Kamila Gądek

Nr indeksu: 408166

Kierunek: GEOINFORMATYKA

WYDAJNOŚĆ ZŁĄCZEŃ I ZAGNIEŻDŻEŃ DLA SCHEMATÓW ZNORMALIZOWANYCH I ZDENORMALIZOWANYCH

Spis treści:

1. Wstęp

Poniższy raport powstał w oparciu o pracę autorstwa mgr inż. Łukasza Jaješnicy. Nad badaniami sprawował piecze prof. nadzw. dr hab. inż. Adama Piórkowski z Akademii Górniczo Hutniczej, Katedry Geoinformatyki i Informatyki Stosowanej.
(źródło: https://docplayer.pl/42608243-Wydajnosc-zlaczen-i-zagniezdzen-dla-schematow-znormalizowanych-i-zdenormalizowanych-1.html).

Głównym tematem projektu jest analiza wydajności przeszukiwania baz danych ze znormalizowanymi i zdenormalizowanymi schematami. Tworząc bazę danych musimy pamiętać, że w większości przypadków przechowuje ona niezwykle dużą ilość informacji, więc należy zwrócić uwagę na jak najlepszą jej optymalizację, zależy nam, aby otrzymywać wyniki zapytań w relatywnie krótkim czasie.

Jeśli przeszukujemy obszerną bazę danych nieodłącznym elementem jest konieczność łączenia wielu tabel. Dokonujemy tego przy pomocy:

- Złączeń tabel (np. wewnętrznych, lewostronnych, prawostronnych)
- Zapytań zagnieżdżonych nieskorelowanych lub skorelowanych.

Trudniejsze w optymalizacji są zapytania zagnieżdżone, z szczególnym naciskiem na te skorelowane, jest to związane z tym, ze dla każdej krotki (wartości) z tabeli w zapytaniu zewnętrznym musi być każdorazowo wykonane zapytanie wewnętrzne. Natomiast złączenia tabel są najczęściej dość efektywnie wykonywane przez SZBD (Systemy Zarządzania Bazami Danych).

Podczas projektu przeprowadzono testy, których zasadniczym celem było sprawdzenie jak normalizacja wpływa na zapytania złożone – złączenia i zagnieżdżenia (skorelowane). Co prowadzi to powstania pytania, która z postaci (znormalizowana czy nie) jest wydajniejsza oraz czy warto w celu optymalizacji użyć indeksów, które rzekomo mają poprawiać wydajność zapytań.

2. Schematy tabel użytych w eksperymencie.

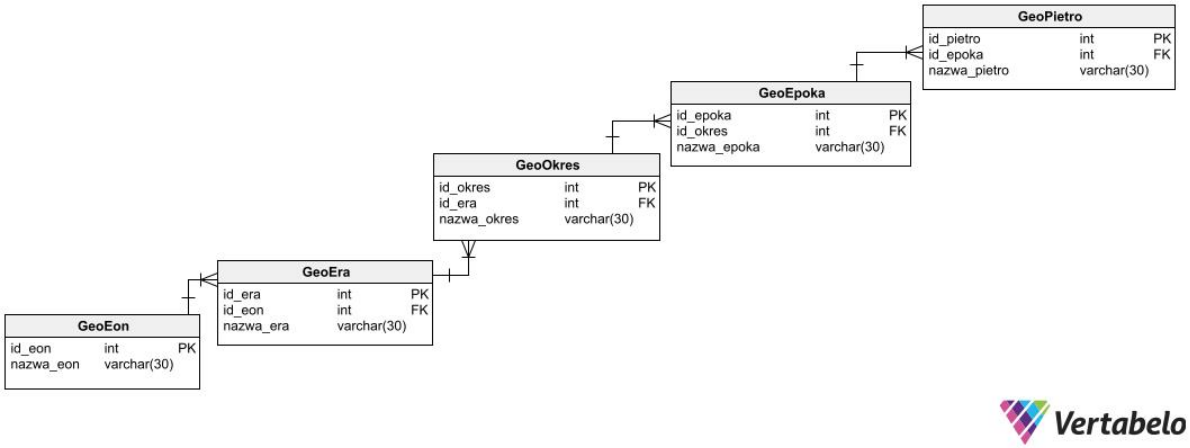
Badanie przeprowadzono na przykładzie bazy danych skonstruowanej na schemat hurtowni. Jest to centralnie zarządzana i zintegrowana baza danych, uporządkowana tematycznie oraz zawierająca wymiar czasowy (posiadająca dane historyczne). Przechowuje ona trwale dane, tzn. nie są one zmieniane ani usuwane, ewentualnie dodaje się nowe informacje. Tworząc taką bazę musimy uwzględniać przede wszystkim fakt, że zawiera ona bardzo dużą ilość danych (bieżących i historycznych) i aby nie czekać na wykonanie zapytań bardzo długo, potrzeba zapewnić odpowiednią wydajność.

