnie i minimalne wartości otrzymanych wyników zostały przedstawione w postaci wykresów i tabeli poniżej.

Czas[msec]	**1ZL**		**2ZL**		**3ZL**		**4ZL**	
Bez indeksów	Min	Śr	Min	Śr	Min	Śr	Min	Śr
PostgreSql	158	163	559	566	7094	7204	158	168
SqlServer	42	48	49	53	39	47	44	49
Z indeksami								
PostgreSql	153	163	351	367	7074	7131	144	163
SqlServer	33	38	39	43	35	38	25	38

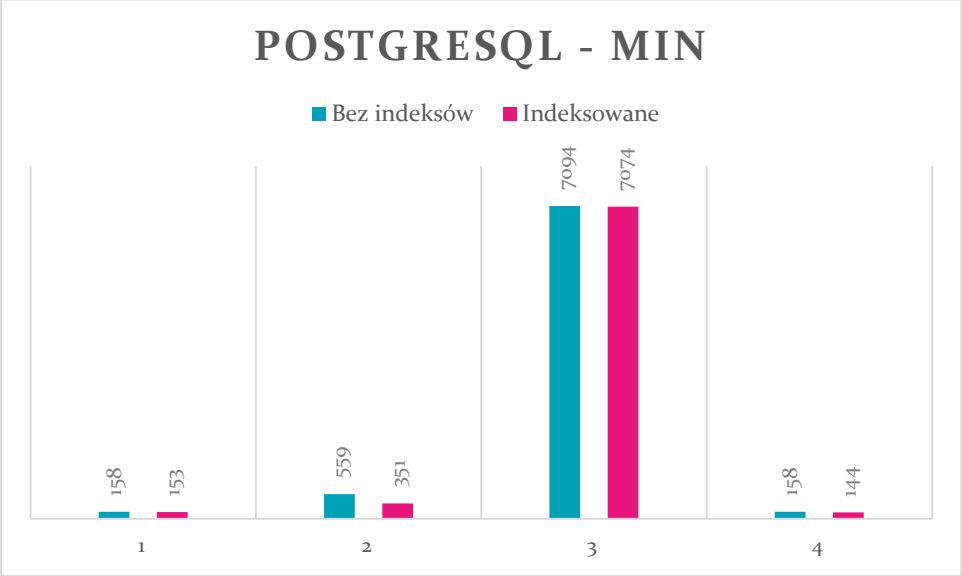
Dla lepszego zobrazowania dane zobrazowano w formie wykresów, obrazujących występowanie różnic:



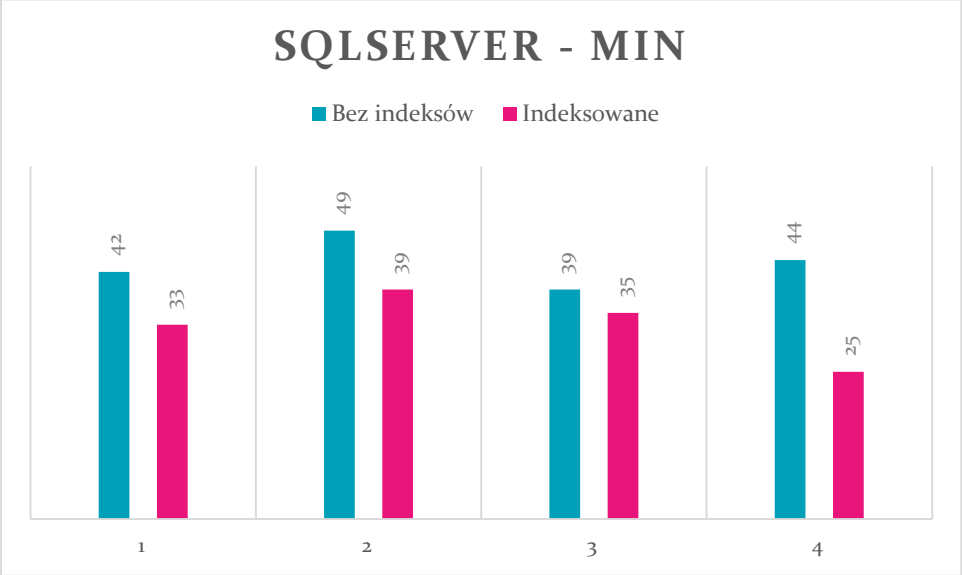
1. Zależność średnich czasów wykonywania poleceń w PostgreSql.



