

Sprawozdanie

Porównanie wydajności złączy i zagnieżdżeń



AGH

AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Szymon Trojak

WGGIOŚ

Semestr 4

Cel sprawozdania

Celem analizy było porównanie wydajności kwerend bazujących na złączeniach i zagnieżdżeniach dla tabeli geologicznej.

Do wykonania zostały użyte dwa programy umożliwiające pracę na dużych bazach danych :

- PostgreSQL
- SQL Server
- MySQL

Konfiguracja sprzętowa

Komputer

- CPU: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz, 1190 MHz, Rdzenie: 4
- RAM: 16
- GPU: NVIDIA GeForce MX250
- System operacyjny: Microsoft Windows 11 Home

Programy

- PostgreSQL 15.3.1- Windows-x64
- SQL Server for Windows
- MySQL

Konstrukcja baz danych

Jako materiał do analizy posłużyła tabela geochronologiczna, która obrazuje przebieg historii Ziemi na podstawie następstwa procesów i warstw skalnych .Obecnie przyjęta tabela geochronologiczna została ustalona przez Międzynarodową Komisję Stratygrafii (ICS). W *tabeli 1* przedstawiono taksonomię dla pięciu jednostek geochronologicznych: eonu, ery, okresu, epoki wieku. Na podstawie części poniższych danych (68 rekordów) wykonano bazę danych znormalizowanych . Osobno dla SQL Server, PostgreSQL.

EONOTEM / EON	ERATEM / ERA	SYSTEM / OKRES		ODDZIAŁ / EPOKA	PIĘTRO / WIEK	MILIONY LAT
F A N E R O Z O I K	K E N O Z O I K	CZWARTORZĘD		HOLOCEN PLEJSTOCEN	GELAS PIACENT ZANKL MESYN TORTON SERRAWAL LANG BURDYGAŁ AKWITAN SZAT RUPEL PRIABON BARTON LUTET IPREZ TANET ZELAND	1,8
		TRZECIORZĘD	NEOGEN	PLIOCEN		
				MIOCEN		
				OLIGOCEN		23,5
			PALEOGEN	EOCEN		
				PALEOCEN		
					DAN MASTRYCHT KAMPAN SANTON KONIAK TURON CENOMAN ALB APT BARREM HOTERYW WALANŻYN BERIAS TYTON KIMERYD OKSFORD KELOWEJ BATON BAJOS AALEN TOARK PLIENSBACH SYNEMUR HETANG RETYK NORYK KARNIK LADYN ANIZYK OLENEK IND TATAR KAZAN UFA KUNGUR ARTINSK SAKMAR ASSEL	
	M E Z O Z O I K	KREDA	GÓRNA / POŹNA			65
			DOLNA / WCZESNA			
		JURA	GÓRNA / POŹNA			135
			ŚRODKOWA			
		DOLNA / WCZESNA				203
	P A L E O Z O I K	TRIAS	GÓRNY / POŹNY			250
			ŚRODKOWY			
		DOLNY / WCZESNY				295
		PERM	GÓRNY / POŹNY			355
			DOLNY / WCZESNY			
		KARBON	GÓRNY / POŹNY	STEFAN	GŻEL KASIMOW MOSKOW BASZKIR SERPUCHOW	410
				WESTFAL		
			DOLNY / WCZESNY	NAMUR		
		DEWON	GÓRNY / POŹNY	WIZEN TURNIEJ		435
			ŚRODKOWY			
		SYLUR	DOLNY / WCZESNY			500
			GÓRNY / POŹNY			
		ORDOWIK	ŚRODKOWY			543
			DOLNY / WCZESNY			
		KAMBR	GÓRNY / POŹNY			2500
			ŚRODKOWY			
		DOLNY / WCZESNY				
PREKAMBR	PROTE-ROZOIK	NEOPROTEROZOIK				
		MEZOPROTEROZOIK				
		PALEOPROTEROZOIK				
		NEOARCHAIK				
		MEZOARCHAIK				
	ARCHAIK	PALEOARCHAIK				
		EOARCHAIK				

Tabela 1 Tabela stratygraficzna

Zapytania

W teście wykonano szereg zapytań sprawdzających wydajność złączeń i zagnieżdżeń z tabelą geochronologiczną.

