Maja Kruszona

Nr 408661

Geoinformatyka, sem. IV

WYDAJNOŚĆ ZŁĄCZEŃ I ZAGNIEŻDŻEŃ DLA SCHEMATÓW ZNORMALIZOWANYCH I ZDENORMALIZOWANYCH

Wstęp

Celem przeprowadzonych rozważań było porównanie wydajności złączeń i zagnieżdżeń z użyciem indeksów i ich pominięciem. Badania przeprowadzone zostały wykonane na tabelach o dużej liczbie rekordów, w wersji znormalizowanej i zdenormalizowanej.

Specyfikacje:

Do porównania wyników dane zostały sprawdzone przez dwie różne bazy danych na tym samym urządzeniu:

- PostgreSQL 14.3.1
- SQL Server 18.11.1

Konfiguracja komputera:

- CPU: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz
- RAM: Pamięć DDR4 8GB (2400MHz)
- Dysk: SSD 237 GB
- S.O.: Windows 11

Przygotowanie:

Do testów wykorzystano tabele geochronologiczną utworzoną na dwa sposoby:

1. Poprzez schemat znormalizowany. Każda tabela została utworzona osobno i zostały one połączone przez klucze obce.

```
CREATE TABLE GeoEon (id_eon int PRIMARY KEY NOT NULL,
                                nazwa VARCHAR(50) NOT NULL);
CREATE TABLE GeoEra (id_era int PRIMARY KEY NOT NULL,
                                id_eon int NOT NULL,
                                nazwa VARCHAR(50) NOT NULL,
                                CONSTRAINT fkIDeon FOREIGN KEY (id_eon) REFERENCES GeoEon(id_eon));
CREATE TABLE GeoOkres (id_okres int PRIMARY KEY NOT NULL,
                                id_era INT NOT NULL,
                                nazwa VARCHAR(50) NOT NULL,
                                CONSTRAINT fkIDera FOREIGN KEY (id_era) REFERENCES GeoEra(id_era));
CREATE TABLE GeoEpoka (id_epoka int PRIMARY KEY NOT NULL,
                                id_okres INT NOT NULL,
                                nazwa VARCHAR(50) NOT NULL,
                                CONSTRAINT fkIDokres FOREIGN KEY (id_okres) REFERENCES GeoOkres(id_okres));
CREATE TABLE GeoPietro (id_pietro int PRIMARY KEY NOT NULL,
                                id_epoka INT NOT NULL,
                                nazwa VARCHAR(50) NOT NULL,
                                CONSTRAINT fkIDepoka FOREIGN KEY (id_epoka) REFERENCES GeoEpoka(id_epoka));
```

2. Poprzez schemat zdenormalizowany Łącząc wcześniej utworzone tabele w jedną, zawierającą wszystkie dane.

```
INSERT INTO GeoTabela
SELECT  GeoPietro.id_pietro, GeoPietro.nazwa, GeoPietro.id_epoka, GeoEpoka.nazwa, GeoEpoka.id_okres,
GeoOkres.nazwa, GeoOkres.id_era, GeoEra.nazwa, GeoEra.id_eon, GeoEon.nazwa
FROM znormalizowany.GeoPietro
FULL OUTER JOIN znormalizowany.GeoEpoka
ON GeoPietro.id_epoka = GeoEpoka.id_epoka
LEFT JOIN znormalizowany.GeoOkres
ON GeoEpoka.id_okres = GeoOkres.id_okres
LEFT JOIN znormalizowany.GeoEra
ON GeoOkres.id_era = GeoEra.id_era
LEFT JOIN znormalizowany.GeoEon
ON GeoEra.id_eon = GeoEon.id_eon;
SELECT * FROM GeoTabela;
```

W tabeli geochronologicznej znalazły miejsce jednostki geochronologiczne mające wymiar czasowy (eon, era, okres, epoka i wiek) oraz odpowiadające im jednostki stratygraficzne.

Utworzenie tabeli GeoTabela ma za zadanie umożliwić szybki dostęp do wszystkich danych tabeli geochronologicznej za pomocą jednego zapytania, co nie byłoby możliwe w przypadku schematu znormalizowanego.

