Bazy Danych

Wykonał: Kacper Czarnik

WYDAJNOŚĆ ZŁĄCZEŃ I ZAGNIEŻDŻEŃ DLA SCHEMATÓW ZNORMALIZOWANYCH I ZDENORMALIZOWANYCH

1) Wstęp

Celem zadania było zbadanie wpływu normalizacji oraz indeksacji tabel na czas wykonywania kwerend w dwóch środowiskach systemu zarządzania bazą danych: Microsoft SQL Server oraz PostgreSQL na podstawie artykułu:
Wydajność złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych — Productivity and nesting join the schemes and standardized and denormalized / Łukasz JAJEŚNICA, Adam PIÓRKOWSKI.

2) Teoria

Normalizacją bazy danych nazywamy doprowadzenie relacji do postaci normalnej, czyli takiej, w której nie występuję nadmiarowość, powtarzanie się tych samych informacji. Edgar Frank Codd wyróżnił następujące postaci normalne:

a) Relacja jest w pierwszej postaci normalnej (1NF), jeżeli wartości atrybutów są elementarne, atomowe(niepodzielne)
b) Relacja jest w drugiej postaci normalnej (2NF), jeżeli relacja jest w 1NF i by każdy atrybut niekluczowy był funkcyjnie zależny od klucza głównego
c) Relacja jest w trzeciej postaci normalnej (3NF), jeżeli relacja jest w 2NF i każdy atrybut wtórny jest bezpośrednio zależny od klucza głównego

Obecnie istnieją inne postacie normalne (4NF, 5NF, postać normalna Boyce`a-Codda), ale postać 3NF jest zazwyczaj wystarczająca, natomiast 4NF i 5NF są głównie używane tylko w ramach rozważań teoretycznych.

Indeks jest strukturą fizyczną w bazach danych, którego zadaniem jest optymalizaca szybkości wykonywania operacji na przeszukanie tabel.

3) Sposób wykonania testów:

Testy zostały wykonane przy wykorzystaniu Tabeli Geochronologicznej w postaci zdenormalizowanej oraz znormalizowanej i złączeniu jej z tabelą zawierającą milion liczb, przy czym złączenie w dwóch przypadkach było wykonane przez zagnieżdżenie skorelowane. Analiza obejmowała dwa etapy:

a) Bez nałożonych indeksów
b) Z nałożonymi indeksami

Wykonane zostało 10 różnych testów dla każdego zapytania, przy czym przedstawiona zostanie ich średnia arytmetyczna.

4) Opis fragmentów kodu w Microsoft SQL Server:

a) Przygotowania do testów

```
CREATE TABLE GeoEra (
    id_era INTEGER PRIMARY KEY,
    id_eon INTEGER,
    nazwa_era VARCHAR(25) NOT NULL
);

ALTER TABLE GeoEra
    ADD FOREIGN KEY (id_eon)
    REFERENCES GeoEon (id_eon);
```

Rys 1 - Utworzenie tabeli GeoEra z kluczem głównym id_era i nazwą oraz dodanie klucza obcego tabeli GeoEon, analogicznie dla innych tabel

```
INSERT INTO GeoEra VALUES
    (1, 1, 'Era Kenozoiczna'),
    (2, 1, 'Era Mezozoiczna'),
    (3, 1, 'Era Paleozoiczna');
```

Rys 2 - Wypełnienie wartościami

```
SELECT GeoPietro.id_pietro,GeoPietro.nazwa_pietro,
GeoEpoka.id_epoka,GeoEpoka.nazwa_epoka,
GeoOkres.id_okres,GeoOkres.nazwa_okres,
GeoEra.id_era,GeoEra.nazwa_era,
GeoEon.id_eon,GeoEon.nazwa_eon
INTO GeoTabela from GeoEon
 JOIN   GeoEra ON GeoEon.id_eon = GeoEra.id_eon
 JOIN  GeoOkres ON GeoEra.id_era = GeoOkres.id_era
 JOIN  GeoEpoka ON GeoOkres.id_okres = GeoEpoka.id_okres
 JOIN  GeoPietro ON GeoEpoka.id_epoka = GeoPietro.id_epoka
```