Zapytanie 1 (1 ZL), którego 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:

```
#1
SELECT COUNT(*) FROM geol.Milion INNER JOIN GeoTabela ON
(mod(Milion.liczba,77)=(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

```
#2
SELECT COUNT(*) FROM geol.Milion INNER JOIN geol.GeoPietro
ON
(mod(Milion.liczba,77)=GeoPietro.id_pietro)
NATURAL JOIN geol.GeoEpoka NATURAL JOIN geol.GeoOkres
NATURAL JOIN geol.GeoEra NATURAL JOIN geol.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:

```
#3
SELECT COUNT(*) FROM geol.Milion WHERE mod(Milion.liczba,77)=
(SELECT id_pietro FROM GeoTabela WHERE
mod(Milion.liczba,77)=(id_pietro));
```

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 tabel poszczególnych jednostek geochronologicznych:

```
#4
SELECT COUNT(*) FROM geol.Milion WHERE mod(Milion.liczba,77)
IN
(SELECT geol.GeoPietro.id_pietro FROM geol.GeoPietro
NATURAL JOIN
geol.GeoEpoka NATURAL JOIN geol.GeoOkres NATURAL JOIN
geol.GeoEra NATURAL JOIN geol.GeoEon);
```

Testy wydajności

W testach skupiono się na porównaniu wydajności złączeń oraz zapytań zagnieżdżonych, wykonywanych na tabelach o dużej liczbie danych. Testy wykonano w programie:

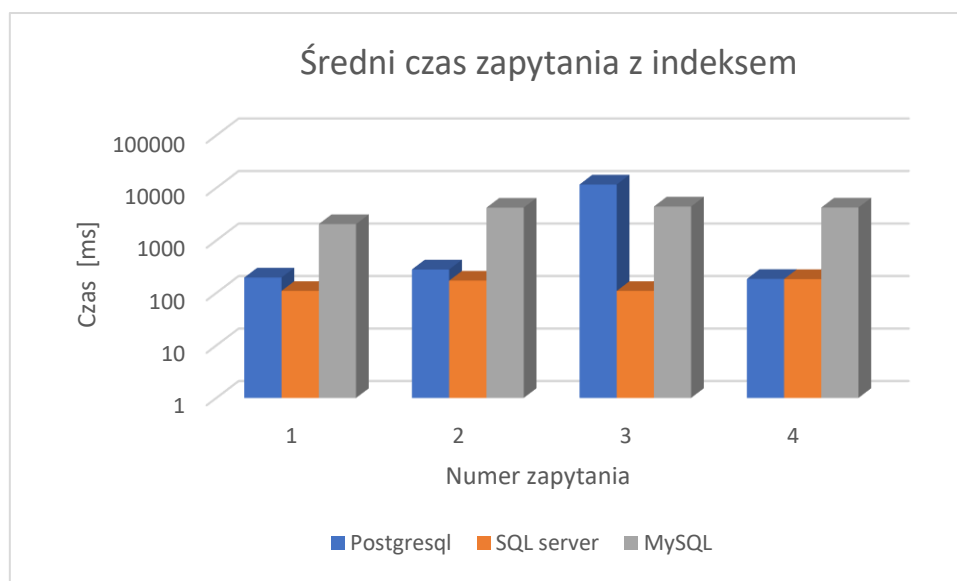
- PostgreSQL
- SQL Server for Windows
- MySQL

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. Aby otrzymać tabelę *Milion* wykonano dodatkową tabelę *Dziesięć* wypełnioną liczbami od 0 do 9, które po złączeniu i niewielkiej modyfikacji umożliwiły otrzymanie oczekiwanego wyniku- tabeli od 0 do 999 999.