Do badania wydajności utworzono również tabelę *Milion*, wypełnionej kolejnymi liczbami naturalnymi od 0 do 999 999. Tabela *Milion* została utworzona na podstawie odpowiedniego autozłączenia tabeli *Dziesiec* wypełnionej liczbami od 0 do 9.

```
INSERT INTO Milion
SELECT a1.cyfra +10* a2.cyfra +100*a3.cyfra + 1000*a4.cyfra
+ 10000*a5.cyfra + 10000*a6.cyfra AS liczba,
a6.cyfra AS cyfra,
a6.bit AS bit
FROM Dziesiec a1, Dziesiec a2, Dziesiec a3, Dziesiec a4, Dziesiec a5, Dziesiec
a6 ;
```

Badanie:

W teście wykonano szereg zapytań sprawdzających wydajność złączeń i zagnieżdżeń z tabelą geochronologiczną w wersji zdenormalizowanej i znormalizowanej. Dla obu wersji tabeli sprawdzono również jak na wynik wpłynie nałożenie indeksów.

Wykorzystano do tego 4 różne zapytania:

- Z1:

Jego 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:

```
--Z1
SELECT COUNT(*) FROM Milion INNER JOIN GeoTabela ON (Milion.liczba%68)=(GeoTabela.id_pietro);
```

- Z2

Jego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci znormalizowanej, reprezentowaną przez złączenia pięciu tabel:

```
--Z2
SELECT COUNT(*) FROM Milion
INNER JOIN GeoPietro ON ((Milion.liczba%68)=GeoPietro.id_pietro)
left JOIN GeoEpoka ON GeoPietro.id_epoka = GeoEpoka.id_epoka
LEFT JOIN GeoOkres ON GeoEpoka.id_okres = GeoOkres.id_okres
LEFT JOIN GeoEra ON GeoOkres.id_era = GeoEra.id_era
LEFT JOIN GeoEon ON GeoEon.id_eon = GeoEra.id_eon;
```

- Z3

Jego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci zdenormalizowanej, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane:

```
--Z3
SELECT COUNT(*) FROM Milion WHERE (Milion.liczba%68)=
(SELECT id_pietro FROM GeoTabela WHERE (Milion.liczba%68)=(id_pietro));
```

- Z4

Jego 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 ta-bel poszczególnych jednostek geochronologicznych:

```
--Z4
SELECT COUNT(*) FROM Milion WHERE mod(Milion.liczba,68) IN
(SELECT GeoPietro.id_pietro FROM GeoPietro
INNER JOIN GeoEpoka ON GeoPietro.id_epoka = GeoEpoka.id_epoka
INNER JOIN GeoOkres ON GeoEpoka.id_okres = GeoOkres.id_okres
INNER JOIN GeoEra ON GeoEra.id_era = GeoOkres.id_era
INNER JOIN GeoEon ON GeoEon.id_eon = GeoEra.id_eon);
```

Wyniki przeprowadzonych testów:

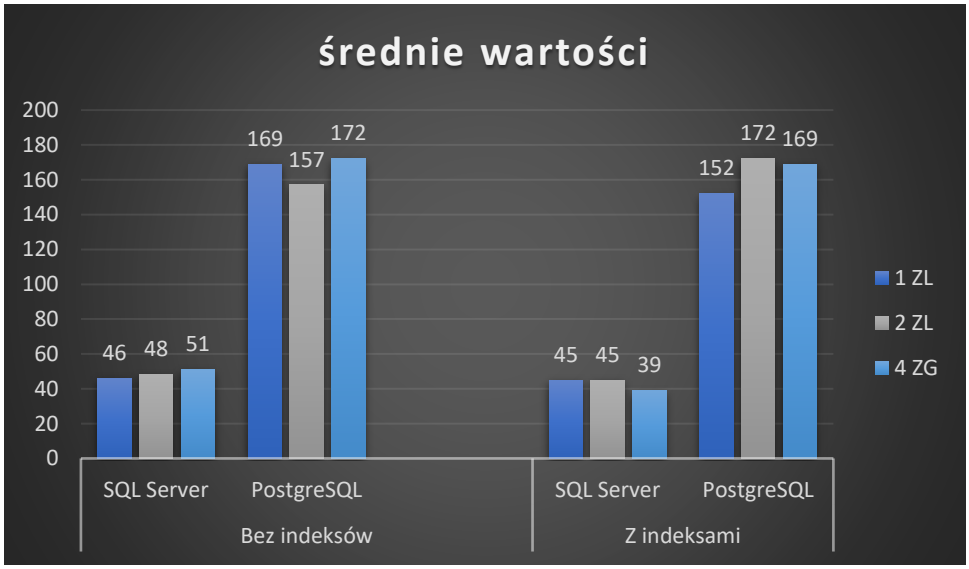
Każdy test został przeprowadzony 5-krotnie w celu pozyskania jak najlepszych statystyk. W wynikach podano wartość średnią i minimalną z tych 5 prób. W trakcie jednej z serii testów pojawił się wynik skrajny, który był najprawdopodobniej spowodowany zawieszeniem się programy, dlatego został pominięty w wynikach i zastąpiony dodatkową próbą. Wyniki, którymi jest czas wykonania zapytań, jest podany w milisekundach.

