Wydajność złączeń I zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych

Bazy danych I
Szymon Flis

Wstęp

Celem analizy było porównanie wpływu normalizacji oraz indeksacji tabel na wydajność wykonywania zapytań w relacyjnych bazach danych. Test przeprowadzono dla dwóch systemów bazodanowych: PostgreSQL oraz Microsoft SQL Server.

Analiza wzorowana była na artykule *Wydajność złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych.*

Konfiguracja sprzętowa

Całość testów przeprowadzono na komputerze o specyfikacji:

- CPU: Intel Core i7 7700HQ
- GPU: NVIDIA GeForce GTX 1050Ti
- RAM: 16GB DDR4 2400
- SO: Windows 10
- PostgreSQL 13.2
- MySQL 8.0.25

Metodyka przeprowadzania testów

Jako parametr informujący o wydajności wykonywania kwerend wykorzystano czas wykonania każdej z nich. Skrypty tworzące bazy danych, poszczególne tabele, indeksy oraz zawierające interesujące nas kwerendy dostępne są w repozytorium GitHub.

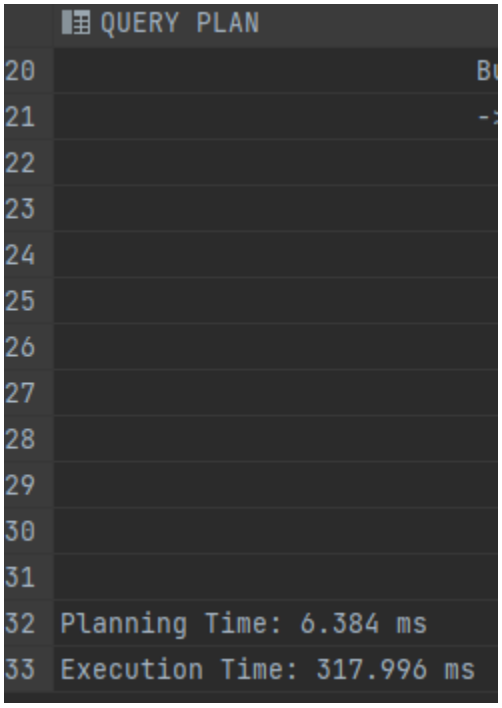
W przypadku Microsoft SQL Server czas wykonania odczytywany był z Client Statistics (Rys. 1), zaś w przypadku PostgreSQL (Rys. 2) odczytywano wartości zwracane przez wykonywanie zapytań z dodatkiem EXPLAIN ANALYZE.

193 %

Results | Messages | Client Statistics

	Trial 10	Trial 9	Trial 8	Trial 7	Trial 6	Trial 5	Trial 4	Trial 3	Trial 2	Trial 1	Average
Client Execution Time	21:55:35	21:54:57	21:54:49	21:54:40	21:54:31	21:54:25	21:54:01	21:53:48	21:53:35	21:53:25	
Query Profile Statistics											
Number of INSERT, DELETE and UPDATE statements	0 → 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0.0000
Rows affected by INSERT, DELETE, or UPDATE stateme...	0 → 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0.0000
Number of SELECT statements	2 → 2	→ 2	→ 2	→ 2	→ 2	↑ 1	↓ 2	→ 2	→ 2	↑ 1	→ 1.8000
Rows returned by SELECT statements	2 → 2	→ 2	→ 2	→ 2	→ 2	↑ 1	↓ 2	→ 2	→ 2	↑ 1	→ 1.8000
Number of transactions	0 → 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0	→ 0.0000
Network Statistics											
Number of server roundtrips	2 → 2	→ 2	→ 2	→ 2	→ 2	→ 2	→ 2	→ 2	→ 2	↑ 1	→ 1.9000
TDS packets sent from client	2 → 2	→ 2	→ 2	→ 2	→ 2	→ 2	→ 2	→ 2	→ 2	↑ 1	→ 1.9000
TDS packets received from server	2 → 2	→ 2	→ 2	→ 2	→ 2	→ 2	→ 2	→ 2	→ 2	↑ 1	→ 1.9000
Bytes sent from client	274 ↓ 754	↑ 334	↓ 734	↑ 304	↓ 872	↑ 750	↑ 334	↓ 730	↑ 216	→ 530.2000	
Bytes received from server	78 → 78	→ 78	→ 78	→ 78	↓ 136	↑ 78	→ 78	→ 78	↑ 41	→ 80.1000	
Time Statistics											
Client processing time	1 ↓ 3	↑ 1	→ 1	↓ 2	↑ 0	↓ 1	→ 1	→ 1	↓ 4	→ 1.5000	
Total execution time	244 ↓ 270	↑ 248	↓ 267	↑ 239	↓ 645	↑ 274	↓ 567	↑ 304	↑ 285	→ 334.3000	
Wait time on server replies	243 ↓ 267	↑ 247	↓ 266	↑ 237	↓ 645	↑ 273	↓ 566	↑ 303	↑ 281	→ 332.8000	

Rysunek 1. Tabela statystyk zawierająca czas wykonywania poszczególnych kwerend (Microsoft SQL Server Management Studio).



