WYDAJNOŚĆ ZŁĄCZEŃ I ZAGNIEŻDŻEŃ DLA SCHEMATÓW ZNORMALIZOWANYCH I ZDENORMALIZOWANYCH

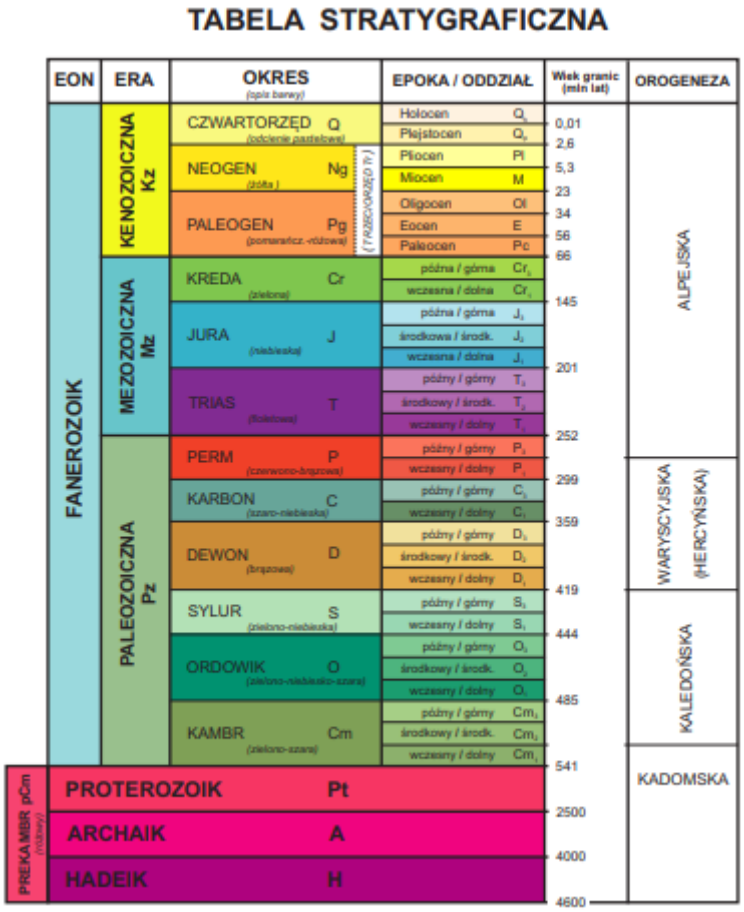
1.Wprowadzenie

Celem projektu jest sprawdzenie wydajności zapytań zindeksowanych lub niezindeksowanych dla schematów znormalizowanych i nieznormalizowanych. Realizacja przebiegała w oparciu o rozwiązania bazodanowe SQL Server i PostgreSQL.

2. Tabela geochronologiczna

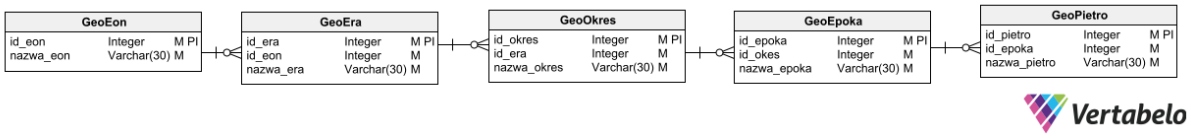
Tabela geochronologiczna obrazuje przebieg historii Ziemi, która zawiera jednostki geochronologiczne mające wymiar czasowy takie jak eon, era, okres, epoka, wiek oraz odpowiadające im jednostki stratygraficznie. Obecnie przyjęta tabela geochronologiczna została ustalona przez Międzynarodową Komisję Stratygrafii.

Tabela 1: Tabela Geochronologiczna



W tabeli 1 przedstawiono taksonomię dla czterech jednostek geochronologicznych – eonu, ery, okresu i epoki.

Rys. 1: Znormalizowana tabela geochronologiczna



3. Konstrukcja wymiaru Geochronologicznego

Rys. 2: Tabela Geochronologiczna „GeoTabela” w postaci zdenormalizowanej (schemat gwiazdy)

Na podstawie tabel z Rys. 1, powstało złączenie naturalne (Natural Join), które pozwoliło na stworzenie tabeli zdenormalizowanej – schemat gwiazdy (Rys. 2).

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

Utworzenie tabeli GeoTabela umożliwiło szybki dostęp do wszystkich danych tabeli geochronologicznej przy użyciu jednego zapytania prostego.

