SPRAWOZDANIE



Temat: Porównanie wydajności złączeń i zagnieżdżeń

Numer ćwiczenia: ćwiczenie 9

Autor: Filip Hałys

Wydział: WGGiOŚ

**1. Cel ćwiczenia**

Celem wykonanego ćwiczenia było zbadanie wydajności kwerend bazujących na złączeniach i zagnieżdżeniach dla tabeli geochronologicznej przedstawionej poniżej *(Tabela 1)*:

Obraz zawierający tekst, zrzut ekranu, numer, Równolegle

Opis wygenerowany automatycznie

*(Tabela 1)* Tabela Geochronologiczna

Obraz zawierający tekst, zrzut ekranu, Czcionka, biały

Opis wygenerowany automatycznie

*(Schemat 1)* Znormalizowany schemat tabeli geochronologicznej

Obraz zawierający tekst, Czcionka, zrzut ekranu, numer

Opis wygenerowany automatycznie

*(Schemat 2)* Zdenormalizowany schemat tabeli geochronologicznej

Ćwiczenie wykonano na systemie PostgreSQL i SQL Server Management Studio, a następnie dokonano porównania i analizy otrzymanych wyników.

**2. Konfiguracja sprzętowa**

* Procesor: AMD Ryzen 5 4500U with Radeon Graphics 2.38 GHz
* Pamięć RAM: 8 GB
* Typ system: 64-bitowy system operacyjny, procesor x64
* System operacyjny: Windows 11 Home
* PostgreSQL: wersja 15.3; 64 Bit
* SQL Server Management Studio: wersja 19.0.1

**3. Przeprowadzenie ćwiczenia**

Na początku utworzono w postgreSQL znormalizowany schemat tabeli geochronologicznej zgodnie z modelem przedstawionym na *(schemat 1).*

Obraz zawierający tekst, zrzut ekranu, Czcionka

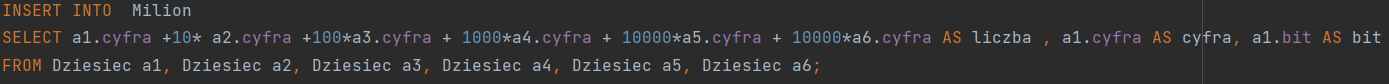
Opis wygenerowany automatycznie Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Wszystkie pięć tabeli uzupełniono danymi geochronologicznymi. Następnie utworzono tabele zdenormalizowaną, zgodnie z modelem przedstawionym na *(schemat 2).*



W kolejnym kroku utworzono tabelę „Milion”, zawierającą milion liczb naturalnych z zakresu od 0 do 999 999. Została ona utworzona za pomocą autozłączenia tabeli „Dziesiec” (jest to tabela wypełniona cyframi 0-9). Liczby wygenerowano kodem:



Ostatnim krokiem w tej części ćwiczenia było wygenerowanie czterech zapytań:

1. Zapytanie numer 1 (ZL) – celem tego zapytania było złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci zdenormalizowanej, przy czym do warunku złączenia dodano operację modulo, dopasowującą zakresy wartości złączanych kolumn:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

1. Zapytanie numer 2 (ZL) – celem tego zapytania było złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci zdenormalizowanej, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

1. Zapytanie numer 3 (ZG) – celem tego zapytania było złączenie syntetycznej tabeli miliona wyników z tabelą geochronologiczną w postaci zdenormalizowanej, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane:



1. Zapytanie numer 4 (ZG) – celem tego zapytania było złączenie syntetycznej tabeli miliona wyników z tabelą geochronologiczną w postaci znormalizowanej, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane, a zapytanie jest złączeniem tabel poszczególnych jednostek geochronologicznych:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Wszystkie kody wykonane w tej części (3. Przeprowadzenie ćwiczenia) zostały odwzorowane w SQL Server Management Studio. Kody w obu środowiskach różnią się nieznacznie (zauważone różnice to chociażby inne nazwy funkcji).

**4. Wyniki przeprowadzonych testów**

Każdy test uruchomiono łącznie 20 razy; pięciokrotnie w PostgreSQL bez indeksowania, pięciokrotnie w PostgreSQL z indeksowaniem, pięciokrotnie w SQL Server Management Studio bez indeksowania oraz pięciokrotnie w SQL Server Management Studio z indeksowaniem. Warto zaznaczyć, że pomiary czasowe znacznie różniące się od pozostałych zostały usunięte, a w ich miejsce zrobione nowe.

Tabela *(tabela 3)* przedstawia statystyki (wartość minimalną czasu i wartość średnią) obliczone na potrzeby wyciągnięcia wniosków. Czas przedstawiono w jednostce milisekund (ms).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 (ZL) |  | 2 (ZL) |  | 3 (ZG) |  | 4 (ZG) |  |
|  | MIN | AVG | MIN | AVG | MIN | AVG | MIN | AVG |
|  |  |  |  | BEZINDEKSOWANIA |  |  |  |  |
| PostgreSQL | 221 | 231 | 461 | 470 | 8635 | 8772 | 212 | 216 |
| SQL SMS | 16 | 21 | 15 | 20 | 12 | 18 | 15 | 17 |
|  |  |  |  | Z INDEKSOWANIEM |  |  |  |  |
| PostgreSQL | 271 | 280 | 527 | 535 | 8749 | 8856 | 236 | 242 |
| SQL SMS | 15 | 17 | 26 | 27 | 84 | 87 | 22 | 24 |

*(Tabela 3)*

**5. Wnioski**

Na podstawie powyższych obserwacji zauważam, iż indeksacja wydłużyła czas wykonywania się poszczególnych zapytań. Zaledwie w jednym przypadku średnia czasu kompilacji się zmniejszyła po dodaniu indeksów (jest to średnia 1 testu w SQL Server Management Studio). Wnioskuję zatem, że indeksowanie niekorzystnie wpływa na wydajność zapytań.

Ponadto, dla stworzonych tabel SQL Server Management Studio sprawuje się znacznie lepiej niż PostgreSQL. Wyjątkowo dużą różnicę w czasach zaobserwowano dla testu 3.

Zaobserwowano również, że normalizacja w większości przypadków prowadzi do spadku wydajności. Zauważyć natomiast trzeba, iż pozwala ona na dużo lepsze segregowanie danych, niż ma się to w tabelach nieznormalizowanych. Argumentem za tabelami znormalizowanymi jest również fakt, że są one przyjazne dalszemu rozwojowi tabel (to znaczy modyfikacji, dodawaniu i usuwaniu danych).