Rys 3 - Utworzenie zdenormalizowanej tabeli GeoTabela

```
CREATE TABLE Dziesiec(cyfra INT,bit INT)
INSERT INTO Dziesiec VALUES
    (0,1),
    (1,1),
    (2,1),
    (3,1),
    (4,1),
    (5,1),
    (6,1),
    (7,1),
    (8,1),
    (9,1);
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 + 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;
```

Rys 4 - Utworzenie tabeli Dziesięć oraz Milion i wypełnienie ich wartościami

b) Wykonanie testów

```
SELECT COUNT(*) FROM Milion JOIN GeoTabela ON
Milion.liczba%77=(GeoTabela.id_pietro);
```

Rys 5 - Zapytanie 1 (1 ZL) – złączenie tablicy miliona wyników z tabelą geochronologiczną w postaci zdenormalizowanej

```
SELECT COUNT(*) FROM Milion  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 GeoEra.id_era = GeoOkres.id_era
JOIN GeoEon ON GeoEon.id_eon = GeoEra.id_eon;
```

Rys 6 - Zapytanie 2 (2 ZL) – złączenie tablicy miliona wyników z tabelą geochronologiczną w postaci znormalizowanej

```
SELECT COUNT(*) FROM Milion WHERE Milion.liczba%77 =
(SELECT id_pietro FROM GeoTabela WHERE Milion.liczba%77=id_pietro);
```

Rys 7 - Zapytanie 3 (3 ZG) – złączenie tablicy miliona wyników z tabelą geochronologiczną w postaci zdenormalizowanej, przy czym złączenie jest wykonywane przez zagnieżdżenie skorelowane

```
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 GeoEra.id_era = GeoOkres.id_era
JOIN GeoEon ON GeoEon.id_eon = GeoEra.id_eon);
```

Rys 8 - Zapytanie 4 (4 ZG) – złączenie tablicy miliona wyników z tabelą geochronologiczną w postaci znormalizowanej, przy czym złączenie jest wykonywane przez zagnieżdżenie skorelowane

```
CREATE INDEX  index_Eon ON GeoEon(id_eon);
CREATE INDEX   index_Era ON GeoEra(id_era, id_eon);
CREATE INDEX   index_Okres ON GeoOkres(id_okres, id_era);
CREATE INDEX   index_Epoka ON GeoEpoka(id_epoka, id_okres);
CREATE INDEX   index_Pietro ON GeoPietro(id_pietro, id_epoka);

CREATE INDEX   index_Liczba ON Milion(liczba);
CREATE INDEX   index_GeoTabela ON GeoTabela(id_pietro, id_epoka, id_era, id_okres, id_eon);
```

Rys 9 - Dodanie indeksów do tabel

5) Konfiguracja sprzętowa i programowa

CPU: Intel Core i5-8300H, 2.5 GHz
RAM: DDR4 8GB
S.O.: Windows 10
Microsoft SQL Server 2019 (RTM) 15.0.2000.5
PostgreSQL 14.2