Użyte w pracy tabele zostały skonstruowane wg dwóch najpopularniejszych schematów hurtowni baz danych:

- Schemat płatka śniegu – postać znormalizowana
- Schemat gwiazdy – postać zdenormalizowana

2.1. Schemat płatka śniegu.

Bazuje na schemacie gwiazdy, jest jej bardziej złożoną wersją. Posiada znormalizowane tabele (zbudowane zgodnie z modelem relacyjnej bazy danych), co w porównaniu ze schematem gwiazdy powoduje, że schemat płatka śniegu jest mniej wydajny ze względu na większą ilość relacji. Jednak posiada on strukturę łatwiejszą do modyfikacji oraz ładuje dane do tabel szybciej (z powodu normalizacji – tzn. nie ma powtarzających się informacji).



Rys. 1 Tabele z projektu w schemacie płatka śniegu.

2.2. Postać znormalizowana

Normalizacja baz danych to proces polegający na modyfikacji struktury jej tabel oraz relacji pomiędzy atrybutami, tak aby pozbyć się problemów powstałych przy tworzeniu bazy danych takich jak:

- Redundancja danych – nadmiarowość danych, powtarzanie się tych samych informacji
- Anomalia wprowadzania – niemożność wprowadzenia nowego atrybutu, ponieważ zależy on od istnienia innego atrybutu w bazie danych
- Anomalia usuwania – usunięcie niektórych atrybutów prowadzi do jednoczesnej utraty innych atrybutów w tabeli (np. są one zależne od tego usuwanego)
- Anomalia modyfikacji – aktualizacja jednej wartości/ informacji musi być dokonywana w wielu miejscach w tym samym czasie, w innym wypadku baza danych traci integralność (dane nie są spójne).

W celu uniknięcia tych ww. problemów dokonujemy normalizacji (dekompozycji) tabel oraz relacji pomiędzy nimi. Dokonujemy tego zgodnie z opracowanymi regułami doprowadzając tabelę do najoptymalniejszych dla nas postaci. Dekompozycja schematu baz danych polega w większości na rozbiciu tabel na mniejsze, nie powoduje to usunięcia żadnych danych, baza w dalszym ciągu zachowuje swoją spójność, jedynymi zmianami w strukturze jakie mogą zostać zmienione to klucze główne (zazwyczaj dodawane są nowe).

Istnieje kilka postaci normalnych, Edgar Frank Codd (prekursor normalizacji) początkowo zaproponował 3:

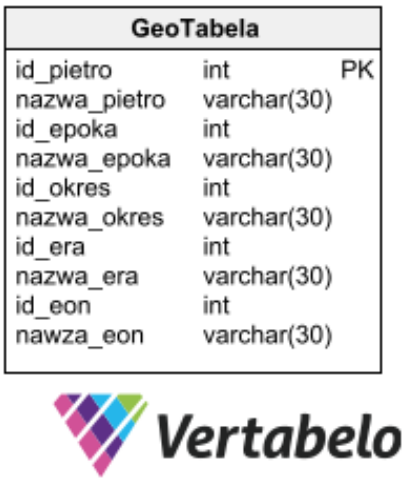
- 1NF – Pierwsza postać normalna
  Aby ją stworzyć wystarczy doprowadzić do tego, aby wartości atrybutów były atomowe, czyli niepodzielne.

- 2NF – Druga postać normalna
  Druga postać normalna powstaje, kiedy tabela:
  - jest w postaci 1NF
  - wszystkie jej kolumny niekluczowe są zależne od klucza głównego tabeli.

- 3NF – trzecia postać normalna
  W trzeciej postaci normalnej tabele:
  - są w postaci 2NF
  - nie są możliwe zależności pomiędzy atrybutami niekluczowymi
  - nie występują zależności przechodnie.

