1. Wstęp

Celem badań było przetestowanie wydajności systemów zarządzania bazami danych pod kątem postaci baz, a także zastosowania indeksów na tychże bazach.

1. Dane

Na potrzeby testów utworzono:

1. 5 tabel (Eon, Era, Okres, Epoka, Pietro) tworzących znormalizowaną tablicę stratygraficzną. Prawie wszystkie tabele opierają się na tej samej budowie, tj. składają się z klucza głównego (id\_[nazwa tabeli]), nazwy jednostki oraz identyfikatora jednostki wyższej do której dana jednostka przynależy (wyjątkiem jest tabela Eon). Baza będzie wykorzystywana przez złączenie tych pięciu tabel ze sobą.

Przykład dla tabeli Epoka (oba SZBD):

CREATE TABLE Epoka(id\_epoka int Primary Key, nazwa\_epoka text, id\_okres int);

1. Tabelę o nazwie ‘tabela’, powstałą przez złączenie tabel z bazy znormalizowanej, będącą postacią zdenormalizowaną tablicy stratygraficznej.

Przykład utworzenia:

* PostgreSQL:

CREATE TABLE Tabela AS (SELECT \* FROM Pietro NATURAL JOIN Epoka NATURAL JOIN Okres NATURAL JOIN Era NATURAL JOIN Eon );

* SQL Server:

SELECT Eon.id\_eon, Era.id\_era, Okres.id\_okres, Epoka.id\_epoka, Pietro.id\_pietro, nazwa\_pietro, nazwa\_okres, nazwa\_era, nazwa\_epoka, nazwa\_eon into Tabela FROM (((Pietro

Inner JOIN Epoka on Pietro.id\_epoka = Epoka.id\_epoka)

Inner JOIN Okres on Epoka.id\_okres = Okres.id\_okres)

Inner JOIN Era on Okres.id\_era = Era.id\_era)

Inner JOIN Eon on Era.id\_eon = Eon.id\_eon

1. Tabelę o nazwie milion będącą stertą danych w postaci liczb z przedziału od 0 do 999999, utworzona z tabeli dziesięć, zawierającej cyfry od 0 do 9. Złączenie jej z wyżej wspomnianymi bazami danych będzie przedmiotem testów.

Przykład utworzenia (oba SZBD):

CREATE TABLE Milion(liczba int not null,cyfra int, bit int);

INSERT INTO Milion SELECT a1.cyfra +10\* a2.cyfra +100\*a3.cyfra + 1000\*a4.cyfra

+ 10000\*a5.cyfra + 100000\*a6.cyfra AS liczba , a1.cyfra AS cyfra, a1.bit AS bit

FROM Dziesiec a1, Dziesiec a2, Dziesiec a3, Dziesiec a4, Dziesiec a5, Dziesiec

a6 ;

1. Parametry komputera oraz SZBD:
2. CPU: Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz
3. RAM: Pamięć DDR4 (8 GB) 2666 MHz
4. Dysk Twardy: SSD, 512 GB,
5. System Operacyjny.: Windows 11 Home

Użyte SZBD:

1. Microsoft SQL Server Management Studio 15.0.18404.0
2. PostgreSQL 14.4.
3. Przebieg testów

Badania oparto na zastosowaniu 4 zapytań złożonych (do każdego z nich zastosowano także operację modulo, która pozwala dopasować zakresy kolumn złączanych:

1. 1 ZL – złączenie tablicy milion z tablicą stratygraficzną w postaci zdenormalizowanej

Przykłady:

* PostgreSQL

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

* SQL Server

SELECT COUNT(\*) FROM Milion INNER JOIN Tabela ON (Milion.liczba % 68=(Tabela.id\_pietro));

1. 2 ZL – złączenie tablicy milion z tablicą stratygraficzną w postaci znormalizowanej

Przykłady:

* PostgreSQL

SELECT COUNT(\*) FROM Milion INNER JOIN Pietro ON (mod(Milion.liczba,68)=Pietro.id\_pietro) NATURAL JOIN Epoka NATURAL JOIN Okres NATURAL JOIN Era NATURAL JOIN Eon;

* SQL Server

SELECT COUNT(\*) FROM (((Milion INNER JOIN Pietro ON (Milion.liczba % 68=Pietro.id\_pietro)

Inner JOIN Epoka on Pietro.id\_epoka = Epoka.id\_epoka)

Inner JOIN Okres on Epoka.id\_okres = Okres.id\_okres)

Inner JOIN Era on Okres.id\_era = Era.id\_era)

Inner JOIN Eon on Era.id\_eon = Eon.id\_eon

1. 3 ZG – złączenie tablicy milion z postacią zdenormalizowaną tabeli, przez zagnieżdżenie skorelowane

Przykłady:

* PostgreSQL

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

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

* SQL Server

SELECT COUNT(\*) FROM Milion WHERE Milion.liczba % 68=

(SELECT id\_pietro FROM Tabela WHERE Milion.liczba % 68=(id\_pietro));

1. 4 ZG – złączenie tablicy milion z postacią znormalizowaną tabeli, przez zagnieżdżenie skorelowane

Przykłady:

* PostgreSQL

SELECT COUNT(\*) FROM Milion

WHERE mod(Milion.liczba,68)=

(SELECT Pietro.id\_pietro FROM Pietro

NATURAL JOIN Epoka

NATURAL JOIN Okres

NATURAL JOIN Era

NATURAL JOIN Eon Where mod(Milion.liczba,68) = id\_pietro);

* SQL Server

SELECT COUNT(\*) FROM Milion

WHERE Milion.liczba % 68 =

(SELECT Pietro.id\_pietro FROM (((Pietro

Inner JOIN Epoka on Pietro.id\_epoka = Epoka.id\_epoka)

Inner JOIN Okres on Epoka.id\_okres = Okres.id\_okres)

Inner JOIN Era on Okres.id\_era = Era.id\_era)

Inner JOIN Eon on Era.id\_eon = Eon.id\_eon Where Milion.liczba % 68 = id\_pietro)

Dla każdego osobnego przypadku przeprowadzono kolejno 10 testów, wyciągając następnie z nich średnią. Ten sam zestaw czynności powtórzono po nałożeniu indeksów na wszystkie kolumny biorące udział w złączeniu.

1. Wyniki testów[[1]](#footnote-1):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PostgreSQL | | SQL Server | |
|  | Z indeksami | Bez indeksów | Z indeksami | Bez indeksów |
| 1 ZL | 160,7 | 172,2 | 131,3 | 85,3 |
| 2 ZL | 218,1 | 234 | 154,2 | 115,9 |
| 3 ZG | 14495,9 | 14855,7 | 115,6 | 72,5 |
| 4 ZG | 26738,9 | 26672,4 | 5578,8 | 7464,8 |

1. Wnioski

- Pod względem wydajności, we wszystkich testowanych przypadkach, lepsza okazuje się postać zdenormalizowana tablicy stratygraficznej

- Użycie indeksów w tabelach nie wpłynęło w znaczący sposób na szybkość wykonania zapytania przez oba programy. Jedynie w przypadku zapytania zagnieżdżonego dla postaci znormalizowane w SQL Serverze wydajność uległa zwiększeniu o ok. 25%.

1. Idx – zapytania po nałożeniu indeksów; DN – zapytania z postacią zdenormalizowaną tablicy; N – zapytania z postacią znormalizowaną tablicy [↑](#footnote-ref-1)