4. Testy Wydajności

W testach skupiono się na porównaniu wydajności złączeń oraz zapytań zagnieżdżonych.Wykorzystano rozwiązania bazodanowe takie jak:
- SQl Serwer,

- PostgreSQL.

Początkowo stworzono tabelę o nazwie „Dziesiec”, wypełnionej liczbami od 0 do 9. Tabela ta umożliwiła utworzenie kolejnej, bazującej na niej tabeli Milion.

CREATE TABLE Dziesiec(cyfra int, bit int);

INSERT INTO Dziesiec VALUES

(0, 0000000),

(1, 0000001),

(2, 0000010),

(3, 0000011),

(4, 0000100),

(5, 0000110),

(6, 0000111),

(7, 0001000),

(8, 0001001),

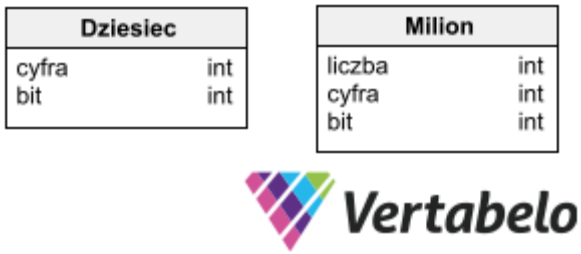
(9, 0000000);

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 auto złączenia tabeli *Dziesięć* wypełnionej liczbami od 0 do 9;

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

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

Rys. 3: Schemat tabel Dziesiec i Milion



4.1 KONFIGURACJA SPRZĘTOWA

CPU: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz

RAM: DDR4 8GB 2666 CL19 SO-DIMMSSD

SSD: WDC PC SN520 SDAPNUW-512G-1202

S.O.: Windows 10 Home.

Środowiska bazodanowe:
- SQL Server 2019
- PostgreSQL 14.3

4.2 KRYTERIA TESTÓW

W teście wykonano cztery różne zapytania sprawdzające wydajność złączeń i zagnieżdżeń z tabelą geochronologiczną w wersji zdenormalizowanej i znormalizowanej. Procedurę przeprowadzono w dwóch etapach:
-zapytania bez nałożonych indeksów na kolumny danych

-zapytania z nałożonymi indeksami na wszystkie kolumny biorące udział w złączeniu.

Zapytanie 1 (1 ZL), którego celem jest złączenie tableli milion z tabelą geochronologiczną w postaci zdemoralizowane, do warunku złączenia dodano operację modulo, dopasowującą zakresy wartości złączanych kolumn:
SELECT COUNT(*) FROM Milion INNER JOIN GeoTabela ON (mod(Milion.liczba,74)=(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 wyko-nywane poprzez zagnieżdżenie skorelowane, a zapytanie wewnętrzne jest złączeniem ta-bel 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;

5. WYNIKI TESTÓW

Każdy test przeprowadzono wielokrotnie (10 razy), wyniki skrajne pominięto. Wyniki testów dla PostgreSQL przedstawiono w tabeli 2 natomiast dla SQL Server w tabeli 3.

