Autor: Krystian Woźniak, nr indeksu: 304361
Data: 12.06.2024r.

Sprawozdanie z projektu zaliczeniowego

Wydajność złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych

Spis treści

1. Wstęp

Niniejsze sprawozdanie dotyczy przeprowadzonych badań i eksperymentów mających na celu ocenę czasu wykonania zapytań pod kątem wpływu normalizacji na wydajność bazy danych. Podstawą teoretyczną do przeprowadzenia projektu był artykuł naukowy autorstwa Łukasza Jajeśnicy i Adama Piórkowskiego, opublikowany w „Studia Informatica" w 2010 roku. Artykuł ten porusza aspekty wydajnościowe związane z normalizacją schematów baz danych, prezentując koncepcję tworzenia dużych baz danych i hurtowni w formach znormalizowanych (schemat płatka śniegu) i zdenormalizowanych (schemat gwiazdy). Zbadano również wpływ zapytań bazujących na złączeniach i zagnieżdżeniach skorelowanych. Eksperymenty przeprowadzono z użyciem systemów zarządzania bazami danych Microsoft SQL server i PostgreSQL.

Normalizacja bazy danych to proces mający na celu eliminację powtarzających się danych w relacyjnej bazie danych. Istnieją sposoby ustalenia, czy dany schemat bazy danych jest znormalizowany i jeżeli jest – to w jakim stopniu. Jednym ze sposobów jest transformowanie danej bazy do schematów zwanych postaciami normalnymi (ang. *normal forms*, NF). Normalizacja bazy danych do konkretnej postaci może wymagać rozbicia dużych tabel na mniejsze i przy każdym wykonywaniu zapytania do bazy danych ponownego ich łączenia.

Złączenia tabel są zazwyczaj efektywnie wykonywane przez systemy zarządzania bazami danych, natomiast zapytania zagnieżdżone, szczególnie skorelowane, są zazwyczaj bardzo trudne w optymalizacji, co jest związane z faktem, iż dla każdej krotki z tabeli w zapytaniu zewnętrznym musi być każdorazowo wykonane zapytanie wewnętrzne. Taka sytuacja nie występuje w przypadku zapytań zagnieżdżonych nieskorelowanych.

2. Tabela Geochronologiczna

Na potrzeby eksperymentu stworzono bazę danych na podstawie tabeli Geochronologicznej, która obrazuje przebieg historii Ziemi na podstawie procesów i warstw skalnych. W tabeli 1 przedstawiono taksonomię dla czterech jednostek geochronologicznych – eonu, ery, okresu i epoki. Wymiar ostatniej jednostki – piętra został pominięty ze względu na obszerność (71 elementów). W różnych opracowaniach można spotkać różne podziały stratygraficzne z powodu ciągle trwających badań i typowania nowych profili wzorcowych (stratotypów).

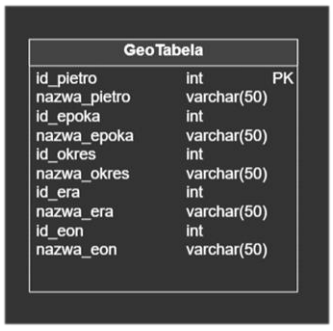
Tabela geochronologiczna

Wiek (mln lat)	Eon	Era	Okres		Epoka
0,010	FANEROZOIK	Kenzoik	Czwar-torząd		Halocen
1,8					Plejstocen
22,5			Trzeciorząd	Neo-gen	Pliocen
					Miocen
65				Paleogen	Oligocen
					Eocen
					Paleocen
140		Mezozoik	Kreda		Górna
					Dolna
195			Jura		Górna
					Środkowa
					Dolna
230			Trias		Górna
					Środkowa
					Dolna
280		Paleozoik	Perm		Górny
					Dolny
345			Karbon		Górny
					Dolny
395			Dewon		Górny
					Środkowy
					Dolny

Grafika 1. Tabela Geochronologiczna

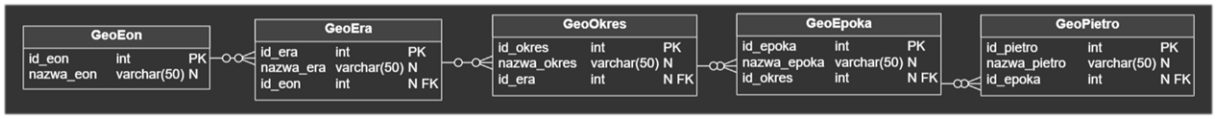
3. Konstrukcja wymiaru Geochronologicznego