2.3. Schemat gwiazdy.

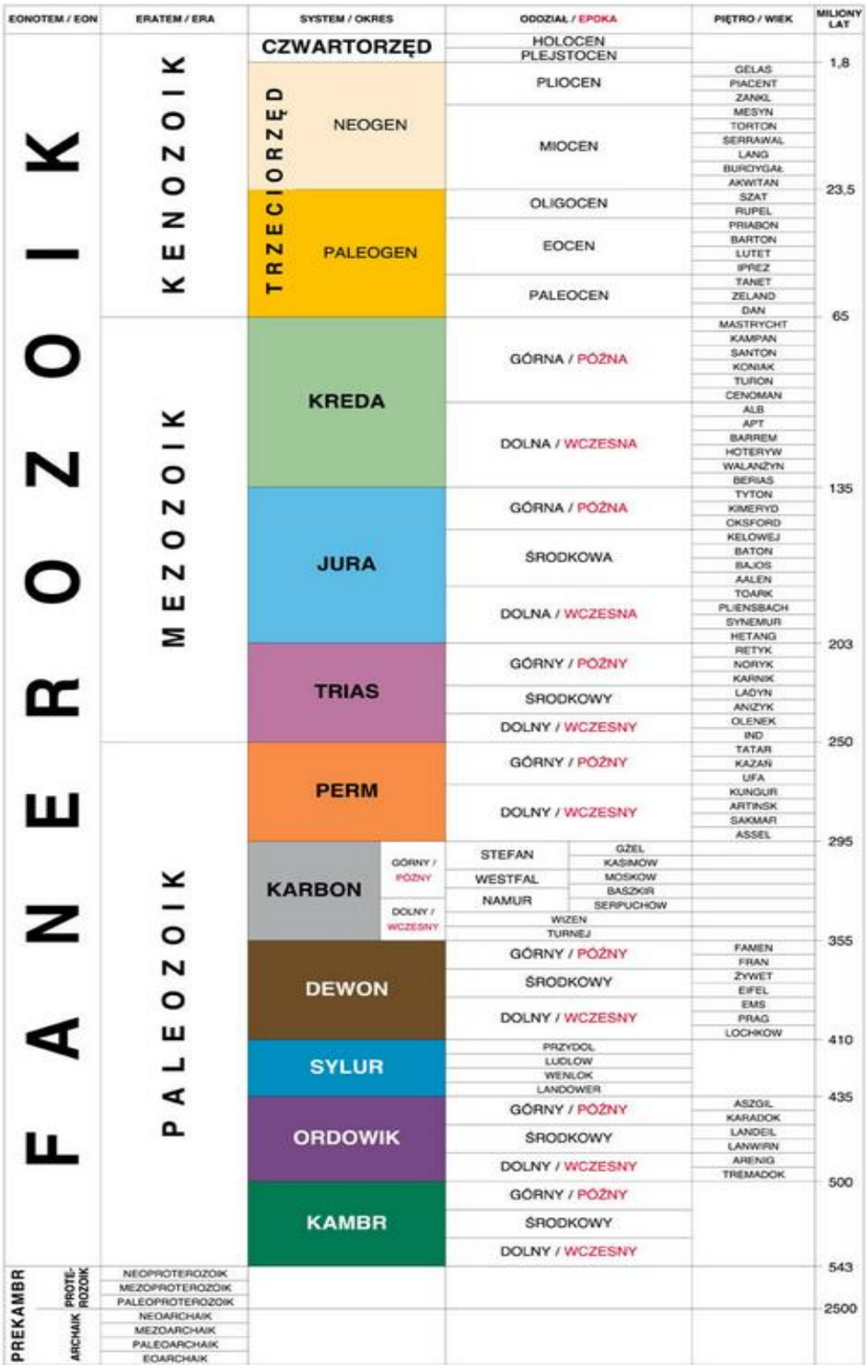
Ma dość prostą strukturę, jego głównym elementem jest tabela centralna (tabela faktów), z którą połączone są tabele wymiarów. Pozwala to na operacje agregacji, przeglądanie poszczególnych kategorii oraz filtrowanie danych. Tabela faktów jest celowo zdenormalizowana, zwiększa to wydajność i zmniejsza czas wykonywania zapytań, co wynika z małej liczby powiązań.



Rys. 2. Zdenormalizowana tabela (schemat gwiazdy)

3. Dane.

Badania przeprowadzano analizując zapytania wykonujące się na danych zawierających tabelę geochronologiczną. Przedstawia ona przebieg historii Ziemi na podstawie następstwa procesów geologicznych i układu warstw skalnych. Obecnie przyjęta tabela geochronologiczna została ustalona przez Międzynarodową Komisję Stratygrafii (ICS).