	1 ZL		2 ZL		3 ZG		4 ZG	
Bez indeksów	MIN	AVG	MIN	AVG	MIN	AVG	MIN	AVG
SQL Server	43	46	42	48	32	44	42	51
PostgreSQL	153	169	126	156	8846	8935	151	172
Z indeksami								
SQL Server	31	45	39	45	40	45	36	39
PostgreSQL	126	152	147	172	8873	8930	156	169

Podkreślone zostały wartości: najkrótszy test (czas trwania 31ms dla SQL Server), najmniejszy średni czas (równy 39ms dla SQL Servera), największy minimalny czas testu (równy 8sek 873ms) oraz największy średni czas (równy 8sek 935ms).

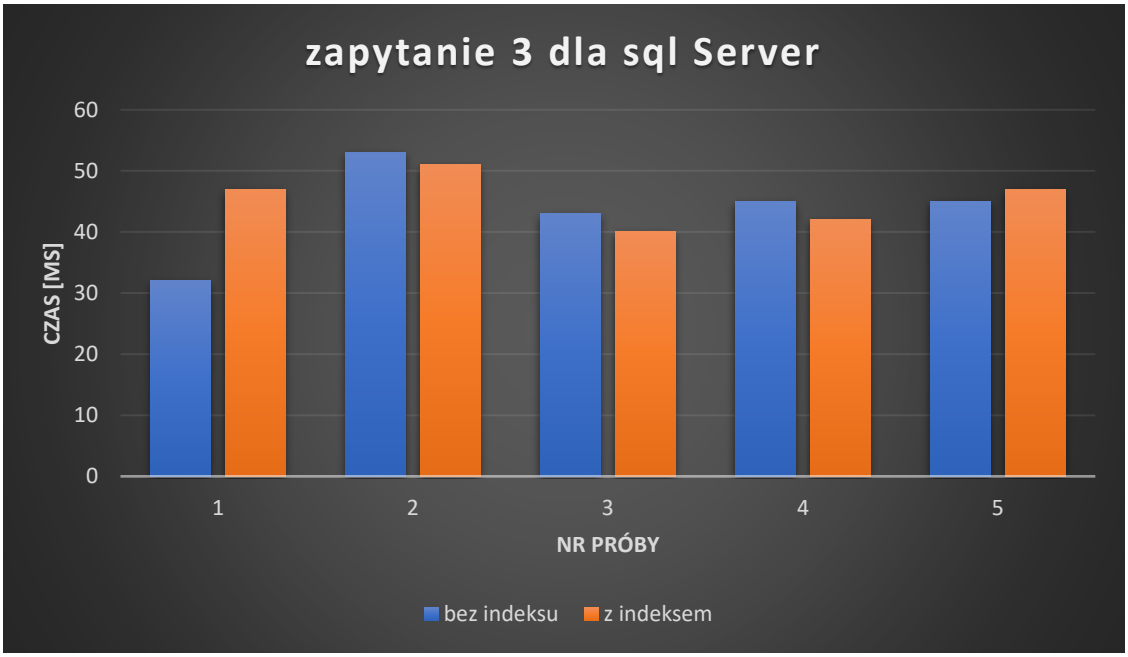
Ponieważ najkrótszym czasem wyróżnia się baza danych SQL Server, dodatkowo podkreślony został najkrótszy średni czas dla PostgreSQL (równy 152ms, w tym samym przypadku występuje również dla tej bazy najkrótszy czas minimalny). Z kolei najwyższymi wartościami wyróżnił się jedynie PostgreSQL, dlatego dodatkowo wyróżniony został najdłuższy średni czas dla SQL Servera (równy 51ms).

Do ułatwienia analizy wyników testów załączony został wykres z danymi średnich wartości 1, 2 i 4 zapytania.