Tabela 2: SQL Server

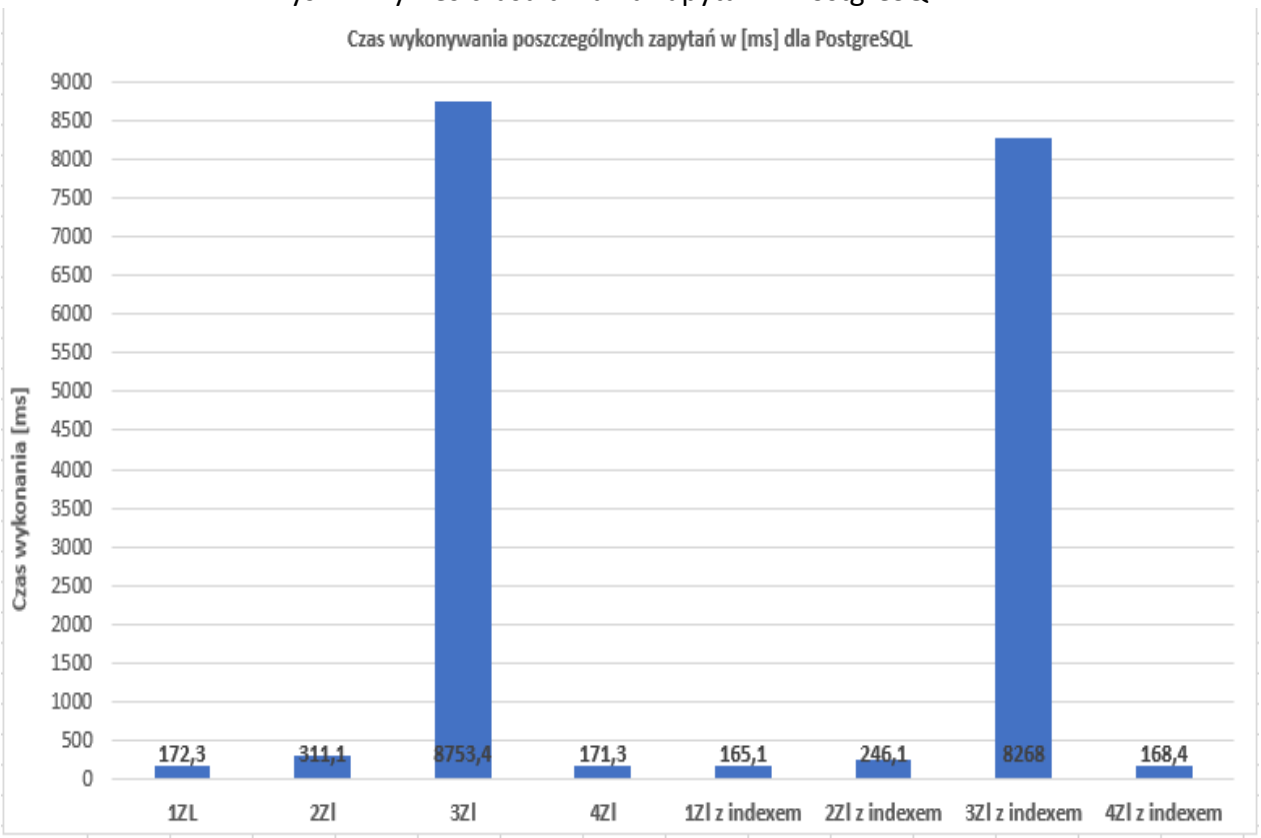
id pomiaru	1ZL	2Zl	3Zl	4Zl	1Zl z indexem	2Zl z indexem	3Zl z indexem	4Zl z indexem
1	164	302	8659	170	153	220	8540	170
2	184	317	8323	147	180	263	8318	171
3	155	307	8615	193	162	229	7880	144
4	156	295	8461	150	158	252	8243	162
5	165	338	8703	175	161	267	8290	173
6	173	303	8826	159	166	240	8197	182
7	150	367	8946	167	187	253	8229	181
8	194	319	8958	187	141	250	8221	168
9	210	292	9157	197	168	246	8281	155
10	172	271	8886	168	175	241	8481	178
min	150	271	8323	147	141	220	7880	144
max	210	367	9157	197	187	267	8540	182
avg	172,3	311,1	8753,4	171,3	165,1	246,1	8268	168,4

