WSTĘP

Celem zadania jest zmierzenie czasu wykonywania się zapytań bazujących na złączeniach i zagnieżdżeniach dla tabeli geochronologicznej. Testy przeprowadzono dla tabel z indeksami oraz bez indeksów w MySQL oraz PostgreSQL na jednym urządzeniu.

KONFIGURACJA SPRZĘTOWA I PROGRAMOWA

KOMPUTER:

- CPU: Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz
- GPU: Intel(R) UHD Graphics
- RAM: 8 GB
- SSD: WDC PC SN503 SDBPNPZ-512G-1027
- OS: Microsoft Windows 11 64 bitowy

UŻYTE SERWERY BAZ DANYCH:

- PostgreSQL 15.2
- MySQL 8.0.33

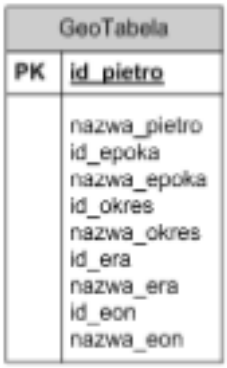
UŻYTE ŚRODOWISKA:

- MySQL Workbench 8.0 CE
- DataGrip 2023.1.2

TABELA GEOCHRONOLOGICZNA

Baza składa się z 5 tabel o nazwach: eon, era, okres, epoka, piętro w wariancie znormalizowanym (Rys. 1.), a w wariancie zdenormalizowanym (Rys. 2.) dodana została tabela stworzona poprzez złączenie wewnętrzne wszystkich tabel.

Rys. 1. Znormalizowany schemat tabeli geochronologicznej.S



Rys. 2. Zdenormalizowny schemat tabeli geochronologicznej

TESTY WYDAJNOŚCI

Na początku stworzono tabelę dziesięć, wypełnioną liczbami od 0 do 9, na której podstawie stworzono tabelę milion wypełnioną kolejnymi liczbami naturalnymi od 0 do 999999:

```
SELECT
a1.wartosc+10*a2.wartosc+100*a3.wartosc+1000*a4.wartosc+10000*a5.warto
sc+100000*a6.wartosc AS liczba
INTO liczby.milion
FROM liczby.dziesiec a1, liczby.dziesiec a2, liczby.dziesiec a3,
liczby.dziesiec a4, liczby.dziesiec a5, liczby.dziesiec a6;
```

Następnym etapem było wykonanie 4 zapytań po 10 razy i zmierzenie czasu ich wykonania z nałożonym indeksem i bez indeksu:

- Zapytanie 1 ZL, którego zadaniem jest złączenie tablicy milion z zdenormalizowaną tabelą geochronologiczną, z warunkiem złączenia modulo, odpowiadającemu liczbie pięter w tabeli (tu 68):

```
SELECT COUNT(*) FROM liczby.milion INNER JOIN geo.tabela ON
(mod(milion.liczba,68)=(geo.tabela.id_pietra));
```

- Zapytanie 2 ZL, który łączy tablicę milion z znormalizowaną tabelą geochronoliczną, poprzez łączenie pięciu tabel:

```
COUNT(*) FROM liczby.milion INNER JOIN  geo.pietro  ON
mod(milion.liczba,68)=geo.pietro.id_pietra) NATURAL JOIN
geo.epoka NATURAL JOIN geo.okres NATURAL JOIN geo.era
NATURAL JOIN geo.eon;
```

- Zapytanie 3 ZG, który poprzez zagnieżdżenie skorelowane łączy tabelę milion z zdenormalizowaną tabelą geochronologiczną:

```
SELECT COUNT(*) FROM liczby.milion WHERE
mod(milion.liczba,68)= (SELECT id_pietra FROM geo.tabela
WHERE mod(milion.liczba,68)=(id_pietra));
```

- Zapytanie 4 ZG, którego celem jest złączenie tablicy milion z znormalizowaną tabelą geochronologiczną, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane, a zapytanie wewnętrzne jest złączeniem tabel poszczególnych jednostek geochronologicznych:

```
SELECT COUNT(*) FROM liczby.milion WHERE
mod(milion.liczba,68) IN (SELECT geo.pietro.id_pietra FROM
geo.pietro NATURAL JOIN geo.epoka NATURAL JOIN geo.okres
NATURAL JOIN geo.era NATURAL JOIN geo.eon);
```