Źródło: http://stareaneksy.pwn.pl/historia_ziemi/przyklady/?pokaz=tabela

Tabela służy do datowania wieku skał oraz różnych procesów zachodzących we wnętrzu jak i w Ziemi. Bez niej praca geologów i paleontologów byłaby trudna do wykonywania. Nie ma jednej spójnej wersji tabeli, dalej trwają badania.

4. Testy wydajności.

Zgodnie z tematem pracy podczas testów sprawdzano wydajności złączeń oraz zapytań zagnieżdżonych, wykonywanych na tabelach o dużej liczbie informacji. Użyto w tym celu darmowe rozwiązania bazodanowe:

- PostgreSQL
- Microsoft SQL Server

Tabele z danymi łączono z syntetycznymi danymi o rozkładzie jednostajnym z tabeli Milion, która powstała przy pomocy autozłączenia tabeli Dziesiec wypełnionej liczbami od 0 do 9. Tabela Milion uzupełniona została kolejno liczbami od 0 do 999 999.

Tworzenie tabeli Dziesiec:

```
CREATE TABLE Dziesiec(
cyfra int,
bit int
);
```

Wprowadzanie rekordów do tabeli 10:

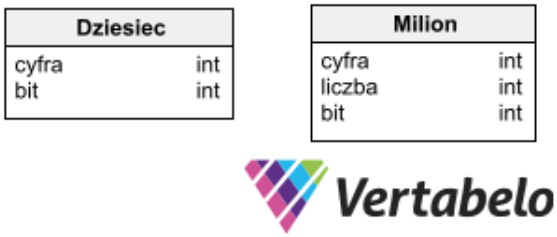
```
INSERT INTO Dziesiec VALUES(0,00000000);
INSERT INTO DZiesiec VALUES(1,00000001);
INSERT INTO Dziesiec VALUES(2,00000010);
INSERT INTO Dziesiec VALUES(3,00000011);
INSERT INTO Dziesiec VALUES(4,00000100);
INSERT INTO Dziesiec VALUES(5,00000101);
INSERT INTO Dziesiec VALUES(6,00000110);
INSERT INTO Dziesiec VALUES(7,00000111);
INSERT INTO Dziesiec VALUES(8,00001000);
INSERT INTO Dziesiec VALUES(9,00001001);
```

Tworzenie tabeli Milion **:**

```
CREATE TABLE Milion(
liczba int,
cyfra int,
bit int
);
```

Wprowadzanie rekordów do tabeli Milion przy pomocy autozłączenia tabeli Dziesiec:

```
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
```



Rys.3. Schematy tabel Dziesiec i Milion.

4.1. Konfiguracja sprzętowa i programowa.

Testy zostały wykonane na komputerze o następujących parametrach:

- CPU: AMD Ryzen 5 4500U with Radeon Graphics 2.38 GHz
- RAM: LPDDR4x 4266 MHz 16 GB
- SSD: M.2 PCIe 512 GB
- S.O.: Windows 10 Home Edition

Jako systemy zarządzania bazami danych wybrano wolno dostępne oprogramowanie:

- PostgreSQL 14.3
- Microsoft SQLServer 2019

4.2. Kryteria testów

Podczas testów sprawdzono czas wykonywania się 4 zapytań:

- **Zapytanie 1** – łączyło tablicy miliona wyników z tabelą geochronologiczną w postaci zdenormalizowanej, w warunku złączenia użyto operacji *modulo,* która dopasowała zakresy wartości złączonych kolumn:

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

- **Zapytanie** 2 – łączyło tablicę miliona wyników z tabelą geochronologiczną w postaci znormalizowanej, wykonano złączenie 5 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** – łączyło tabelę miliona wyników z tabelą geochronologiczną w postaci zdenormalizowanej, złączenie wykonano przy pomocy zagnieżdżenia skorelowanego:

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

- **Zapytanie 4** – łączyło tabelę miliona wyników z tabelą geochronologiczną w postaci znormalizowanej, złączenie wykonano przy pomocy zagnieżdżenia skorelowanego, natomiast zapytanie wewnętrzne jest zwykłym złączeniem tabel poszczególnych jednostek geochronologicznych:

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

Testy wykonano w dwóch etapach, jeden z nich obejmował wykonywanie zapytań bez nałożonych indeksów na kolumny danych (jedynymi indeksami były klucze główne poszczególnych tabel). Drugi etap wykonywano po nałożeniu indeksów na wszystkie kolumny, które biorą udział w złączeniu.