Tabela 3: PostgreSQL

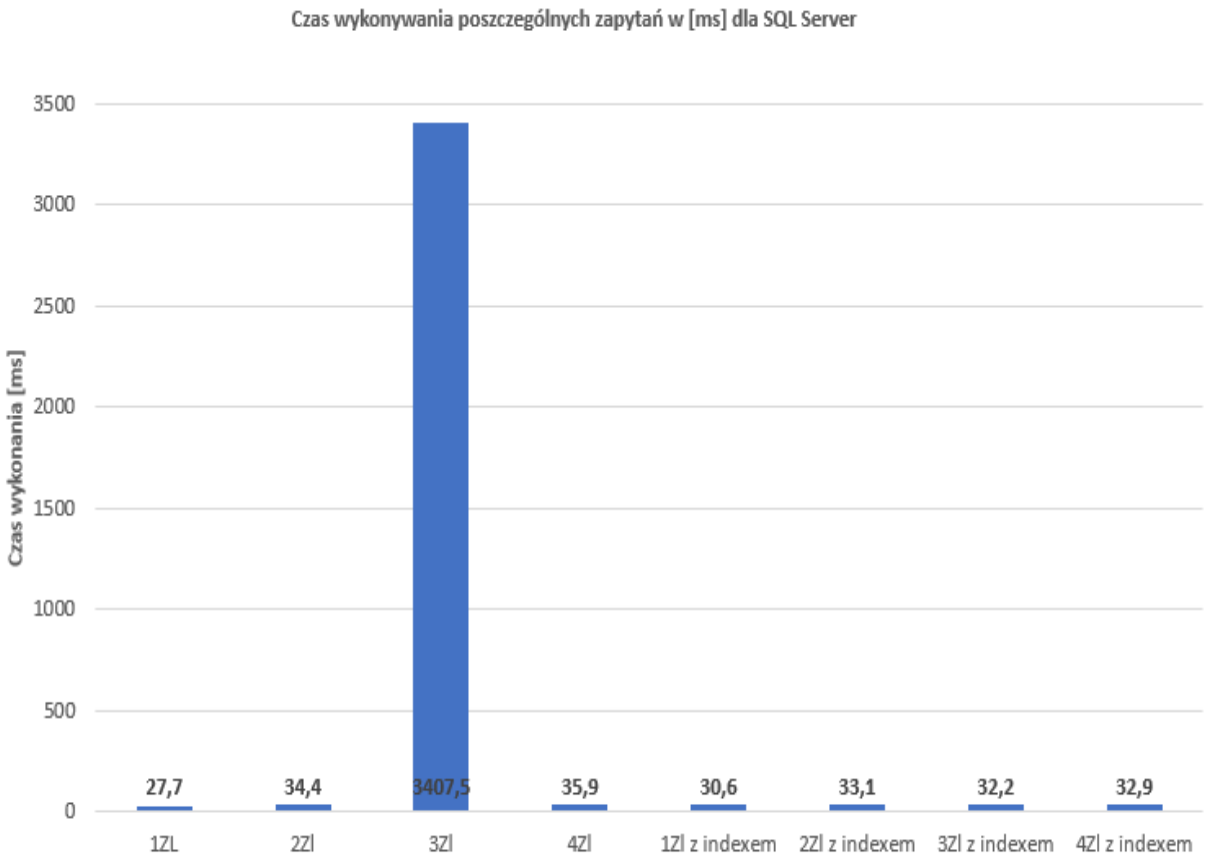
id	1ZL	2Zl	3Zl	4Zl	1Zl z indexem	2Zl z indexem	3Zl z indexem	4Zl z indexem
1	36	34	3299	36	33	34	32	37
2	28	33	3628	38	28	33	34	25
3	22	35	3908	34	30	35	33	36
4	23	33	3250	33	34	33	43	34
5	28	37	3289	37	32	43	28	37
6	33	34	3415	36	30	31	26	32
7	23	37	3268	33	31	33	32	31
8	27	33	3290	35	26	26	31	32
9	26	35	3429	41	28	32	33	33
10	31	33	3299	36	34	31	30	32
min	22	33	3250	33	26	26	26	25
max	36	37	3908	41	34	43	43	37
avg	27,7	34,4	3407,5	35,9	30,6	33,1	32,2	32,9

Analizę wyników ułatwiają wykresy (rys. 4 i 5) – ze względu na dość duże wartości pojedynczych przypadków użyto etykiet, aby były wartości które odzwierciedlają kolumny były lepiej widoczne.

Rys. 4: Wykres czasu trwania zapytań w PostgreSQL



Rys. 5: Wykres trwania zapytań w SQL Server



6. WNIOSKI

Po przeprowadzeniu analizy, można wyciągnąć następujące wnioski:
- W oprogramowaniu PostgreSQL, po użyciu indeksowania w każdym przypadku widać przyśpieszenie wykonania zapytania.
- W oprogramowaniu PostgreSQL, indeksowanie w przypadku zapytania 3ZL dało znacznie mniejsze przyśpieszenie niż w oprogramowaniu SQL Server
- Najwolniej wykonywanym zapytaniem przy użyciu PostgreSQL jak i SQL Server jest zapytanie 3Zl (zapytanie w postaci znormalizowanej z użyciem złączenia wykonywanego poprzez zagnieżdżenie skorelowane)
- Zapytanie 3Zl jest wykonywane znacznie szybciej w środowisku SQL Server niż PostgreSQL (różnica około 5s)
- Bardziej wydajnym systemem bazodanowym pod każdym względem okazał się SQL Server, którego wydajność jest nawet o kilkaset procent wyższa niż w przypadku PostgreSQL
- Postać zdenormalizowana powoduje wzrost wydajności natomiast znormalizowana prowadzi do jej spadku. Wprowadza ona jednak ład i przejrzystość, więc warto rozważyć jej plusy jak i minusy.

BIBLIOGRAFIA

1. www.vertabelo.com
2. Łukasz Jajeśnica, Adam Piórkowski - Wydajność złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych; Akademia Górniczo –Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej
3. https://readgur.com/doc/1389661/uproszczona-tabela-geochronologiczna
4. dr inż. Michał Lupa – 2022 Bazy Danych I