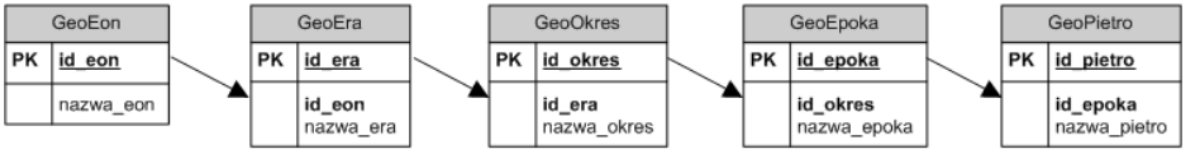
WYDAJNOŚĆ ZŁĄCZEŃ I ZAGNIEŻDŻEŃ DLA SCHEMATÓW ZNORMALIZOWANYCH I ZDENORMALIZOWANYCH

1. Wprowadzenie

Wykonany projekt miał na celu sprawdzeniu różnic w wydajności pomiędzy znormalizowanymi i zdenormalizowanymi tabelami przed i po nałożeniu na nie indeksów dla systemów zarządzania bazami danych MySQL i PostgreSQL.

2. Konstrukcja

Na potrzeby testów zostało stworzonych kilka tabel znormalizowanych symulujące tabele geochronologiczną. W każdej z nich znalazł się osobny wymiar czasowy (eon, era, okres, epoka i piętro).



rys. 1

Za pomocą połączenia naturalnego stworzono także zdemoralizowaną tabele geochronologiczną obejmująco wszystkie wyżej wymienione tabele.
Użyto do tego następującego polecenia:
*CREATE TABLE GeoTabela AS (SELECT * FROM GeoPietro NATURAL JOIN GeoEpoka NATURAL JOIN GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon);*

Utworzenie tabeli GeoTabela umożliwia szybki dostęp do wszystkich danych tabeli geo-chronologicznej za pomocą jednego zapytania prostego, co nie jest możliwe w przypadku schematu znormalizowanego opisanego powyżej rys.1.

3. Informacje sprzętowe

Nazwa systemu operacyjnego: *Microsoft Windows 10 Home*
Procesor: *Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz, 1498 MHz, Rdzenie: 4*
Zainstalowana pamięć fizyczna (RAM): *16,0 GB*
SSD: NVMe KIOXIA 512 GB

Systemy zarządzania bazami danych:
PostgreSQL: 13.2
MySQL: 8.0.25

4. Kryteria testów

- 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 połączeniu.Zasadniczym celem testów była ocena wpływu normalizacji na zapytania złożone –złączenia i zagnieżdżenia (skorelowane) [14]. W tym celu zaproponowano cztery zapytania (dla lepszej przejrzystości w sprawozdaniu ucięte zostały nazwy schematu) :

- 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,102)=(GeoTabela.id_pietro));

- Zapytanie 2 (2 ZL), którego celem jest złączenie syntetycznej tablicy miliona wyników ztabelą geochronologiczną w postaci znormalizowanej, reprezentowaną przez złączenia pięciu tabel:
  SELECT COUNT(*) FROM Milion INNER JOIN GeoPietro ON (mod(Milion.liczba,102)=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 ztabelą geochronologiczną w postaci zdenormalizowanej, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane:

SELECT COUNT(*) FROM Milion WHERE mod(Milion.liczba,102)= (SELECT id_pietro FROM GeoTabela WHERE mod(Milion.liczba,102)=(id_pietro));

- Zapytanie 4 (4 ZG), którego celem jest złączenie syntetycznej tablicy miliona wyników ztabelą geochronologiczną w postaci znormalizowanej, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane, a zapytanie wewnętrzne jest złączeniem tabel poszczególnych jednostek geochronologicznych:
  (w podstawowej wersji w miejscu 'IN' znajdowało się '=' lecz zostało ono zamienione ze względu na występujący błąd)
  SELECT COUNT(*) FROM Milion WHERE mod(Milion.liczba,102) IN (SELECT GeoPietro.id_pietro FROM GeoPietro NATURAL JOIN GeoEpoka NATURAL JOIN GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon;