Rysunek 2. Czas wykonania zwracany przez funkcje EXPLAIN ANALYZE (PostgreSQL).

Każda seria testów przeprowadzana była na nowo założonej bazie danych w celu minimalizacji wpływu wewnętrznych optymalizacji systemów bazodanowych na czasy wykonania zapytań.

Kolejne kwerendy których czas był spisywany:

Q1

```
select COUNT(*) from Milion INNER JOIN GeoTabela on Milion.liczba%68 =
GeoTabela.id_pietro;
```

Q2

```
select COUNT(*) from Milion inner join GeoPietro on
(Milion.liczba%68=GeoPietro.id_pietro)
inner join GeoEpoka on GeoPietro.id_epoka =GeoEpoka.id_epoka
inner join GeoOkres on GeoEpoka.id_okres = GeoOkres.id_okres
inner join GeoEra on GeoEra.id_era = GeoOkres.id_era
inner join GeoEon on GeoEon.id_eon = GeoEra.id_eon
```

Q3

```
select COUNT(*) from Milion where Milion.liczba%68 =
(select id_pietro from GeoTabela where Milion.liczba%68=id_pietro)
```

Q4

```
select COUNT(*) from Milion where Milion.liczba%68 in
(select GeoPietro.id_pietro from GeoPietro
inner join GeoEpoka on GeoPietro.id_epoka = GeoEpoka.id_epoka
inner join GeoOkres on GeoEpoka.id_okres = GeoOkres.id_okres
inner join GeoEra on GeoEra.id_era = GeoOkres.id_era
inner join GeoEon on GeoEon.id_eon = GeoEra.id_eon)
```

Wyniki testów

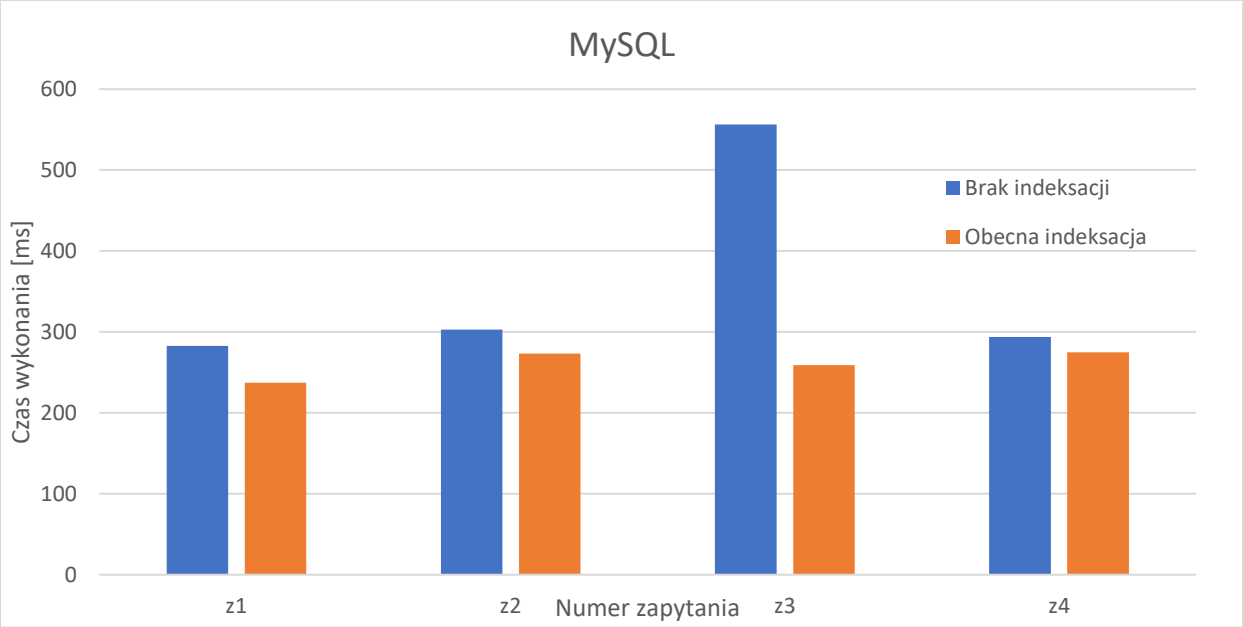
Przeprowadzono w sumie 40 testów – 5 serii po 8 w czym 4 testy bez indeksacji tabel, 4 z indeksacją.

Wyniki uśredniono i przedstawiono w tabeli 1.