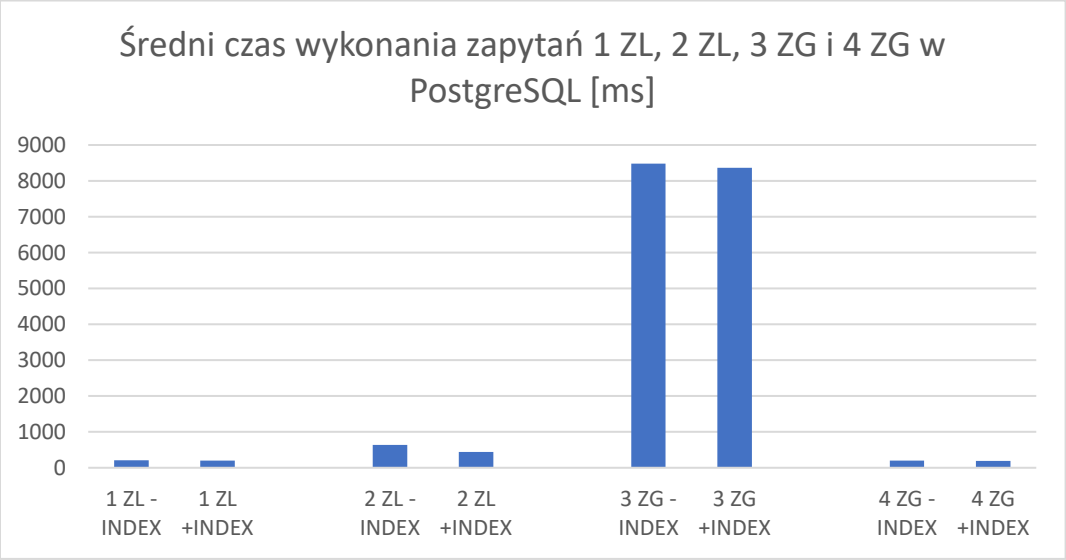
WYNIKI TESTÓW

Każdy z testów został wykonany po 10 razy. Na ich podstawie wyliczono średni czas wykonania testu, czas minimalny oraz maksymalny:

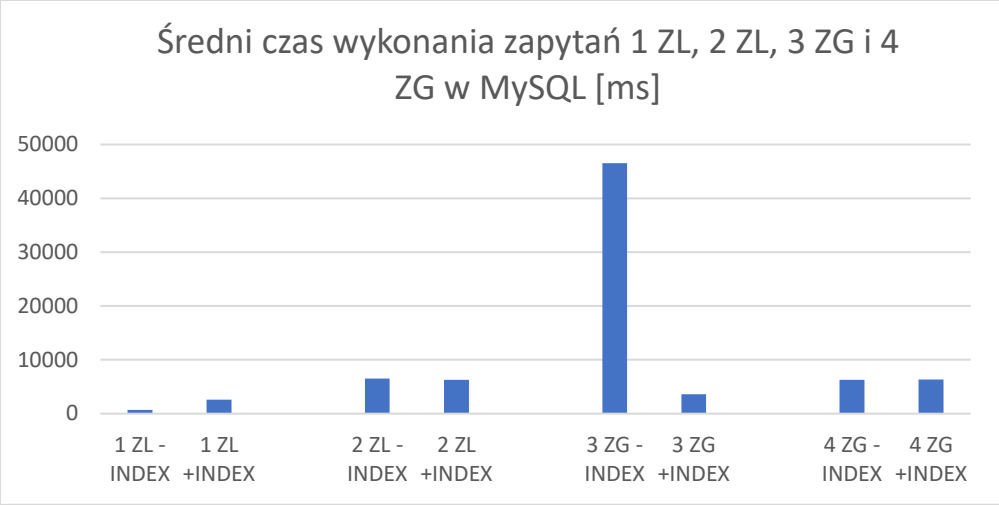
	1 ZL			2 ZL			3 ZG			4 ZG		
BEZ INDEKSÓW	**MIN**	**MAX**	**ŚR**	**MIN**	**MAX**	**ŚR**	**MIN**	**MAX**	**ŚR**	**MIN**	**MAX**	**ŚR**
MySQL	687	812	710,8	6266	7000	6501,6	45828	47219	46504,7	6078	6657	6290,7
PostgreSQL	196	228	207,6	620	649	635,8	8165	8927	8481	182	253	200,5
Z INDEKSAMI												
MySQL	2469	2766	2596,9	6125	6625	6295,4	3453	3766	3582,4	6172	6578	6314
PostgreSQL	189	220	200,2	429	487	442,6	8110	8638	8360,4	186	207	194,9