4.3. Indeksy

Indeksy są strukturami fizycznymi stosowanymi w bazach danych w celu poprawy szybkości dostępu do danych (przeszukanie tabel). Tworzone są na pojedynczych atrybutach lub zbiorach. Są pewnego rodzaju wskaźnikami do danych zawartych w tabeli. Pozwalają przyspieszyć zapytania SELECT oraz WHERE, nie warto ich jednak używać jeśli chcemy dodawać dane, bądź aktualizować (INSERT oaz UPDATE), indeksy powodują wtedy spowolnienie działania zapytań.

Indeksy użyte w testach:

```
CREATE INDEX idx_Pietro ON GeoPietro(id_epoka);
CREATE INDEX idx_Epoka ON GeoEpoka(id_okres);
CREATE INDEX idx_Okres ON GeoOkres(id_era);
CREATE INDEX idx_Era ON GeoEra(id_eon);
CREATE INDEX idx_Liczba ON Milion(liczba);
CREATE INDEX idx_GeoTabela ON GeoTabela(id_epoka, id_era, id_okres,
id_eon);
```

5. Wyniki testów.

Testy przeprowadzono w 2 różnych systemach zarządzania bazami danych. Zapytania wykonano 7 razy, w celu zmniejszenia ryzyka pojawienia się błędów pomiaru, odrzucono wyniki skrajne. Następnie wyznaczono minimum z zachowanych wartości oraz obliczono średnie czasy wykonywania.

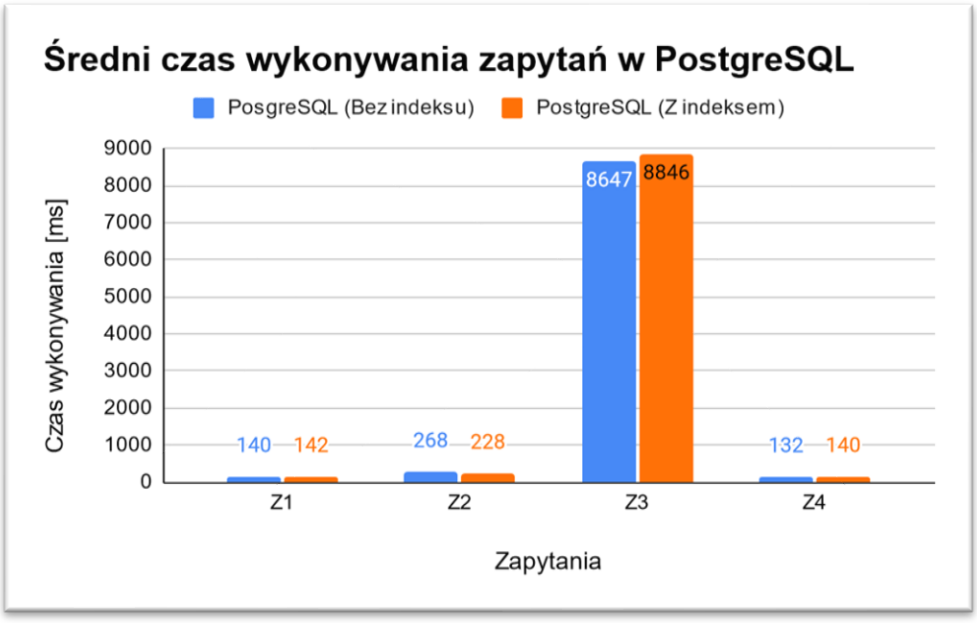
Wyniki testów przedstawiono w tabelach dla poszczególnych SZBD poniżej.

5.1. PostgreSQL

PostgreSQL bez indeksów				
	Z1	Z2	Z3	Z4
1	~~129~~	~~257~~	~~8580~~	~~126~~
2	133	258	8581	128
3	137	263	8593	129
4	140	263	8595	129
5	141	271	8643	130
6	150	283	8823	146
7	~~153~~	~~291~~	~~8974~~	~~154~~
Min	133	258	8581	128
Avg	140	268	8647	132

PostgreSQL z indeksami				
	Z1	Z2	Z3	Z4
1	~~136~~	~~214~~	~~8781~~	~~134~~
2	137	220	8785	135
3	140	221	8829	137
4	143	222	8850	138
5	144	227	8860	139
6	144	252	8907	149
7	~~147~~	~~262~~	~~8937~~	~~149~~
Min	137	220	8785	135
Avg	142	228	8846	140