5. Wyniki

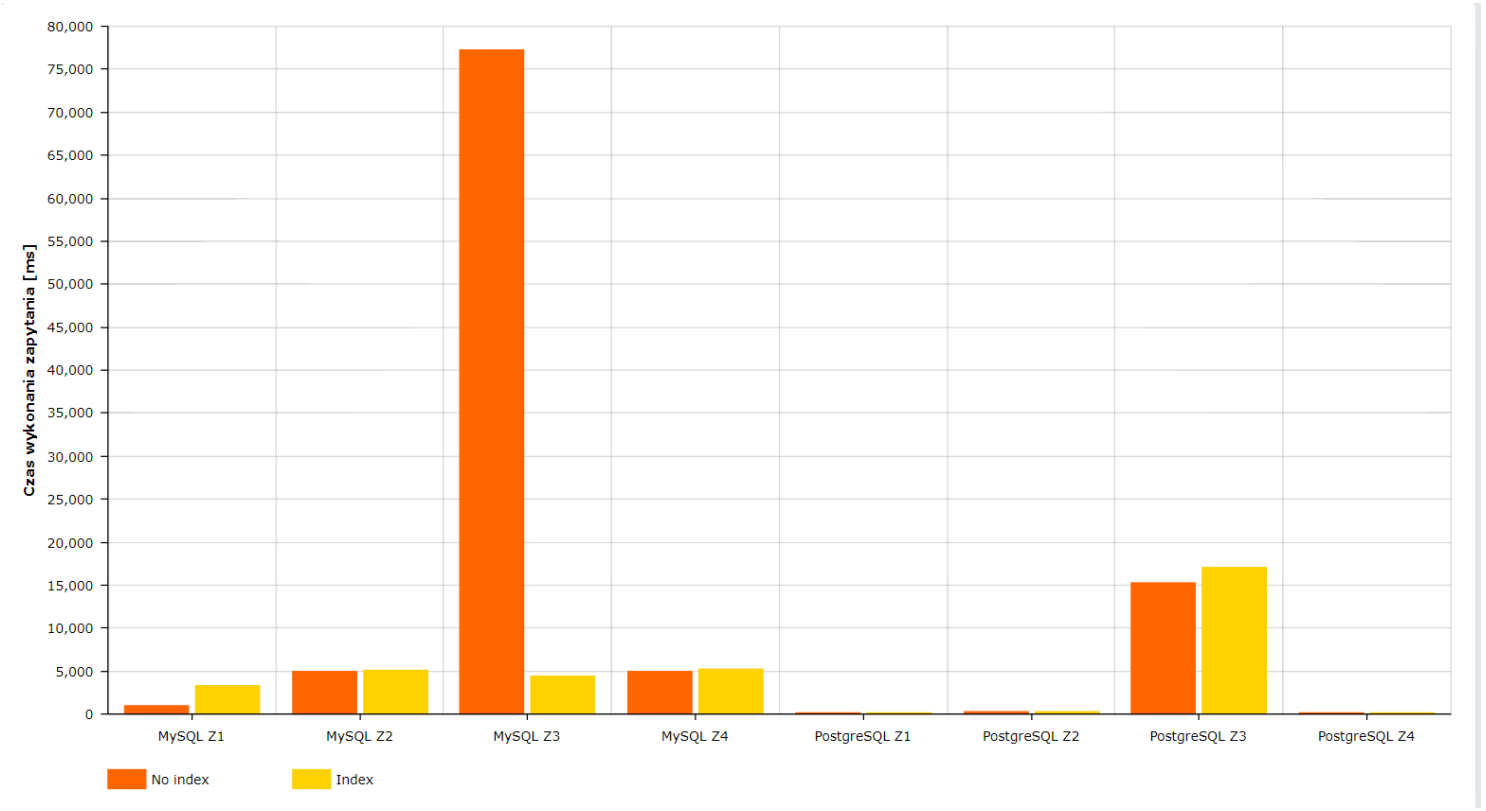
Każdy test przeprowadzono 15 razy, przy czym wyniki z błędami grubymi zostały powtórzone.