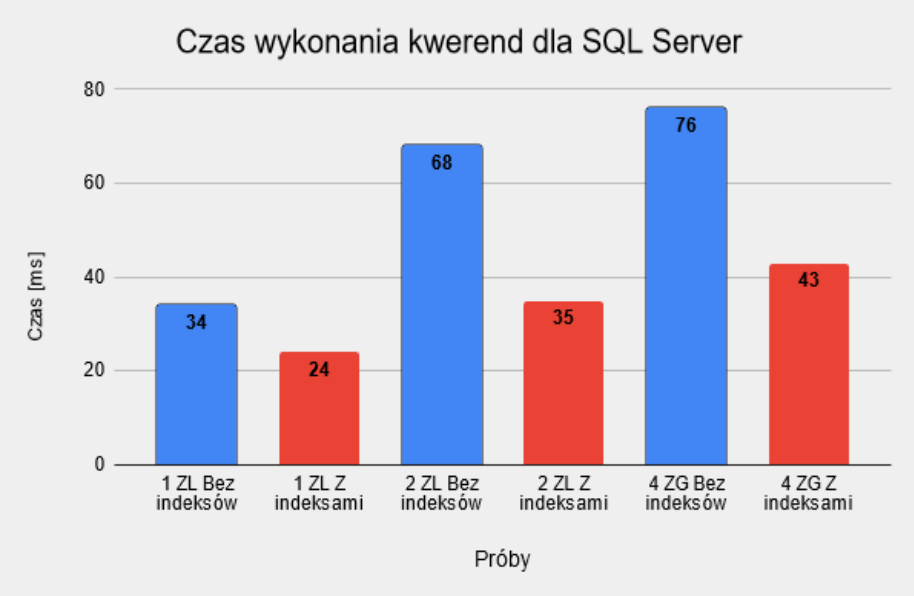
6) Wyniki testów:

Czasy wykonania zapytań 1 ZL, 2 ZL, 3 ZG i 4 ZG [ms]

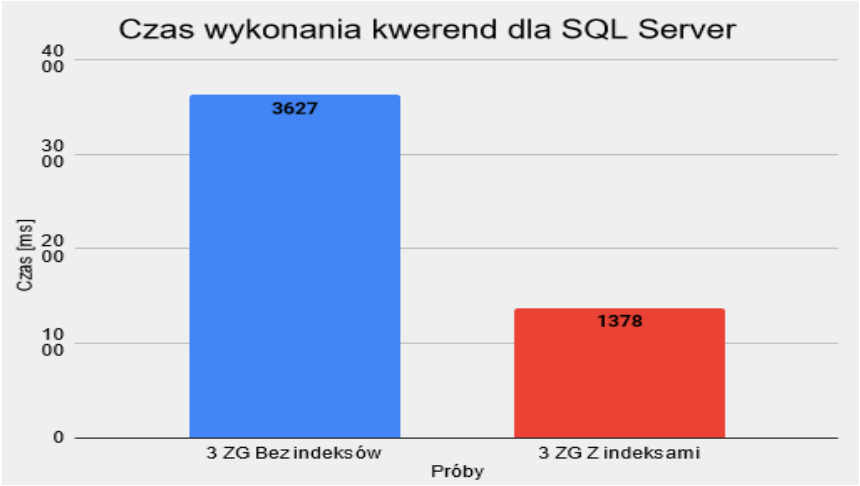
BEZ INDEKSÓW	1 ZL ŚREDNIA	2 ZL ŚREDNIA	3 ZG ŚREDNIA	4 ZG ŚREDNIA
SQL Server	34	68	3627	76
PostgreSQL	188	441	12656	229
Z INDEKSAMI				
SQL Server	24	35	1378	43
PostgreSQL	222	375	13143	307

Tabela 1 – wyniki testów

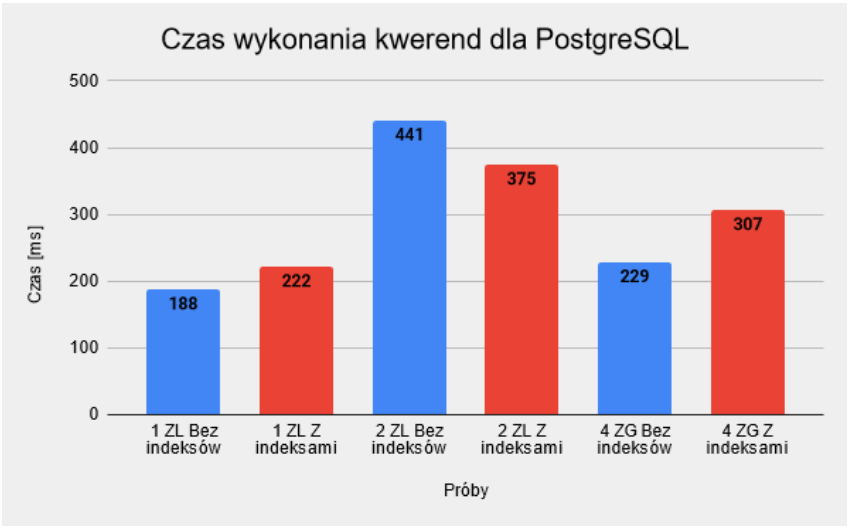
7) Wykresy:



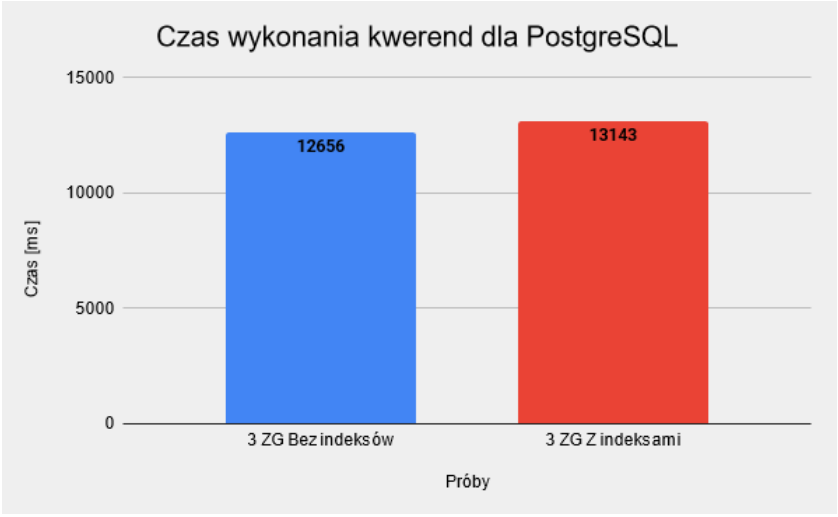
Rys 10 – Wykres przedstawiający czas wykonania kwerend dla 3 różnych zapytań w Microsoft SQL Server



Rys 11 – Wykres przedstawiający czas wykonania kwerend dla zapytania 3 ZG w Microsoft SQL Server



Rys 12 – Wykres przedstawiający czas wykonania kwerend dla 3 różnych zapytań w PostgreSQL



Rys 13 – Wykres przedstawiający czas wykonania kwerend dla zapytania 3 ZG w Microsoft SQL Server

8) Wnioski

Dla programu Microsoft SQL Server indeksowanie znacząco przyspiesza czas wykonywania kwerend. Efekt ten nie jest widoczny w PostgreSQL, a w 3 przypadkach czas ten jest nawet dłuższy, jednakże może mieć to związek z chwilowym obciążeniem komputera albo zwyczajnej losowości, która spowodowała że w PostgreSQL indeksacja nie ma aż takiego wpływu na czas wykonywania zapytań typu SELECT. Największy wpływ indeksacji jest widoczny w zapytaniu 3 ZG w Microsoft SQL Server, w którym czas jest niemal trzykrotnie krótszy.
Normalizacja miała większy wpływ jedynie przy zagnieżdżeniach skorelowanych, w obu środowiskach 3 ZG wykonywało się znacznie dłużej niż 4 ZG , a różnicą było jedynie znormalizowanie tabeli w przypadku zapytania 4 ZG.
Ważnym punktem jest to, że wszystkie zapytania w Microsoft SQL Server wykonywały się szybciej niż w PostgreSQL i to kilkukrotnie szybciej. Analiza pokazała jeszcze, że zagnieżdżenia skorelowane wykonują się dłużej niż złączenia.

9) Bibliografia

Wydajność złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych — Productivity and nesting join the schemes and standardized and denormalized / Łukasz JAJEŚNICA, Adam PIÓRKOWSKI.

https://pl.wikipedia.org