Tab.1 i Tab2. Czas wykonywania zapytań



Wyk. 1. Średni czas wykonywania zapytań w PostgreSQL

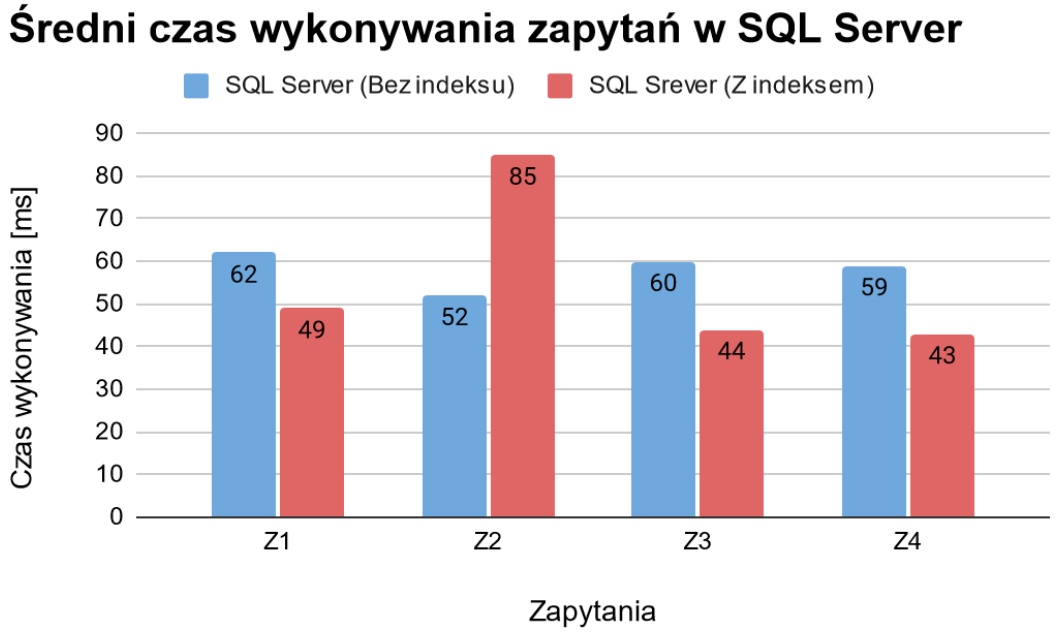
Widać na pierwszy rzut oka, że najdłużej wykonującym się zapytaniem, jest zapytanie 3, które łączy tabele poprzez zagnieżdżenie skorelowane. Można dostrzec jeszcze, że indeksy w większości przypadków nie polepszyły wydajności, a nawet (nie uwzględniając zapytania 2) wydłużyły średni czas ich wykonywania.

5.2. SQL Server

SQLServer bez indeksów				
	Z1	Z2	Z3	Z4
1	59	44	53	47
2	59	49	53	52
3	60	50	57	54
4	60	50	60	54
5	61	56	64	65
6	68	57	67	68
7	72	74	97	74
Min	59	49	53	52
Avg	62	52	60	59

SQLServer z indeksami				
	S	Z2	Z3	Z4
1	42	81	43	37
2	43	82	43	38
3	48	83	43	39
4	49	84	44	44
5	51	86	44	46
6	55	90	45	47
7	61	102	46	50
Min	43	82	43	38
Avg	49	85	44	43

Tab.3. i Tab.4. Czas wykonywania zapytań SQL Server



Wyk.2. Średni czas wykonywania zapytań SQLServer.

W Microsoft SQL Server zapytania wykonywały się w krótszym czasie niż w PostgreSQL. Widać również, że tym razem indeksy zadziałały lepiej i poza przykładem 2, zapytania wykonały się szybciej.

Analizując przypadek bez indeksów, można zauważyć, że średni czas wykonania zapytań jest mniej więcej podobny dla każdego z osobna. Wartości mieszczą w przedziale od 52 do 62 ms. Najszybszym zapytaniem jest Z2, niestety po wprowadzeniu indeksów traci na wydajności.

5.3. Zestawienie wyników

Microsoft SQL Server sprawdził się lepiej w testach wydajności, zapytania wykonywały się szybciej niż w PostgreSQL, nawet o parę rzędów wielkości (!). Niemniej jednak nie to było przedmiotem badań, a wydajność złączeń oraz zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych .

