SPRAWOZDANIE

Celem ćwiczenia 9 było przeprowadzenie eksperymentu polegającego na sprawdzeniu wydajności złączeń i zagnieżdżeń skorelowanych w dwóch popularnych systemach zarządzania bazami danych – SQL Server oraz PostgreSQL. Zadanie zostało przeprowadzone na podstawie artykułu napisanego przez Adama Piórkowskiego oraz Łukasza Jajeśnica (Akademia Górniczo–Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej) o tytule "Wydajność złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych".

W celu przeprowadzenia eksperymentu utworzyłem bazę danych o nazwie GeoChron zawierającą dane z tabeli stratygraficznej znajdujące się w tabelach odpowiadających odpowiednim jednostkom: GeoEon, GeoEra, GeoOkres, GeoEpoka oraz GeoPietro. Do odpowiednich tabel wprowadziłem dane – nazwy oraz indeksy eonu, er, okresów, epok oraz pięter, dzięki czemu baza obrazuje przebieg historii Ziemi na podstawie następstwa procesów geologicznych i układu warstw skalnych.

Następnie, z wcześniej powstałych tabel, utworzyłem zdenormalizowaną tabelę GeoTabela zawierającą wszystkie dane z tabeli stratygraficznej:

Składnia SQL Server:

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

Składnia PostgreSQL:

```
CREATE TABLE GeoTabela AS (SELECT * FROM GeoPiętro
NATURAL JOIN GeoEpoka NATURAL JOIN GeoOkres NATURAL JOIN GeoEra
NATURAL JOIN GeoEon );
```

Dzięki tej tabeli można szybko odnaleźć wszystkie dane z tabeli stratygraficznej przy pomocy jednego zapytania.

Następnie w celu późniejszego przeprowadzenia testu wydajności złączeń i zagnieżdżeń utworzyłem tabelę o dużej ilości danych - tabela Milion. W celu jej utworzenia do bazy danych dodałem schemat Dziesięć, a w nim 6 tabel zawierające cyfry od 0 do 9:

```
INSERT INTO Dziesiec.a1 VALUES
(0,1),(1,1),(2,1),(3,1),(4,1),(5,1),(6,1),(7,1),(8,1),(9,1);
INSERT INTO Dziesiec.a2 VALUES
(0,1),(1,1),(2,1),(3,1),(4,1),(5,1),(6,1),(7,1),(8,1),(9,1);
INSERT INTO Dziesiec.a3 VALUES
(0,1),(1,1),(2,1),(3,1),(4,1),(5,1),(6,1),(7,1),(8,1),(9,1);
INSERT INTO Dziesiec.a4 VALUES
(0,1),(1,1),(2,1),(3,1),(4,1),(5,1),(6,1),(7,1),(8,1),(9,1);
INSERT INTO Dziesiec.a5 VALUES
(0,1),(1,1),(2,1),(3,1),(4,1),(5,1),(6,1),(7,1),(8,1),(9,1);
INSERT INTO Dziesiec.a6 VALUES
(0,1),(1,1),(2,1),(3,1),(4,1),(5,1),(6,1),(7,1),(8,1),(9,1);
```

Tabele te wykorzystałem do stworzenia tabeli Milion:

Składnia SQL Server:

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

Składnia PostgreSQL:

```
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, a1.cyfra AS cyfra, a1.bit AS bit
FROM Dziesiec.a1, Dziesiec.a2, Dziesiec.a3, Dziesiec.a4, Dziesiec.a5, Dziesiec.a6;
```

Poniższe testy zostały wykonane przy pomocy komputera o parametrach:

CPU: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz

RAM: Pamięć DDR4 32 GB

SSD: 953 GB

S.O.: Windows 11 Pro

Jako systemy zarządzania bazami danych wybrano oprogramowanie wolno dostępne:

SQL Server management studio v18.11.1

PostgreSQL 14(pgAdmin 6.10)

Następnie wykorzystałem kilka zapytań mających na celu sprawdzenie wydajności złączeń i zagnieżdżeń z wykorzystaniem uprzednio utworzonych tabel w obrębie tabeli stratygraficznej oraz tabeli Milion:

Zapytanie 1 (1 ZL):

Składnia SQL Server:

```
SET STATISTICS TIME ON;
SELECT COUNT(*) FROM Milion  INNER JOIN GeoTabela ON Milion.liczba%68 =
GeoTabela.id_pietro;
SET STATISTICS TIME OFF;
```

Składnia PostgreSQL:

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

Zapytanie 2 (2 ZL):

Składnia SQL Server:

```
SET STATISTICS TIME ON;
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 GeoOkres.id_era = GeoEra.id_era)
INNER JOIN GeoEon ON GeoEra.id_eon = GeoEon.id_eon );
SET STATISTICS TIME OFF;
```

Składnia PostgreSQL:

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

Zapytanie 3 (3 ZG):

Składnia SQL Server:

```
SET STATISTICS TIME ON;
SELECT COUNT(*) FROM Milion WHERE Milion.liczba%68 =
(SELECT id_pietro FROM GeoTabela WHERE Milion.liczba%68 =(id_pietro));
SET STATISTICS TIME OFF;
```

Składnia PostgreSQL:

```
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):

Składnia SQL Server:

```
SET STATISTICS TIME ON;
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 GeoOkres.id_era = GeoEra.id_era)
INNER JOIN GeoEon ON GeoEra.id_eon = GeoEon.id_eon ));
SET STATISTICS TIME OFF;
```

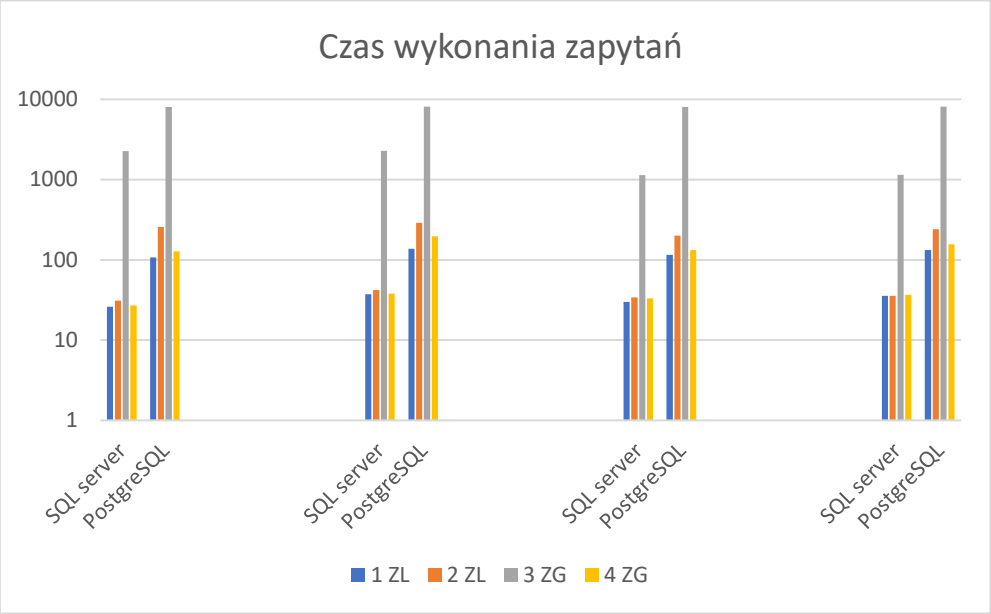
Składnia PostgreSQL:

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

Tabela wyników pomiarów czasu wykonanych zapytań (czas wykonania zapytań wyrażony jest w [ms]).

	1 ZL		2 ZL		3 ZG		4 ZG	
Bez Indeksów	Minimum	Średnia	Minimum	Średnia	Minimum	Średnia	Minimum	Średnia
SQL Server	26	37,2	31	42	2271	2296,8	27	38
PostgreSQL	107	137	258	288,2	8049	8134,6	128	197
Z Indeksami	Minimum	Średnia	Minimum	Średnia	Minimum	Średnia	Minimum	Średnia
SQL Server	30	35,8	34	35,8	1142	1145,2	33	36,6
PostgreSQL	116	133	201	240,8	8049	8093,6	133	157

Histogram z wynikami pomiarów (pierwsze dwa wykresy słupkowe to minimum bez indeksów, drugie to średnie bez indeksów, trzecie to minimum z indeksami, a czwarte to średnie z indeksami):



Powyższy wykres przedstawiony jest w skali logarytmicznej dla lepszego zobrazowania wyników.

Podsumowując, czas wykonania zapytań jest wyraźnie mniejszy w przypadku systemu SQL Server niż w PostgreSQL, natomiast dodanie indeksów zwiększa wydajność w obu przypadkach. Różnice w czasie wykonania zapytań na wykresie są widoczne najwyraźniej dla zapytania trzeciego, lecz po spojrzeniu do tabeli widać różnice w reszcie zapytań.