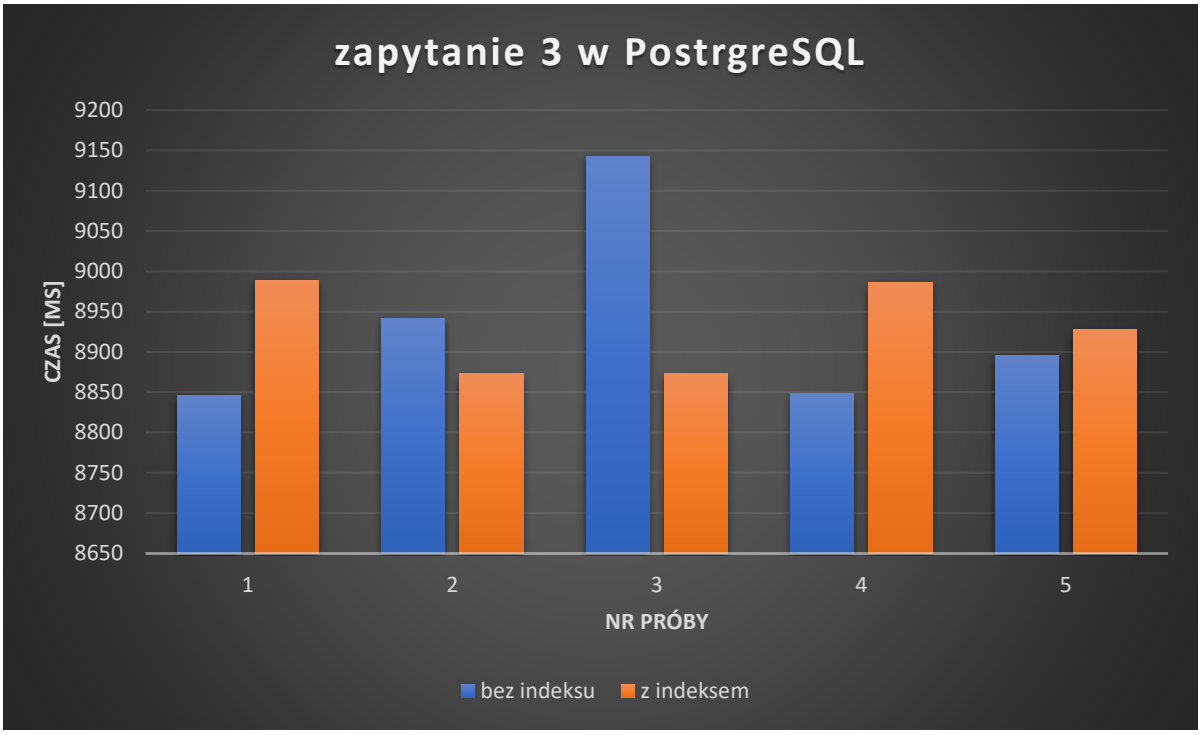


Ze względu na skrajne wartości dla każdej próby dla PostgreSQL w zapytaniu 3, wyniki z tego zapytania zostały załączone w dwóch kolejnych wykresach.

Na poniższym wykresie zostało pokazany test zapytania 3 z indeksami i bez dla SQL Server. Ukazane zostały wszystkie 5 prób dla każdego wariantu.



Na poniższym wykresie zostało pokazany test zapytania 3 z indeksami i bez dla PostgreSQL. Ukazane zostały wszystkie 5 prób dla każdego wariantu.



Wnioski:

Postać zdenormalizowana jest w większości przypadków wydajniejsza.
Przypadki, w których postać zdenormalizowana jest wolniejsza to:

- Zapytanie 2 dla PostgreSQL
- Zapytanie 3 dla SQL Server: wyniki różnią się jedynie o 1ms

Testy wykonywane były znacznie szybciej w SQL Server niż w PostgreSQL.
Zagnieżdżenia skorelowane w postaci zdenormalizowanej są znacząco wolniejsze w wykonaniu niż złączenia, co szczególnie jest uwidocznione korzystając z PostgreSQL.

Najwidoczniejsza różnica wyników jest widoczna przy zagnieżdżeniu skorelowanym w postaci znormalizowanej, gdzie dla SQL Server wynik bez indeksacji jest najwyższym średnim wynikiem tej bazy, natomiast po nałożeniu indeksów wynik jest najktrótszym średnim czasem spośród wyników dla wszystkich testów.

Podsumowanie:

Po analizie wyników oraz wniosków można stwierdzić, że normalizacja w większości przypadków prowadzi do spadku wydajności. Właściwy wybór bazy danych miał duży wpływ na wyniki, ponieważ SQL Server w każdym przypadku dawał widocznie szybsze rezultaty niż PostgreSQL.