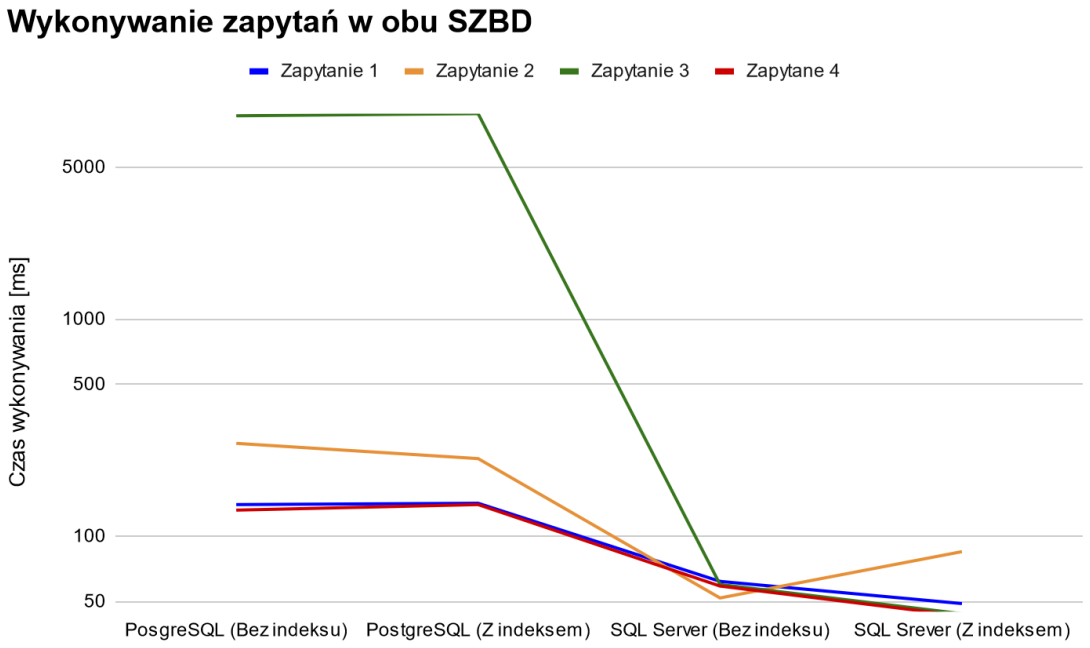
Dla przypomnienia, wykonano 4 zapytania i sprawdzono ich średni czas wykonywania się.

- Zapytanie 1 – postać zdenormalizowana, złączona
- Zapytanie 2 – postać znormalizowana, złączona
- Zapytanie 3 – postać zdenormalizowana, zagnieżdżona
- Zapytanie 4 – postać znormalizowana, zagnieżdżona

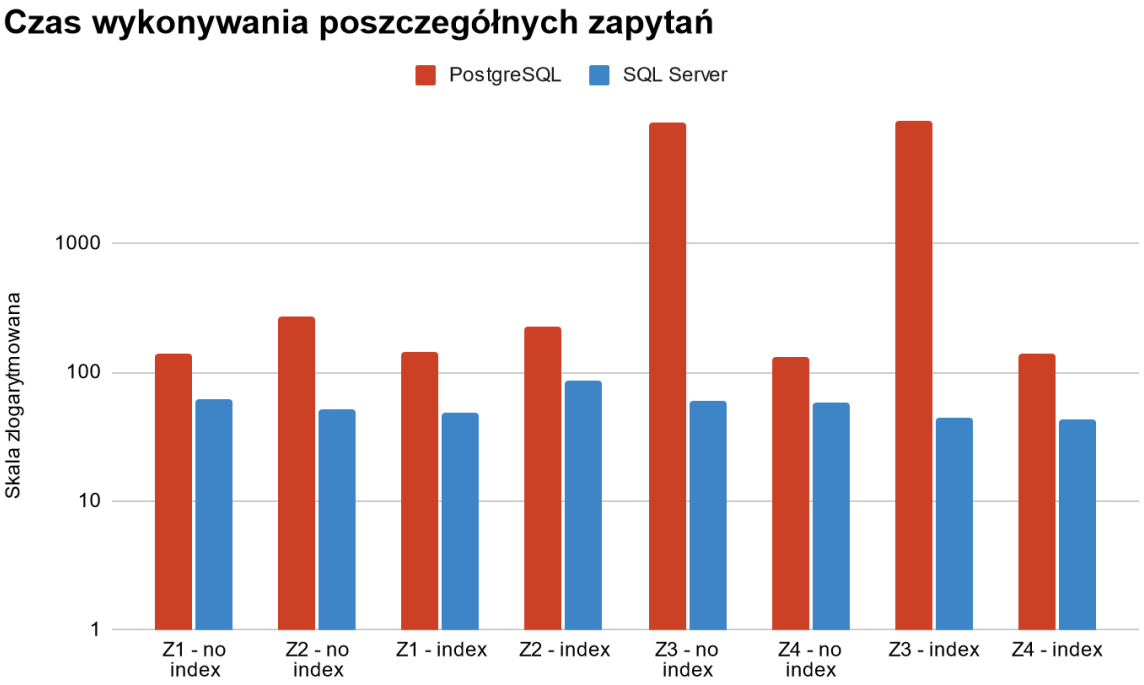
Poniżej przedstawiono zestawienie wyników testów dla obu środowisk bazodanowych oraz wykresy, Wyk.4. posiada skalę zlogarytmowaną, aby wyniki były bardziej czytelne.

