Testy wydajności

Wyk. Gabriela Jarosz

# **Konfiguracja sprzętowa i programowa**

Testy wydajności zostały wykonane na komputerze o następujących parametrach:

* CPU: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
* RAM: 8,00 GB (dostępne: 7,89 GB)
* SSD: WD Blue SN550 NVM
* S.O.: Windows 10 Home

Jako systemy zarządzania bazami danych wybrano oprogramowanie wolno dostępne:

* MS SQL Server Management Studio 18, wersja: 15.0.2000.5
* PostgresSQL, wersja: 13.3

# **Kryteria testów**

W teście wykonano szereg zapytań sprawdzających wydajność złączeń i zagnieżdżeń

z tabelą geochronologiczną w wersji zdenormalizowanej i znormalizowanej. Procedurę tekstową przeprowadzono w dwóch etapach:

* pierwszy etap obejmował zapytania bez nałożonych indeksów na kolumny danych (jedynymi indeksowanymi danymi były dane w kolumnach będących kluczami głównymi poszczególnych tabel),
* w drugim etapie nałożono indeksy na wszystkie kolumny biorące udział w złączeniu.

Zasadniczym celem testów była ocena wpływu normalizacji na zapytania złożone – złączenia i zagnieżdżenia (skorelowane). W tym celu zaproponowano cztery zapytania:

* 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:

PostgreSQL:

SELECT COUNT(\*) FROM Milion INNER JOIN GeoTabela ON (mod(Milion.liczba,68)=(GeoTabela.id\_pietro));

MS SQL Server:

SELECT COUNT(\*) FROM Milion

INNER JOIN GeoTabela

ON 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:

PostgreSQL:

SELECT COUNT(\*) FROM Milion INNER JOIN GeoPietro ON

(mod(Milion.liczba,68)=GeoPietro.id\_pietro) NATURAL JOIN GeoEpoka NATURAL JOIN

GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon;

MS SQL Server:

SELECT COUNT(\*) FROM Milion

INNER JOIN GeoPietro

ON (Milion.liczba%77) = GeoPietro.id\_pietro

JOIN GeoEpoka ON GeoPietro.id\_epoka = GeoEpoka.id\_epoka

JOIN GeoOkres ON GeoEpoka.id\_okres = GeoOkres.id\_okres

JOIN GeoEra ON GeoOkres.id\_era = GeoEra.id\_era

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,68)=

(SELECT id\_pietro FROM GeoTabela WHERE mod(Milion.liczba,68)=(id\_pietro));

MS SQL Server:

SELECT COUNT(\*) FROM Milion

WHERE Milion.liczba%77 =

(SELECT id\_pietro FROM GeoTabela

WHERE 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:

PostgreSQL:

SELECT COUNT(\*) FROM Milion WHERE mod(Milion.liczba,68) IN

(SELECT GeoPietro.id\_pietro FROM GeoPietro NATURAL JOIN GeoEpoka NATURAL JOIN

GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon;

MS SQL Server:

SELECT COUNT(\*) FROM Milion

WHERE Milion.liczba%77 IN

(SELECT GeoPietro.id\_pietro FROM GeoPietro

JOIN GeoEpoka ON GeoPietro.id\_epoka = GeoEpoka.id\_epoka

JOIN GeoOkres ON GeoEpoka.id\_okres = GeoOkres.id\_okres

JOIN GeoEra ON GeoOkres.id\_era = GeoEra.id\_era

JOIN GeoEon ON GeoEra.id\_eon = GeoEon.id\_eon);

# **Wyniki testów**

Każdy test przeprowadzony został wielokrotnie. W przypadku systemu MS SQL Server miała miejsce dość duża zgodność kolejnych prób. Wyniki testów zamieszczono w tabeli 1.

Tabela 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1ZL** | | **2ZL** | | **3ZG** | | **4ZG** | |
| **BEZ INDEKSÓW** | MIN | ŚR | MIN | ŚR | MIN | ŚR | MIN | ŚR |
| **MS SQL Server** | 1835 | 2521,7 | 2161 | 2334,9 | 1876 | 2008,4 | 2208 | 2302 |
| **PostgreSQL** | 242 | 286,3 | 373 | 458,5 | 27824 | 30335,6 | 228 | 256,6 |
| **Z INDEKSAMI** |  |  |  |  |  |  |  |  |
| **MS SQL Server** | 1819 | 1950,3 | 2126 | 2227,5 | 1793 | 1901,2 | 2028 | 2224,5 |
| **PostgreSQL** | 183 | 258,8 | 272 | 359,8 | 28681 | 30379 | 225 | 257,1 |

Analizę wyników ułatwiają wykresy (rys. 1,2 i 3) – ze względu na dość duże wartości pojedyncze w systemie PostgreSQL rozważono dwie wersje – pełna skala liniowa i skala liniowa częściowa (aby ułatwić porównanie niskich wartości). Wyniki zestawiono wysuwając na pierwszy plan związki z tezą artykułu – czy wersja znormalizowana jest wolniejsza czy szybsza od wersji zdenormalizowanej.

Rysunek 1

Rysunek 2

Rysunek 3

# **Wnioski**

Otrzymane wyniki pozwalają wyciągnąć następujące wnioski związane z tezą artykułu:

* Postać zdenormalizowana jest w większości przypadków wydajniejsza.
* Jedyny przypadek, kiedy postać znormalizowana jest szybsza, to zagnieżdżenie skorelowane – w podzapytaniu wewnętrznym w wersji zdenormalizowanej dokonywany jest odczyt dużej tabeli danych niezaindeksowanych; wydłużenie czasu wykonania ma miejsce

w systemie MS SQL Server, PostgreSQL bowiem dokonuje skutecznej optymalizacji takiego zapytania.

Testy pozwalają też przedstawić dodatkowe spostrzeżenia związane z rozważanym przypadkiem:

* Zagnieżdżenia skorelowane są dużo wolniejsze w wykonaniu niż złączenia.
* Użycie indeksów w systemie MS SQL Server we wszystkich rozważanych przypadkach przyśpiesza wykonanie zapytań, zarówno złączeń, jak i zagnieżdżeń skorelowanych.
* System PostgreSQL dokonuje analizy tabeli i indeksacja minimalnie przyspieszyła wykonania przedstawionych złączeń, jedynie dla zapytania 3 ZG użycie indeksów wydłużyło czasy zapytań.
* Złączenia 4Z w PostgreSQL są tak optymalizowane, iż zapytanie składające się z samych złączeń wykonuje się równie szybko dla postaci znormalizowanej, jak i zdenormalizowanej co widać w złączeniu 4Z, jednak postać znormalizowana wykonuje się szybciej przy złączeniu 2Z.
* Systemów zarządzania bazami danych nie da się porównać jednoznacznie – o ile złączenia najszybciej były wykonane przez system PostgreSQL, to zagnieżdżenia w większości przypadków były szybciej przetworzone przez MS SQL Server.

Podsumowaniem rozważań jest wniosek, iż normalizacja w większości przypadków prowadzi do spadku wydajności, ale warto jest tu przypomnieć jej zalety, a mianowicie łatwą konserwację, rozwój schematu oraz porządek, jaki ona wprowadza.