**ĆWICZENIE 10**

**Analiza porównawcza wydajności złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych.**

**1.Wstęp**

Niniejsze opracowanie ma na celu analizę efektywności różnych metod łączenia danych oraz zapytań zagnieżdżonych na dużych zbiorach danych. Dane te zostały stworzone poprzez połączenie informacji z tabeli geochronologicznej z syntetycznymi danymi z tabeli Milion wypełnionej kolejnymi liczbami naturalnymi od 0 do 999 999. Do tego celu posłużono się systemami MySQL oraz PostgreSQL.

**2.Konfiguracja sprzętowa i programowa**

Wszystkie testy omówione w niniejszym opracowaniu odbyły się z wykorzystaniem komputera o następujących parametrach:

* CPU: AMD Ryzen 5 5600X 6-Core Processor, 3.70 GHz
* RAM: 16,0 GB
* OS: Windows 10 Home
* Dysk twardy: SSD 1 TB

Jako systemy zarządzania bazami danych wybrano oprogramowanie wolno dostępne:

* PostgreSQL 16.3
* MySQL 20.1.10

**3.Wykonanie**

**3.1** **Utworzenie baz danych**

**** Na początku stworzona została baza danych, która zawierała tabele: ‘GeoEon’, ‘GeoEra’, ‘GeoOkres’ oraz ‘GeoPietro’ wypełnione odpowiednimi danymi. Następnie utworzono tabelę ‘GeoTabela’ za pomocą polecenia:

Zapytanie tworzy nową tabelę ‘GeoTabela’, łącząc w naturalny sposób dane z pięciu istniejących tabel: GeoPietro, GeoEpoka, GeoOkres, GeoEra, oraz GeoEon. Umożliwiło to szybki dostęp do wszystkich danych tabeli geochronologicznej za pomocą jednego zapytania prostego.

Tabela ‘Milion’ została utworzona na podstawie odpowiedniego autozłączenia tabeli ‘Dziesiec’ wypełnionej liczbami od 0 do 9 [14]:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

**3.2 Kryteria testów oraz kod użyty do ich wykonania na przykładzie MySQL**

* **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

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

* **Zapytanie 2 (2 ZL),** którego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci znormalizowanej, reprezentowaną przez złączenia pięciu tabel: **Obraz zawierający tekst, zrzut ekranu, Czcionka

  Opis wygenerowany automatycznie**
* **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:

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

* **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:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

**3.3 Testy wydajności i wyniki**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 ZL | | 2 ZL | | 3 ZG | | 4 ZG | |
| **Bez indeksów** | MIN | ŚR | MIN | ŚR | MIN | ŚR | MIN | ŚR |
| MySQL | 65 | 71 | 66 | 73 | 1207 | 1258 | 1112 | 1146 |
| PostgreSQL | 112 | 120 | 121 | 137 | 2379 | 2400 | 2090 | 2112 |
| **Z indeksami** |  | | | | | | | |
| MySQL | 63 | 68 | 65 | 70 | 68 | 79 | 1230 | 1258 |
| PostgreSQL | 101 | 105 | 130 | 142 | 302 | 332 | 2602 | 2670 |

Dla każdego zapytania zostały wykonane wielokrotne testy, a ich wyniki pod względem czasu [ms] wykonania zapytań prezentują się następująco:

Rys. 2 Wyniki w MySQL uwzględniające średni czas wykonywania poleceń

Rys. 3 Wyniki w PostgreSQL uwzględniające średni czas wykonywania poleceń

**4.Wnioski**

* W przypadku MySQL zastosowanie indeksów nieznacznie poprawiło wydajność 1 i 2 zapytania. Sytuacja dynamicznie się zmieniła dla 3 zapytania, w którym ten czas był już około 16 razy szybszy. Wyjątkiem jest 4 zapytanie, gdzie czas wykonania był dłuższy z indeksami.
* W PostgreSQL indeksy poprawiły oraz pogorszyły wydajność dla tych samych zapytań (rys.2,rys.3) co w przypadku MySQL.
* PostgreSQL wykazywał ok. 2 razy dłuższe czasy wykonania w porównaniu do MySQL dla większości zapytań zarówno przed, jak i po dodaniu indeksów.
* Postać zdenormalizowana jest w większości przypadków wydajniejsza.

Dodanie odpowiednich indeksów jest kluczowe dla poprawy wydajności obu baz danych, szczególnie przy operacjach złączeń. Wnioski te sugerują, że zarówno MySQL, jak i PostgreSQL mogą być skutecznymi rozwiązaniami w zależności od konkretnych wymagań projektowych, przy czym kluczowe jest optymalne projektowanie struktury bazy danych i odpowiednie indeksowanie dla zapewnienia optymalnej wydajności zapytań.