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

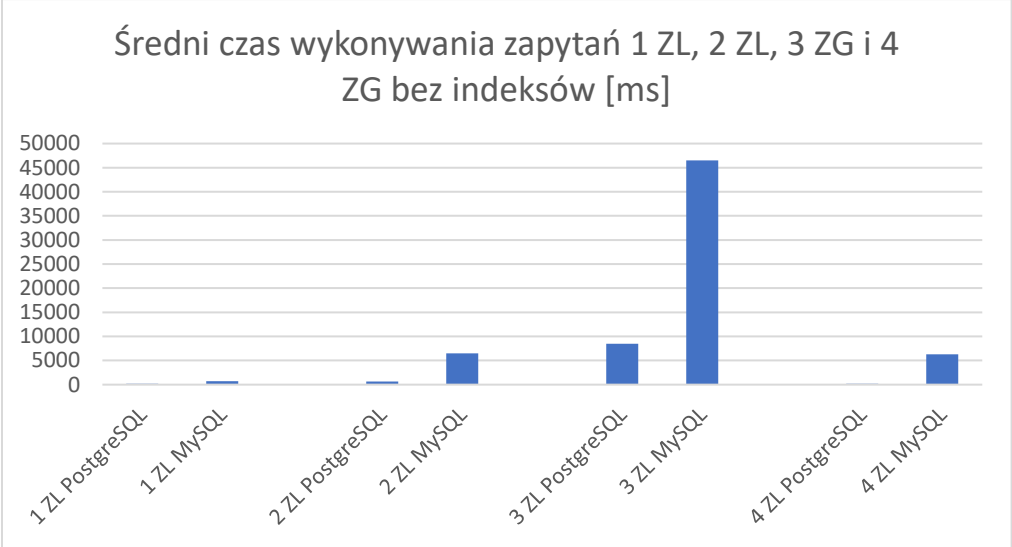
W celu graficznego porównania czasów wykonań zapytań wykonano 4 wykresy:



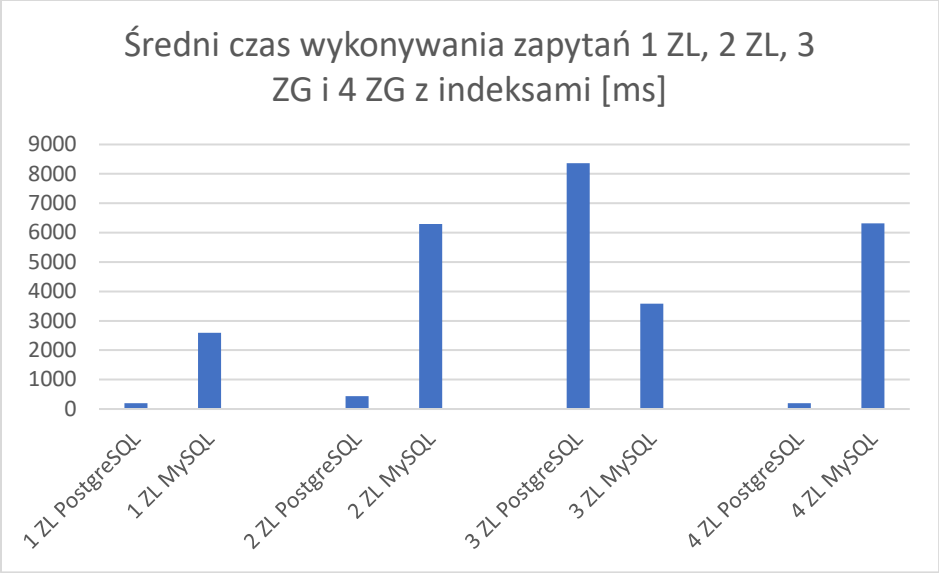
Wyk. 1. średni czas wykonania zapytań w PostgreSQL



Wyk. 2. średni czas wykonania zapytań w MySQL



Wyk. 3. Porównanie średniego czasu wykonania zapytań w MySQL i PostgreSQL bez nadanych indeksów



Wyk. 4. Porównanie średniego czasu wykonania zapytań w MySQL i PostgreSQL z nadanymi indeksami

WNIOSKI

Na podstawie wyników i wykresów można zauważyć następujące zależności:

- W PostgreSQL i MySQL nadanie indeksów nieznacznie przyspiesza wykonywanie zapytań, chociaż zapytanie 3 w MySQL wykonuje się aż 12 razy szybciej.
- Bez używania indeksów PostgreSQL jest zdecydowanie szybszy od MySQL. Z nadanymi indeksami MySQL jest szybszy tylko w wykonywaniu 3 zapytania.
- Normalizacja danych zazwyczaj obniża czas wykonywania zapytań.
- Zagnieżdżenia skorelowane są wolniejsze od złączeń.