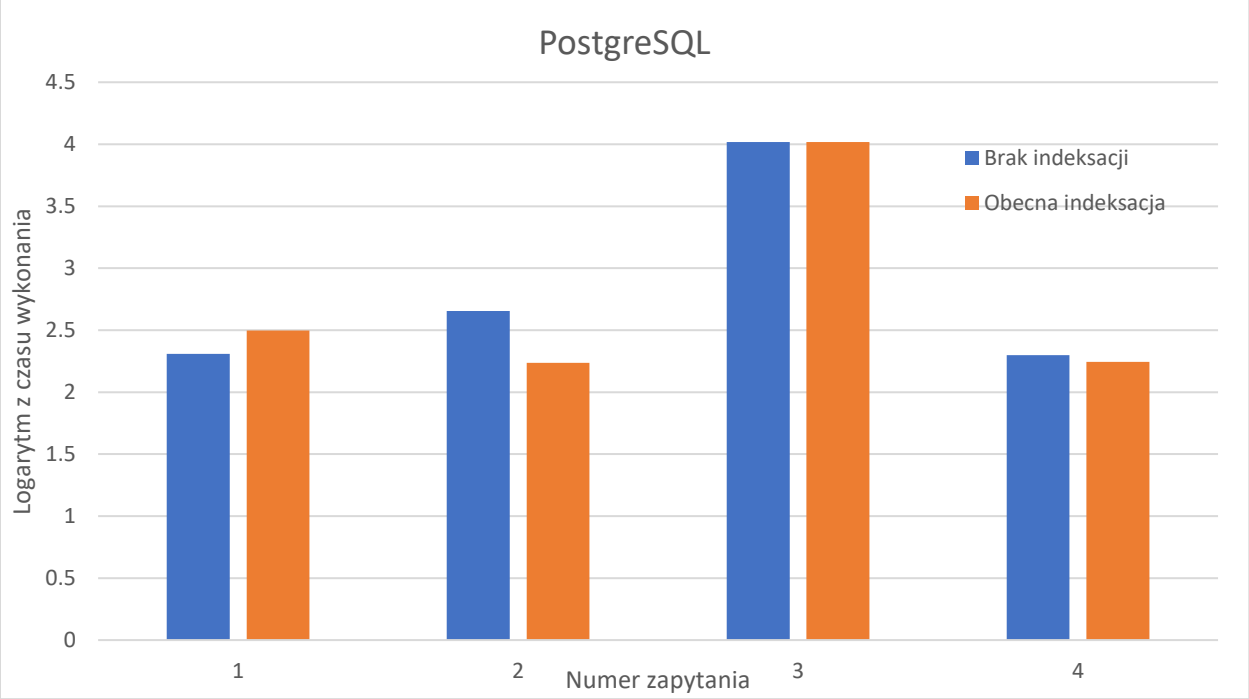
	Q1	Q2	Q3	Q4
MySQL (brak indeksacji)	282.8	303	556.4	293.6
MySQL (obecna indeksacja)	237.2	273.2	258.8	274.8
PostgreSQL (brak indeksacji)	203.28	451.37	10399.4	198.99
PostgreSQL (obecna indeksacja)	172.89	314.76	10388.44	175.75

Tabela 1. Uśrednione czasy wykonania 4 zapytań testowych z uwzględnieniem indeksacji.

Wyniki w postaci histogramu przedstawiono na rysunkach 3 i 4.



Rysunek 3. Histogram uśrednionych wyników testów dla kolejnych zapytań w Microsoft SQL Server.



Rysunek 4. Histogram uśrednionych wyników testów dla kolejnych zapytań w PostgreSQL.

Wnioski

W znakomitej większości indeksacja skróciła czas wykonywania kwerend typu SELECT, jedynie w przypadku zapytania 1 dla bazy PostgreSQL obecność indeksacji spowolniła wykonanie zapytania – niekoniecznie wynika to z samej indeksacji, może to być równie dobrze losowe zdarzenie działające w tle w trakcie wykonywania zapytania po raz pierwszy, oprogramowanie w którym testy były uruchamiane i wiele innych.

Wyniki wskazują również na nierówną prędkość wykonywania zapytań przez wykorzystane systemy bazodanowe – zapytania 1 i 4 wykonywane są szybciej przez PostgreSQL, 2 i 3 przez MySQL. Nie jest to jednak miarodajną statystyką ze względu na chociażby odmienny sposób prowadzenia pomiaru czasów wykonywania – w PostgreSQL była to funkcja EXPLAIN ANALYZE będąca częścią dialektu samego PostgreSQL'a, w MySQL Client statistics będąca częścią środowiska do zarządzania bazą (SMSS) – mogą one w różny sposób mierzyć czas wykonania.

Bibliografia

[1] Ł.Jaśnica, A.Piórkowski, Wydajność złączeń ii zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych, (Studia Ingormatica, Volume 31, Number 2A (89), 2010).

[2] https://www.sqlpedia.pl/pomiar-wydajnosci-zapytan-sql-server/