Kraków - 19.06.2022

PROJEKT NR 1

Bazy Danych - “Indeksowanie”

Wykonał:
Mateusz Pajączek | 408042 | gr. 2

Spis treści

Rozdział I - Wstęp

Artykuł jest przygotowywany na podstawie projektu Łukasza Jajeśnica, który przeprowadził badania pod patronatem Adama Piórowskiego z Akademii Górniczo Hutniczej, Katedry Geoinformatyki i Informatyki Stosowanej.

Projekt ten ma na celu sprawdzenie wydajności zapytań, złączeń i zagnieżdżeń, zindeksowanych lub niezindeksowanych dla schematów znormalizowanych i zdenormalizowanych. Został on zrealizowany w dwóch rozwiązaniach bazoda nowych - PostgreSQL oraz SQL Server. Dane, na których opiera się projekt są związane z tabelą geochronologiczną. Zostały one przedstawione w postaci dwóch schematów bazodanowych - schemat znormalizowany płatka śniegu i schemat zdenormalizowany - gwiazdy.

Rozdział II - Materiały

Głównym źródłem danych była tabela geochronologiczna, zawierająca jednostki geochronologiczne mające wymiar czasowy (eon, era, okres, epoka i wiek), oraz odpowiadające im jednostki stratygraficzne.

Rozdział III - Metody

1. *Testy wydajności.*

   W testach skupiono się na porównaniu wydajności złączeń oraz zapytań zagnieżdżonych, wykonywanych na tabelach o dużej liczbie danych. Przetestowano najpopularniejsze darmowe rozwiązania bazodanowe:
   - SQL Server,
   - PostgreSQL.

   W zapytaniach testowych łączono dane z tabeli geochronologicznej z syntetycznymi danymi o rozkładzie jednostajnym z tabeli *Milion*, wypełnionej kolejnymi liczbami naturalnymi od 0 do 999 999. Tabela *Milion* została utworzona na podstawie odpowiedniego autozłączenia tabeli *Dziesiec* wypełnionej liczbami od 0 do 9.

```
CREATE TABLE Dziesiec(cyfra int, bit int);

INSERT INTO Dziesiec VALUES
(0, 0000000),
(1, 0000001),
(2, 0000010),
(3, 0000011),
(4, 0000100),
(5, 0000101),
(6, 0000110),
(7, 0000111),
(8, 0001000),
(9, 0001001);
```

```
CREATE TABLE Milion(liczba int, cyfra int, bit int);

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

2. *Konfiguracja sprzętowa i programowa.*

CPU: Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz 2.40 GHz,
RAM: 4,00 GB,
SSD: ST1000LM035-1RK172,
S.O.: Windows 10 Home.

Jako systemy zarządzania bazami danych wybrano oprogramowanie wolno dostępne:
MySQL,
PostgreSQL.

3. Etapy analizy wraz z wykorzystanymi funkcjami.

W teście wykonano szereg zapytań sprawdzających wydajność złączeń i zagnieżdżeń z tabelą geochronologiczną w wersji zdenormalizowanej i znormalizowanej. Procedurę testową 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:

```
SELECT COUNT(*) FROM Milion INNER JOIN GeoTabela ON
(mod(Milion.liczba,68)=(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:

```
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;
```

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:

```
SELECT COUNT(*) FROM Milion WHERE mod(Milion.liczba,68)=
(SELECT id_pietro FROM GeoTabela WHERE
mod(Milion.liczba,68)=(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:

```
SELECT COUNT(*) FROM Milion WHERE mod(Milion.liczba,68)=
(SELECT GeoPietro.id_pietro FROM GeoPietro NATURAL JOIN GeoEpoka
NATURAL JOIN GeoOkres NATURAL JOIN GeoEra NATURAL JOIN
GeoEon;
```

Rozdział IV - Wyniki

Każdy test przeprowadzono pięciokrotnie, wyniki skrajne pominięto.

Wyniki testów zamieszczono w tabeli.

	1 ZL		2 ZL		3 ZG		4 ZG	
BEZ INDEKSÓW	MIN	ŚR	MIN	ŚR	MIN	ŚR	MIN	ŚR
SQL Server	86	90	91	101	7478	8219	91	105
PostgreSQL	387	416	708	864	18397	24453	424	479
Z INDEKSAMI								
SQL Server	63	66	67	70	2789	2981	44	54
PostgreSQL	363	374	810	844	17333	22962	423	493

Rozdział V - Podsumowanie

Wnioski:

Teza artykułu:

- W większości przypadków wydajniejsza jest postać zdenormalizowana.
- Postać znormalizowana wykonywana jest w krótszym czasie tylko w jednym przypadku – gdy jest to zagnieżdżenie skorelowane (w podzapytaniu wewnętrznym w wersji zdenormalizowanej dokonywany jest odczyt dużej tabeli danych niezaindeksowanych).

Dodatkowe spostrzeżenia wynikające z przeprowadzonych testów:

- Zagnieżdżenia skorelowane są dużo wolniejsze w wykonaniu niż złączenia.
- Użycie indeksów w systemie SQL Server we wszystkich rozważanych przypadkach przyśpiesza wykonanie zapytań, zarówno złączeń, jak i zagnieżdżeń skorelowanych.
- System PostgreSQL dokonuje analizy tabeli i indeksacja nie przyspieszyła (ani nie spowolniła) wykonania przedstawionych złączeń i zagnieżdżenia 3 ZG, jedynie dla zapytania 4 ZG użycie indeksów wydłużyło czasy zapytań.
- Złączenia w PostgreSQL są tak optymalizowane, iż zapytanie składające się z samych złączeń wykonuje się równie szybko dla postaci znormalizowanej, jak i zdenormalizowanej, różnica (i to istotna) pozostaje w przypadku wystąpienia złączeń jako wewnętrznego podzapytania zapytania skorelowanego, które jest często wykonywane.

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.