Baza danych w postaci zdenormalizowanej przybrała postać schematu Gwiazdy. Jest to najprostszy model projektu bazy danych. Główną cechą schematu gwiazdy jest centralna tabela, nazywana tabelą faktów, z którą połączone są tabele wymiarów. Model taki umożliwia przeglądanie poszczególnych kategorii, operacje agregacji, czy też zaawansowane drążenie i filtrowanie danych.



Grafika 2. Schemat tabeli zdenormalizowanej (Źródło: Vertabelo)

Baza danych w postaci znormalizowanej przybrała postać schematu płatka śniegu. Schemat płatka śniegu to bardziej złożona wersja schematu gwiazdy, gdzie tabele wymiarów są znormalizowane zgodnie z modelem relacyjnej bazy danych. Jest on stosowany, gdy tabele wymiarów są duże i wymagają bardziej szczegółowego przedstawienia danych.



Grafika 3. Schemat tabeli znormalizowanej (Źródło: Vertabelo)

4. Omówienie kodu

Poniżej przedstawiony został kod źródłowy który posłużył do wykonania operacji związanych z tworzeniem poszczególnych tabel schematu znormalizowanego (Grafika 3.) w programie Microsoft SQL server. Analogiczny kod został zaimplementowany w systemie PostgreSQL.

```
CREATE SCHEMA projekt;
GO

CREATE TABLE projekt.GeoEon (
    id_eon INT PRIMARY KEY,
    nazwa_eon VARCHAR(50)
);

CREATE TABLE projekt.GeoEra (
    id_era INT PRIMARY KEY,
    nazwa_era VARCHAR(50),
    id_eon INT,
    FOREIGN KEY (id_eon) REFERENCES projekt.GeoEon(id_eon)
);

CREATE TABLE projekt.GeoOkres (
    id_okres INT PRIMARY KEY,
    nazwa_okres VARCHAR(50),
    id_era INT,
    FOREIGN KEY (id_era) REFERENCES projekt.GeoEra(id_era)
);

CREATE TABLE projekt.GeoEpoka (
    id_epoka INT PRIMARY KEY,
    nazwa_epoka VARCHAR(50),
    id_okres INT,
    FOREIGN KEY (id_okres) REFERENCES projekt.GeoOkres(id_okres)
);

CREATE TABLE projekt.GeoPietro (
    id_pietro INT PRIMARY KEY,
    nazwa_pietro VARCHAR(50),
    id_epoka INT,
    FOREIGN KEY (id_epoka) REFERENCES projekt.GeoEpoka(id_epoka)
);
```

Grafika 4. Kod źródłowy tworzenie tabel do schematu znormalizowanego

Do stworzenia tabel schematu znormalizowanego użyto polecenia CREATE TABLE oraz ustawiono klucze główne PRIMARY KEY na odpowiednich kolumnach. Dodano także klucze obce FOREGIN KEY które tworzą relacje między tabelami schematu znormalizowanego.

Formę zdenormalizowaną tabeli geochronologicznej osiągnięto tworząc jedną tabelę GeoTabela (Grafika 2.), zawierającą wszystkie dane z powyższych tabel. Dokonano tego za pomocą złączenia wewnętrznego, obejmującego wszystkie tabele tworzące hierarchię.

```
SELECT
    GeoPietro.id_pietro,
    GeoEon.nazwa_eon,
         GeoEra.nazwa_era,
         GeoOkres.nazwa_okres,
         GeoEpoka.nazwa_epoka,
         GeoPietro.nazwa_pietro
INTO projekt.GeoTabela
FROM projekt.GeoPietro
INNER JOIN projekt.GeoEpoka ON GeoPietro.id_epoka = GeoEpoka.id_epoka
INNER JOIN projekt.GeoOkres ON GeoEpoka.id_okres = GeoOkres.id_okres
INNER JOIN projekt.GeoEra ON GeoOkres.id_era = GeoEra.id_era
INNER JOIN projekt.GeoEon ON GeoEra.id_eon = GeoEon.id_eon;

ALTER TABLE projekt.GeoTabela
ADD CONSTRAINT PK_GeoTabela PRIMARY KEY (id_pietro);
```

