Wojciech Sutowicz

25.05.2023

WYDAJNOŚĆ ZŁĄCZEŃ I ZAGNIEŻDŻEŃ DLA SCHEMATÓW ZNORMALIZOWANYCH I ZDENORMALIZOWANYCH

Treść ćwiczenia

Naszym zadaniem w ćwiczeniu było przeanalizowanie czasu wykonywania się poszczególnych poleceń, które bazowały na polecaniach złączeń i zagnieżdżeń. W celu przeprowadzenia testów należało stworzyć bazę danych, za wzór użyto tabele stratygraficzna, która obrazuje przebieg historii Ziemi. Dane w tabeli są ustalone przez Międzynarodową Komisję Stratygrafii (ICS), jej ostatnia aktualizacja odbyła się w kwietniu 2023r.

Testy przeprowadzono na systemie operacyjnym Windows 10 w 3 różnych systemach zarządzania bazami danych:

- PostgreSQL
- MySQL
- SQL Server

Sprzęt i oprogramowanie

Specyfikacja urządzenia:

- Procesor -> Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz
- Dysk HDD -> HFS128G39TND-N210A
- Dysk SSD -> WDC WD10SPZX-21Z10TO
- Karta graficzna -> Intel(R) UHD Graphics 630
  Nvidia Geforce GTX 1050 TI
- Pamięć ram -> 8,00 GB
- System operacyjny -> Windows 10

System:

- SQL Server for Windows
- PostgreSQL
- MySQL

Środowisko programistyczne:

- SQL Server Managment Studio
- DataGrip 2023.1.2 Postgres
- MySQL Workbench 8.0 CE

Testy

Celem analizy było zbadanie i porównanie wydajności operacji złączeń i zagnieżdżonych zapytań na tabeli geologicznej. Testy wykonano na najpopularniejszych narzędziach bazodanowych typu open source.

W zapytaniach testowych połączono dane z tabeli stratygraficznej z danymi o rozkładzie jednostajnym z tabeli Milion, która była wypełniona liczbami naturalnymi od 0 do 999 999.

```
CREATE TABLE Milion
(liczba int);

INSERT INTO Milion SELECT
    a1.cyfra+10*a2.cyfra+100*a3.cyfra+1000*a4.cyfra+10000*a5.cyfra+100000*a6.cyfra AS liczba
FROM Dziesiec a1, Dziesiec a2, Dziesiec a3, Dziesiec a4, Dziesiec a5, Dziesiec
a6;
```

Testy wykonywano po kilkanaście razy na komputerze dla każdego systemu zarządzania bazą danych, przy czym w trakcie testów na komputerze były odpalone inne aplikacje, np. przeglądarka Opera.

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:

```
SELECT COUNT(*) FROM Milion
INNER JOIN GeoTabela ON (Milion.liczba % 102 = 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:

PostgreSQL:

```
SELECT COUNT(*) FROM Milion INNER JOIN GeoPietro ON
(mod(Milion.liczba,102)=GeoPietro.id_pietro) NATURAL JOIN GeoEpoka NATURAL JOIN
GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon;
```

MySQL: zapytanie wyglądało identycznie jak w PostgreSQL

SQL Server:

```
SELECT COUNT (*) FROM Milion
INNER JOIN GeoPietro ON (Milion.liczba % 102 = GeoPietro.id_pietro)

INNER JOIN GeoEpoka ON GeoPietro.id_epoka=GeoEpoka.id_epoka
INNER JOIN GeoOkres ON GeoEpoka.id_epoka = GeoOkres.id_okres
INNER JOIN GeoEra ON GeoOkres.id_era = GeoEra.id_era
INNER JOIN 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:

PostgreSQL:

```
SELECT COUNT(*) FROM Milion WHERE mod(Milion.liczba,102)=
(SELECT id_pietro FROM GeoTabela WHERE mod(Milion.liczba,102)=(id_pietro));
```

SQL Server:

```
SELECT COUNT (*) FROM Milion
WHERE ( Milion.liczba % 102) =
(SELECT id_pietro
FROM GeoTabela
WHERE (Milion.liczba % 102) = (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:

PostgreSQL:

```
SELECT COUNT(*) FROM Milion
WHERE Milion.liczba % 102 IN
(SELECT GeoPietro.id_pietro FROM GeoPietro
NATURAL JOIN GeoEpoka
NATURAL JOIN GeoOkres
NATURAL JOIN GeoEra
NATURAL JOIN GeoEon);
```

Omówienie testów

Wszystkie zapytania 1 ZL, 2 ZL, 3 ZG i 4 ZG były wykonywane na 2 sposoby, bez indeksów i razem z nimi. Każde zapytanie było generowanie kilkanaście razy w celu weryfikacji otrzymywanych wartości oraz wyliczeniem średniej, a także minimum i maksimum.

Tabela z wynikami wartości bez indeksów

	Czasy wykonania zapytań [ms]											
	1 ZL			2 ZL			3 ZG			4 ZG		
BEZ INDEKSÓW	MIN	MAX	SREDNIA	MIN	MAX	SREDNIA	MIN	MAX	SREDNIA	MIN	MAX	SREDNIA
PostgreSQL	158	271	209	447	636	513	11838	16073	13388	181	295	239
MySQL	1609	1750	1652	656	813	703	2454	2687	2524	641	750	708
SQL Server	15	50	33	99	122	106	41	55	48	50	90	64

Tabela z wynikami wartości z indeksami

	Czasy wykonania zapytań [ms]											
	1 ZL			2 ZL			3 ZG			4 ZG		
Z INDEKSAMI	MIN	MAX	SREDNIA	MIN	MAX	SREDNIA	MIN	MAX	SREDNIA	MIN	MAX	SREDNIA
PostgreSQL	163	244	191	487	655	537	11330	12287	11883	155	268	195
MySQL	1594	1734	1642	2500	2672	2590	2375	2547	2465	2484	2766	2587
SQL Server	43	50	46	100	123	109	42	53	47	51	77	58

Zauważyć można, iż wartości dla MySQL mamy do czynienia z konsekwentnym czasem w stosunku do reszty narzędzi. Odchylenie między najmniejszą a największą wartością jest stosunkowo małe, ok 200ms.

Widać najszybszy czas wykonania instrukcji dla SQL Server, z kolei najgorzej poradził sobie MySQL mając najgorsze czasy aż w 3 zapytaniach (1 ZL, 2 ZL, 4ZG).

Z pośród wszystkich instrukcji to zapytanie 3 dla PostreSQL wywołało największy okres oczekiwania, średnio 13 388ms. PostreSQL w przypadku pozostałych testów wypadł już lepiej.

Wykresy


Średni czas zapytań bez indeksów
14000
12000
10000
8000
6000
4000
2000
0
CZAS [MS]
1 ZL
2 ZL
3 ZG
4 ZG
ZAPYTANIA
PostgreSQL
MySQL
SQL Server


Średni czas zapytań z indeksami
12000
10000
8000
6000
4000
2000
0
CZAS [MS]
1 ZL
2 ZL
3 ZG
4 ZG
ZAPYTANIA
PostgreSQL
MySQL
SQL Server

Wnioski

- We wszystkich testach SQL Server sprawuje się zdecydowanie najlepiej

- System PostgreSQL jest optymalizowany w ten sposób, iż zapytanie wykonuje się równie szybko dla postaci znormalizowanej, jak i zdenormalizowanej, różnica pozostaje w przypadku zapytania 3.

- PostresSQL najsłabiej poradził sobie w postaci zdenormalizowanej, przy złączeniu jest wykonywanym poprzez zagnieżdżenie skorelowane (3 ZG)

- Użycie indeksów w większości rozważanych przypadkach nie wielko lecz minimalnie zwiększa wykonywanie zapytania, zarówno złączeń, jak i zagnieżdżeń skorelowanych. Niewielka różnica zachodzi w PostgreSQL

Bibliografia

1. https://pl.wikipedia.org/wiki/Tabela_stratygraficzna

2. https://aghedupl.sharepoint.com/sites/Bazydanych140-GIN-1S-044/Materiay%20z%20zaj/Ćwiczenia%209.pdf?CT=1685034164643&OR=ItemsView