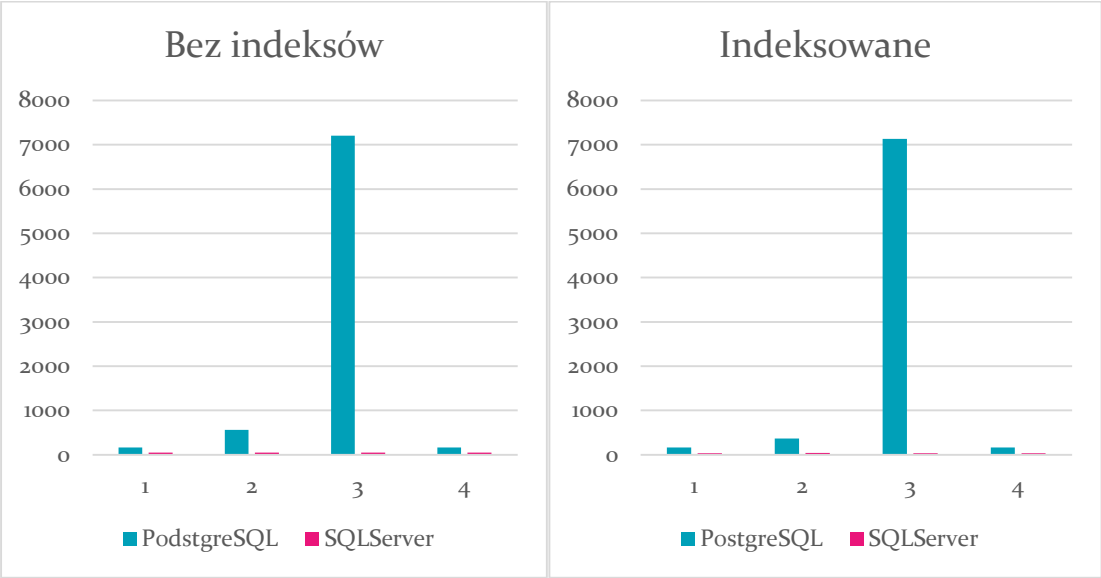
2. Zależność średnich czasów wykonywania poleceń w SqlServerze.



3. Zależność minimalnych czasów wykonywania poleceń w PostgreSql.



4. Zależność minimalnych czasów wykonywania poleceń w SqlServerze.



5. Porównanie średnich czasów między rozwiązaniami.

Wnioski

Porównując wyniki na przestrzeni SqlServera i PostgreSql można dojść do następujących wniosków:

- Czas wykonywania zapytań w MSSQL, był standardowo 3-4 krotnie mniejszy niż w PostgreSql, w przypadku trzeciego zapytania aż 153-krotnie. Skalę pokazują dwa ostatnie wykresy, na których wyniki SS są wręcz niewidoczne w porównaniu do Postgresa.

- Trzecie zapytanie dla PostgreSql, które wykorzystywało zagnieżdżenie skorelowanie, wykonywało się średnio nawet 45 razy dłużej niż reszta kwerend, co pokazuje po
- SqlServer nie wykazuje różnic w realizacji każdego z typów zapytań, jedynie pierwsze wykonane zapytanie wykazywało się większym pochłanianiem czasu, lecz było to nieznaczne lub było usuwane z obliczeń jako błąd statystyczny.
- Dla każdego z rozwiązań bazodanowych widać pozytywny wpływ wprowadzenia indeksów, choć nie jest on bardzo znaczący, ze względu na skalę bardziej odczuwalny jest dla MSSQL, gdyż operujemy tam na znacznie mniejszych czasach.

Żaden z testów nie wykazał większej wrażliwości zapytań zagnieżdżonych lub łączonych na indeksowanie, w każdym przypadku działanie to wywierało stosunkowo podobne wpływy na każdym z nich.