Grafika 5. Kod źródłowy do tabeli GeoTabela (schemat zdenormalizowany)

Za pomocą polecenia ALTER TABLE dodano klucz główny do kolumny id_pietro. Analogiczny kod został zaimplementowany w systemie PostgreSQL gdzie użyto złączenia naturalnego NATURAL JOIN zamiast złączenia wewnętrznego INNER JOIN.

Poniżej znajduje się widok prezentujący rezultat w postaci pierwszych rekordów tabeli GeoTabela. Łączna liczba rekordów wynosi 71.

Results Messages

	id_pietro	nazwa_eon	nazwa_era	nazwa_okres	nazwa_epoka	nazwa_pietro
1	1	Fanerozoik	Kenozoik	Czwartorzęd	Holocen	NULL
2	2	Fanerozoik	Kenozoik	Czwartorzęd	Plejstocen	NULL
3	3	Fanerozoik	Kenozoik	Neogen	Pliocen	Gelas
4	4	Fanerozoik	Kenozoik	Neogen	Pliocen	Piacent
5	5	Fanerozoik	Kenozoik	Neogen	Pliocen	Zankl
6	6	Fanerozoik	Kenozoik	Neogen	Miocen	Mesyn
7	7	Fanerozoik	Kenozoik	Neogen	Miocen	Torton
8	8	Fanerozoik	Kenozoik	Neogen	Miocen	Serrawall
9	9	Fanerozoik	Kenozoik	Neogen	Miocen	Lang
10	10	Fanerozoik	Kenozoik	Neogen	Miocen	Burdigal
11	11	Fanerozoik	Kenozoik	Neogen	Miocen	Akwitan
12	12	Fanerozoik	Kenozoik	Paleogen	Oligocen	Szat
13	13	Fanerozoik	Kenozoik	Paleogen	Oligocen	Rupel
14	14	Fanerozoik	Kenozoik	Paleogen	Eocen	Priabon
15	15	Fanerozoik	Kenozoik	Paleogen	Eocen	Barton
16	16	Fanerozoik	Kenozoik	Paleogen	Eocen	Lutet
17	17	Fanerozoik	Kenozoik	Paleogen	Eocen	Iprez
18	18	Fanerozoik	Kenozoik	Paleogen	Paleocen	Thanet
19	19	Fanerozoik	Kenozoik	Paleogen	Paleocen	Zeland
20	20	Fanerozoik	Kenozoik	Paleogen	Paleocen	Dan
21	21	Fanerozoik	Mezozoik	Kreda	Górna	Mastrycht
22	22	Fanerozoik	Mezozoik	Kreda	Górna	Kampan
23	23	Fanerozoik	Mezozoik	Kreda	Górna	Santon
24	24	Fanerozoik	Mezozoik	Kreda	Górna	Koniak
25	25	Fanerozoik	Mezozoik	Kreda	Górna	Turon
26	26	Fanerozoik	Mezozoik	Kreda	Górna	Cenoman
27	27	Fanerozoik	Mezozoik	Kreda	Dolna	Alb

Grafika 6. Widok pierwszych rekordów tabeli GeoTabela

5. Testy wydajności

Testy polegały na porównaniu czasu wykonywania czterech różnych zapytań bazujących na złączeniach i zagnieżdżeniach zarówno dla bazy znormalizowanej i zdenormalizowanej. Wszystkie Testy wykonano na tej samej maszynie. Przetestowano dwa najpopularniejsze rozwiązania bazodanowe:

- Microsoft SQL Server 2019
- PostgreSQL z programem zarządzającym Dbeaver

W zapytaniach testowych łączono dane z tabeli geochronologicznej z syntetycznymi danymi o rozkładzie jednostajnym z tabeli *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. Poniżej znajduje się kod źródłowy który posłużył do stworzenia tabeli Milion.

```
CREATE TABLE projekt.Dziesiec (
    cyfra INT PRIMARY KEY
);

-- Wstawianie danych do tabeli Dziesiec
INSERT INTO projekt.Dziesiec (cyfra) VALUES
(0), (1), (2), (3), (4), (5), (6), (7), (8), (9);

CREATE TABLE projekt.Milion (
    liczba INT NOT NULL PRIMARY KEY,
        cyfra INT
);

-- Wypełnianie tabeli Milion liczbami od 0 do 999 999 za pomocą autozłączenia tabeli Dziesiec
INSERT INTO projekt.Milion SELECT a1.cyfra +10* a2.cyfra + 100*a3.cyfra + 1000*a4.cyfra + 10000*a5.cyfra +
100000*a6.cyfra AS liczba,
a1.cyfra AS cyfra
FROM projekt.Dziesiec a1, projekt.Dziesiec a2, projekt.Dziesiec a3, projekt.Dziesiec a4, projekt.Dziesiec
a5, projekt.Dziesiec a6
ORDER BY liczba ASC;
```

Grafika 7. Kod źródłowy - Tworzenie tabeli Milion

5.1. Konfiguracja sprzętowa i programowa

Wszystkie testy zostały wykonane na tej samym komputerze o następujących parametrach:

- CPU: AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz 8 rdzeni, 16 wątków
- RAM: 16 GB 4266 MHz LPDDR4X
- SSD: Intel 670p 512GB PCI-E x4 Gen3 NVMe 256 MB cache
- S.O.: Windows 11 Home 23H2

Jako systemy zarządzania bazami danych wybrano oprogramowanie wolno dostępne:

- Microsoft SQL Server Management Studio 20 – wersja developerska
- PostgreSQL 16 z programem DBeaver Community 24.1

Każdy test wykonywany był dziesięciokrotnie dla każdego systemu zarządzania bazą danych.

5.2. Kryteria Testów

W eksperymencie przeprowadzono szereg zapytań sprawdzających wydajność złączeń i zagnieżdżeń z tabelą geochronologiczną w wersji zdenormalizowanej i znormalizowanej. Procedurę testową przeprowadzono w dwóch etapach:

- Pierwszy etap polegał na wykonywaniu zapytań bez nałożonych indeksów na kolumny danych. Jedynymi indeksowanymi danymi były dane w kolumnach będących kluczami głównymi poszczególnych tabel (indeksy klastrowane)
- W drugim etapie nałożono indeksy nieklastrowane na wszystkie kolumny biorące udział w zapytaniu

Poniżej znajduje się kod źródłowy, który posłużył do wykonania indeksacji w drugim etapie:

```
-- Indeksy dla tabeli Milion
CREATE NONCLUSTERED INDEX idx_Milion_liczba ON projekt.Milion (liczba);

-- Indeksy dla tabeli GeoPietro
CREATE NONCLUSTERED INDEX idx_GeoPietro_id_pietro ON projekt.GeoPietro (id_pietro);
CREATE NONCLUSTERED INDEX idx_GeoPietro_id_epoka ON projekt.GeoPietro (id_epoka);

-- Indeksy dla tabeli GeoEpoka
CREATE NONCLUSTERED INDEX idx_GeoEpoka_id_epoka ON projekt.GeoEpoka (id_epoka);
CREATE NONCLUSTERED INDEX idx_GeoEpoka_id_okres ON projekt.GeoEpoka (id_okres);

-- Indeksy dla tabeli GeoOkres
CREATE NONCLUSTERED INDEX idx_GeoOkres_id_okres ON projekt.GeoOkres (id_okres);
CREATE NONCLUSTERED INDEX idx_GeoOkres_id_era ON projekt.GeoOkres (id_era);

-- Indeksy dla tabeli GeoEra
CREATE NONCLUSTERED INDEX idx_GeoEra_id_era ON projekt.GeoEra (id_era);
CREATE NONCLUSTERED INDEX idx_GeoEra_id_eon ON projekt.GeoEra (id_eon);

-- Indeksy dla tabeli GeoEon
CREATE NONCLUSTERED INDEX idx_GeoEon_id_eon ON projekt.GeoEon (id_eon);

-- Indeksy dla tabeli GeoTabela
CREATE NONCLUSTERED INDEX idx_GeoTabela_id_pietro ON projekt.GeoTabela (id_pietro);
```