Tabela 1

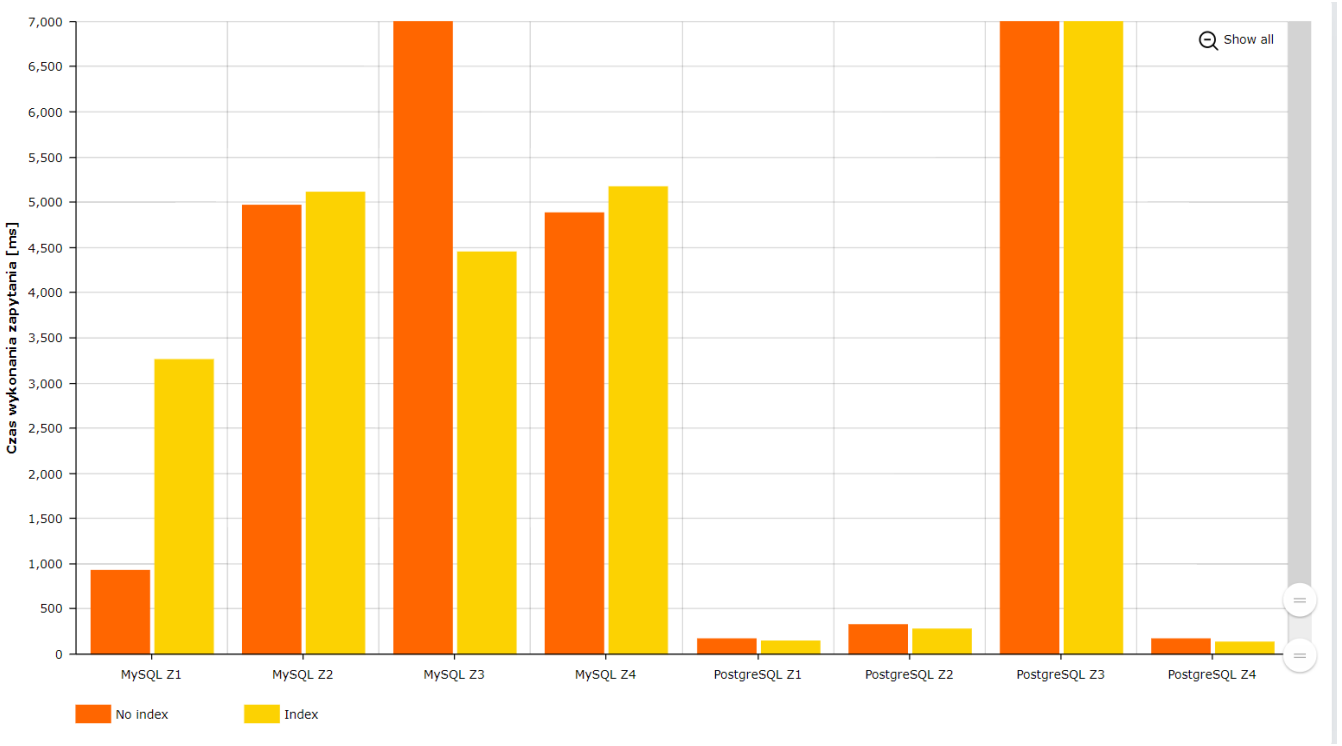
Czasy wykonania zapytań 1 ZL, 2 ZL, 3 ZG i 4 ZG [ms]

	Z1		**Z2**		**Z3**		**Z4**	
BEZ INDEKSÓW	**MIN [ms]**	**ŚR [ms]**	**MIN [ms]**	**ŚR [ms]**	**MIN [ms]**	**ŚR [ms]**	**MIN [ms]**	**ŚR [ms]**
PostgreSQL	*151*	*169,1*	*303*	*327,3*	*14151*	*15292,2*	*159*	*173,7*
MySQL	*782,0*	*929,2*	*4609,0*	*4970,7*	*76093,0*	*77309,0*	*4578,0*	*4881,3*
Z INDEKSAMI	**MIN [ms]**	**ŚR [ms]**	**MIN [ms]**	**ŚR [ms]**	**MIN [ms]**	**ŚR [ms]**	**MIN [ms]**	**ŚR [ms]**
PostgreSQL	*126,0*	*145,2*	*237,0*	*279,1*	*16576,0*	*17089,5*	*111,0*	*128,7*
MySQL	*2860,0*	*3259,6*	*4656,0*	*5110,5*	*4093,0*	*4446,7*	*4891,0*	*5169,8*

Wyniki zostały również przedstawione na wykresach Rys. 2 oraz Rys. 3 który ze względu na duże rozbieżności wyniku został przedstawiony w częściowej skali liniowej.



Rys. 2



Rys. 3

6. Wnioski

- Zagnieżdżone zapytanie 3 w każdym z systemów zajmuje znacznie większą ilość czasu niż pozostałe.
- Nałożenie indeksów w PostgreSQL tylko w nieznacznym stopniu wpływa na czas wykonania zapytań.
- W MySQL postać zdenormalizowana w większości przypadków jest szybsza, zaś w PostgreSQL różnice są minimalne ciężko ocenić wpływ indeksów na zapytania.
- Ogromna różnica występuje dla zapytania 3 w MySQL gdzie czas wykonania zapytania został zmniejszony ponad 15-krotnie.
- W każdej z prób system zarządzania bazami danych PostgreSQL wykazał się znacznie lepszą wydajnością.

Podsumowaniem rozważań jest wniosek, iż normalizacja w większości przypadków prowadzi do spadku wydajności, jednakże nie należy zapominać o jej pozytywach jakimi są przejrzystość, łatwa konserwacja oraz porządek jaki za nią idzie.