Wyniki

Wykonano pomiary czasu dla zapytania z indeksem i bez indeksu. Dla każdego programu: PostgreSQL, SQL Server i MySQL wykonano ręcznie 10 razy po 4 zapytania, a następnie policzono średnią i przedstawiono wyniki w postaci histogramu w skali logarytmicznej. Każdy z poniższych wykresów obrazuje ilość czasu jaki potrzebował konkretny program na wykonanie zadanego mu zapytania.

Poniżej znajdują się tabele z średnimi dla danego zapytania .



Wykres 1 Średni czas zapytania dla wartości indeksowanych

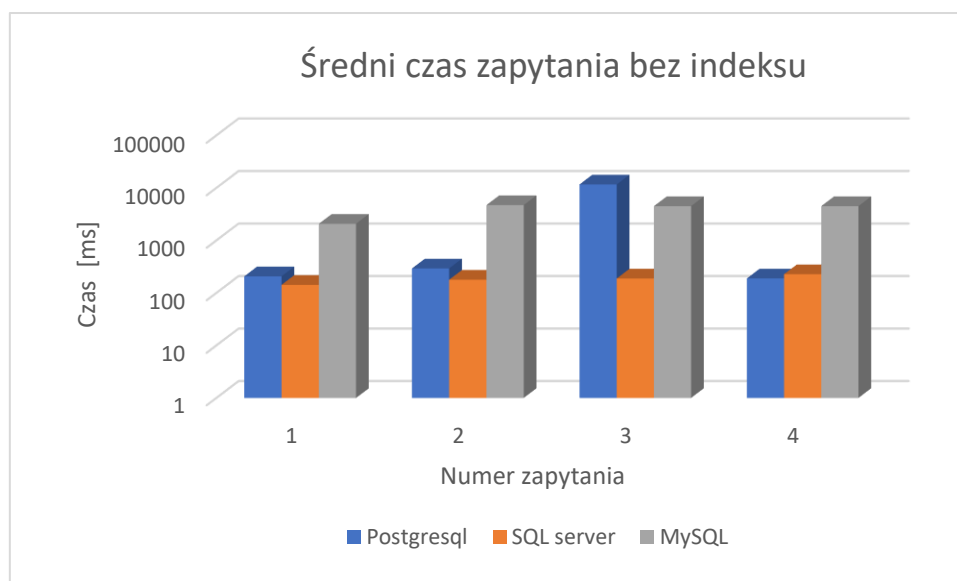
z indeksem

nr zapytania	1	2	3	4
Postgresql	196,4	279,9	11614,3	184,7
SQL server	109,4	171,8	109,2	183,7
MySQL	2043	4210	4424,4	4233,5

bez indeksu

nr zapytania	1	2	3	4
Postgresql	207,2	292,3	11629,4	188,8
SQL server	142,5	178,7	188,6	227,5
MySQL	2073,7	4700,667	4494	4493,667

Z powyższych tabel bardzo łatwo można wywnioskować, że zapytania z indeksem są szybsze od zapytań bez indeksu.



Wykres 2 Średni czas zapytania dla wartości nie indeksowanych

Podsumowanie

Wnioski jakie można wyciągnąć odczytując powyższe tabele i wykresy :

- Zapytania indeksowane są szybsze od zapytań nie indeksowanych, niezależnie od środowiska (programu) w którym zostały wykonane.
- SQL Server szybciej niż PostgreSQL przetwarza zadane mu zapytania niezależnie czy są indeksowane czy nie.
- MySQL najdłużej przetwarza dane.
- Postać zdenormalizowana w większości przypadków jest szybsza od postaci znormalizowanej

Podsumowując, mimo iż normalizacja jest mniej wydajna, pozwala ona na przejrzyste i zrozumiałe przechowywanie danych, przez co zmniejsza szanse na wystąpienie błędów oraz ułatwia zarządzanie takimi danymi.