Czas wykonywania zapytań [ms]								
	Zapytanie 1		Zapytanie 2		Zapytanie 3		Zapytanie 4	
BEZ INDEKSÓW	Min	AVG	Min	AVG	Min	AVG	Min	AVG
PostgreSQL	129	138	257	262	8580	8598	126	128
SQL Server	59	65	49	52	53	60	52	59
Z INDEKSAMI								
PostgreSQL	137	142	220	228	8785	8846	135	140
SQL Server	43	49	82	85	43	44	38	43

Tab.5. Zestawienie czasów wykonywania się zapytań dla obu SZBD.



Wyk.3. Złączone czasy wykonywania się zapytań



Wyk.4. Wyniki testów w ujęciu w celu normalizacji

6. Wnioski

Po analizie otrzymanych wyników można stwierdzić, że:

1) **Postać zdenormalizowana jest mimo wszystko w większości przypadków wydajniejsza.**
   Zapytanie 1 (patrz Wyk.4.) wykonuję się szybciej niż zapytanie 2.

2) Patrząc jednak na zapytanie 3 i 4 widzimy, że tam szybciej wykonuje się to drugie wspomniane. Wynika to z tego, że w zapytaniu 3 złączenie jest wykonywane poprzez zagnieżdżenie skorelowane. Przykład ten pozwala dojść do kolejnego wniosku jakim jest to, że **zagnieżdżenia skorelowane są znacznie wolniejsze w postaci zdenormalizowanej niż w znormalizowanej.**

Dodatkowe wnioski:

3) Przyglądając się wykresowi 1, przedstawiającemu średni czas wykonywania zapytań w PostgreSQL widzimy, że zastosowane indeksy nie poprawiły wydajności. Jedynie dla postaci znormalizowanej, gdzie użyto złączeń tabel czas po indeksacji jest szybszy. Można wnioskować, że korzystanie z indeksów w systemie bazodanowym PostgreSQL, w tym przypadku nie było konieczne.

4) Analizując tabele 5, gdzie zestawione zostały wyniku testów wykonanych w obu SZBD widać, że Microsoft SQL Server lepiej poradził sobie z zapytaniami. Czas ich wykonywań jest szybszy.

7. Podsumowanie

Finalnie jednak w większości przypadków zapytania z tabelą w postaci zdenormalizowanej miały lepszą wydajność. Normalizacja, poza małymi wyjątkami powodowała wydłużenie się czasu. Nie należy jednak z niej rezygnować, bo wciąż ma wiele zalet, które w pewnych warunkach mogą pomóc nam zoptymalizować nam zapytania. Posiada strukturę łatwiejszą do modyfikacji oraz szybciej ładuje dane. W przypadku operowania na konkretnych informacjach (podziałach tematycznych), praca na znormalizowanych tabelach może okazać się szybsza.

8. Bibliografia

1. Mgr inż. Łukasz Jajeśnica, prof. nadzw. dr hab.inż. Adam Piórkowski – Wydajność złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych
2. dr inż. Michał Lupa – Bazy danych 2022
3. https://pl.wikipedia.org/wiki/Posta%C4%87_normalna_(bazy_danych)
4. https://pg.edu.pl/documents/1403427/1bacfc1e-012d-4800-b6b2-4f0c4cce3c99
5. http://stareaneksy.pwn.pl/historia_ziemi/przyklady/?pokaz=tabela
6. https://math.uwb.edu.pl/~mariusz/share/classes/hd/hd.pdf
7. https://www.plukasiewicz.net/SQL