Grafika 8. Wykonanie indeksacji w celu przeprowadzenia testów w drugim etapie

Celem testów była ocena wpływu normalizacji na zapytania złożone (złączenia i zagnieżdżenia). W tym celu zaproponowano cztery zapytania:

- Zapytanie 1 (1ZL), którego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci znormalizowanej, reprezentowaną przez złączenia pięciu tabel:

```
set statistics time on
SELECT COUNT(*) FROM projekt.Milion M
INNER JOIN projekt.GeoTabela ON (M.liczba % 71) = (projekt.GeoTabela.id_pietro);
set statistics time off
```

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

```
SELECT COUNT(*)
FROM projekt.Milion M
INNER JOIN projekt.GeoPietro ON (M.liczba % 71) = GeoPietro.id_pietro
INNER JOIN projekt.GeoEpoka ON GeoPietro.id_epoka = GeoEpoka.id_epoka
INNER JOIN projekt.GeoOkres ON GeoEpoka.id_okres = GeoOkres.id_okres
INNER JOIN projekt.GeoEra ON GeoOkres.id_era = GeoEra.id_era
INNER JOIN projekt.GeoEon ON GeoEra.id_eon = GeoEon.id_eon;
```

- 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 projekt.Milion M
WHERE (M.liczba % 71) IN (
    SELECT id_pietro
    FROM projekt.GeoPietro
    INNER JOIN projekt.GeoEpoka ON GeoPietro.id_epoka = GeoEpoka.id_epoka
    INNER JOIN projekt.GeoOkres ON GeoEpoka.id_okres = GeoOkres.id_okres
    INNER JOIN projekt.GeoEra ON GeoOkres.id_era = GeoEra.id_era
    INNER JOIN projekt.GeoEon ON GeoEra.id_eon = GeoEon.id_eon);
```

- 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 ta-bel poszczególnych jednostek geochronologicznych:

```
SELECT COUNT(*)
FROM projekt.Milion M
WHERE (M.liczba % 71) IN (
    SELECT id_pietro
    FROM projekt.GeoPietro
    INNER JOIN projekt.GeoEpoka ON GeoPietro.id_epoka = GeoEpoka.id_epoka
    INNER JOIN projekt.GeoOkres ON GeoEpoka.id_okres = GeoOkres.id_okres
    INNER JOIN projekt.GeoEra ON GeoOkres.id_era = GeoEra.id_era
    INNER JOIN projekt.GeoEon ON GeoEra.id_eon = GeoEon.id_eon);
```

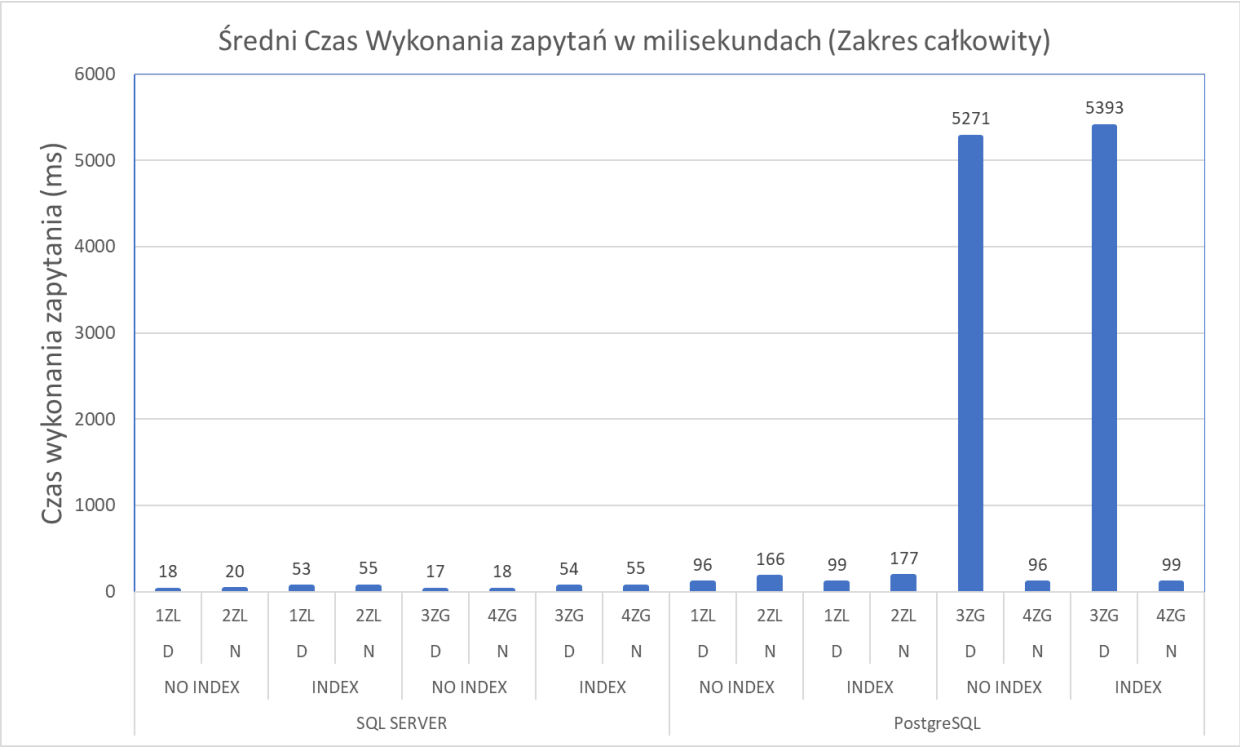
6. Wyniki Testów

Każdy test został powtórzony dziesięciokrotnie, a wyniki skrajne pominięto. W przypadku systemu Microsoft SQL server odnotowano większą zgodność kolejnych prób. Każde zapytanie zostało wykonane co najmniej kilkukrotnie szybciej w porównaniu do systemu PostgreSQL. W szczególności zapytanie 3ZL w systemie PostgreSQL wykonywało się zdecydowanie dłużej niż w przypadku Microsoft SQL server. Wyniki testów przedstawiono w poniższej tabeli:

Tabela 1. Czas wykonania zapytań w milisekundach (ms)

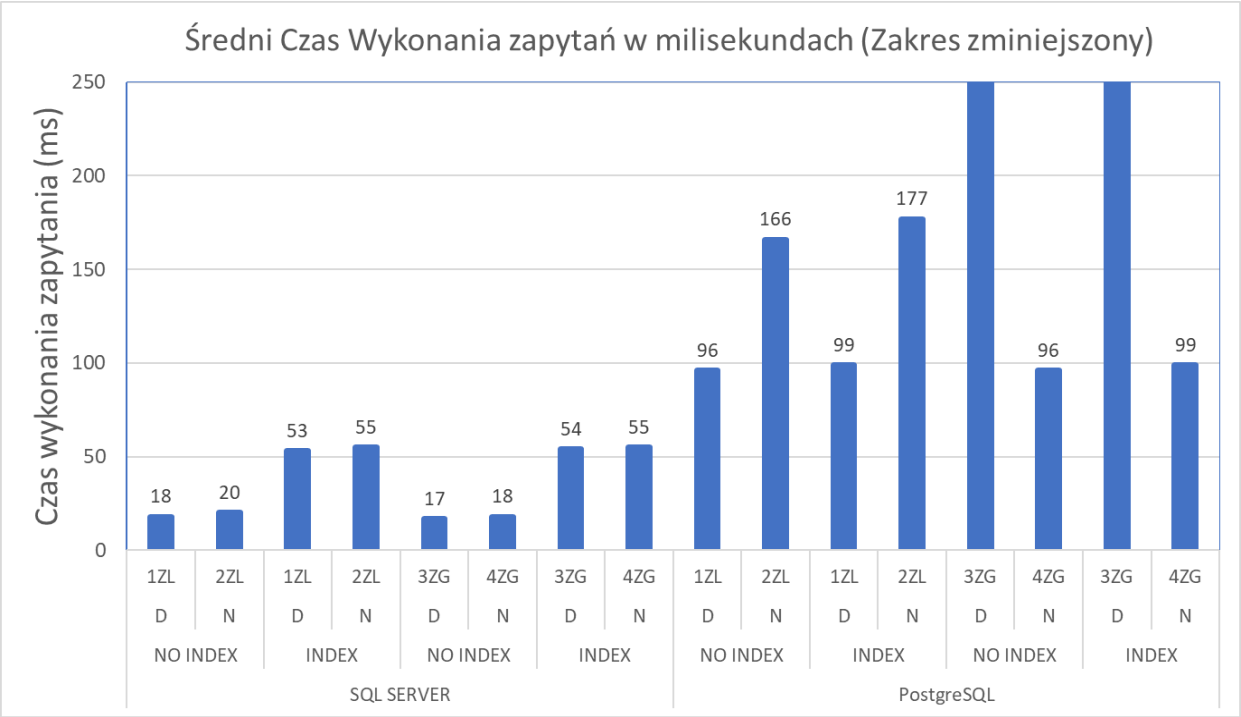
	1 ZL		2ZL		3ZG		4ZG	
BEZ INDEKSÓW	**MIN**	**ŚR**	**MIN**	**ŚR**	**MIN**	**ŚR**	**MIN**	**ŚR**
SQL server	16	18	18	20	16	17	16	18
PostgreSQL	92	96	163	166	5211	5271	94	96
Z INDEKSAMI								
SQL server	52	53	54	55	52	54	53	55
PostgreSQL	96	99	165	177	5289	5393	94	99

Aby ułatwić analizę otrzymanych wyników załączono wykres 1 prezentujący średni czas wykonania zapytań. Wyniki zestawiono wysuwając na pierwszy plan związki z normalizacją – czy wersja znormalizowana (N) jest wolniejsza czy szybsza od wersji zdenormalizowanej (D).



Wykres 1. Wyniki eksperymentu (zakres całkowity)

Ponieważ wyniki zapytania 3ZG dla Systemu PostgreSQL są wyraźnie odstające i uniemożliwiają poprawną interpretację danych sporządzono również wykres 2 z częściowym (zmniejszonym zakresem z 6000 ms do 250ms).



Wykres 2. Wyniki eksperymentu (zakres częściowy)

7. Wnioski

Otrzymane Wyniki pozwalają wyciągnąć następujące wnioski:

- Wyniki różnią się znacznie w zależności od zastosowanego systemu zarządzającego bazą danych.
- W systemie SQL server praktycznie nie ma różnicy w wydajności pomiędzy schematem znormalizowanym a zdenormalizowanym. Jest delikatna różnica w szybkości zapytań na korzyść schematu zdenormalizowanego natomiast mieści się ona w granicach błędu pomiarowego.
- W systemie SQL server widać znaczący wpływ indeksacji na szybkość wykonywania zapytań. Zapytania dla tabel posiadających indeksy nieklastrowane są wykonywane znacznie wolniej.

- W systemie SQL server nie odnotowano różnic w czasie wykonania zapytań dla zapytań bazujących na złączeniach w porównaniu do zapytań bazujących na zagnieżdżeniach skorelowanych.
- W systemie PostgreSQL normalizacja ma znaczący wpływ na czas wykonywania zapytań. Zapytania bazujące na złączeniach naturalnych zdecydowanie szybciej wykonują się dla schematu znormalizowanego. Zupełnie inaczej jest w przypadku zapytań bazujących na zagnieżdżeniach skorelowanych natomiast w tym przypadku największy wpływ na to miała sama konstrukcja zapytania 3ZG gdzie Warunek WHERE sprawdzany jest również dla zapytania wewnętrznego co znacznie wydłużyło czas wykonania zapytania w porównaniu do zapytania 4ZG gdzie zapytanie wewnętrzne nie zawiera dodatkowego warunku WHERE.
- Co ciekawe w Systemie PostgreSQL indeksacja praktycznie nie wpłynęła na szybkość wykonywania zapytań, a ewentualne różnice są minimalne.

Podsumowując, można jednoznacznie stwierdzić iż system SQL server jest zdecydowanie lepiej zoptymalizowanym rozwiązaniem. W systemie tym znaczny wpływ na wydajność baz danych ma indeksacja. Wyniki uzyskane w systemie PostgreSQL lepiej odzwierciedlają tezę iż schematy zdenormalizowane są z reguły wydajniejsze natomiast warto pamiętać o zaletach schematu znormalizowanego, a mianowicie łatwą konserwację, porządek oraz lepszą funkcjonalność bazy.