AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ GEOLOGII, GEOFIZYKI I OCHRONY ŚRODOWISKA

Sprawozdanie z ćwiczenia 10 – Bazy danych

Autor: *Alicja Ryś (415053)*
Kierunek studiów: Geoinformatyka

Kraków, 7 czerwca 2024

Spis treści

Rozdział 1. Wstęp

Niniejsze sprawozdanie powstało w ramach ćwiczenia zaliczeniowego, którego celem jest zbadanie wydajności złączeń i zagnieżdżeń skorelowanych dla schematów znormalizowanych i zdenormalizowanych. Ćwiczenie wykonano na podstawie artykułu *„Wydajność złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych”* autorstwa Łukasza Jajeśnicy i Adama Piórkowskiego. W artykule zostały omówione dwa najpopularniejsze schematy hurtowni baz danych: schemat płatka śniegu (znormalizowany) i schemat gwiazdy (zdenormalizowany).

Testy zostały przeprowadzone przy użyciu systemów zarządzania bazami danych MySQL oraz PostgreSQL.

Celem tych eksperymentów było m.in. określenie, który ze schematów - znormalizowany czy zdenormalizowany - zapewnia lepszą wydajność dla określonych zapytań.

Rozdział 2. Materiały

W celu przeprowadzenia eksperymentów, wykorzystano tabelę geochronologiczną, obrazującą przebieg historii Ziemi na podstawie procesów i warstw skalnych. W tabeli przedstawiono taksonomię dla czterech jednostek geochronologicznych - eonu, ery, okresu i epoki.

Tabela geochronologiczna

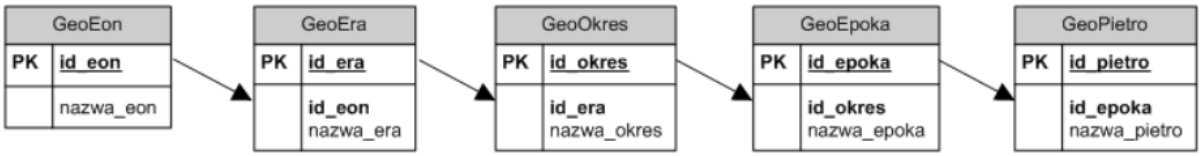
Wiek (mln lat)	Eon	Era	Okres		Epoka
0,010	FANEROZOIK	Kenzoik	Czwar-torząd		Halocen
1,8					Plejstocen
22,5			Trzeciorząd	Neo-gen	Pliocen
					Miocen
65				Paleogen	Oligocen
					Eocen
					Paleocen
140		Mezozoik	Kreda		Górna
					Dolna
195			Jura		Górna
					Środkowa
					Dolna
230			Trias		Górna
					Środkowa
					Dolna
280		Paleozoik	Perm		Górny
					Dolny
345			Karbon		Górny
					Dolny
395			Dewon		Górny
					Środkowy
					Dolny

Tabela 1. Tabela geochronologiczna

W celu przeprowadzenia eksperymentów stworzono następujące tabele:

1) znormalizowana tabela geochronologiczna,
2) zdenormalizowana tabela geochronologiczna,
3) tabela wypełniona liczbami od 0 do 9,
4) tabela wypełniona liczba naturalnymi od 0 do 999 999.

W pierwszej kolejności utworzono tabelę geochronologiczną na podstawie poniższego znormalizowanego schematu tabeli geochronologicznej. Postanowiono zrezygnować z ostatniej jednostki – piętra, z uwagi na obszerność danych.



Rysunek 1. Znormalizowany schemat tabeli geochronologicznej

Zapytania wykonano wykorzystując system zarządzania bazami danych MySQL. Utworzono 4 oddzielne tabele:

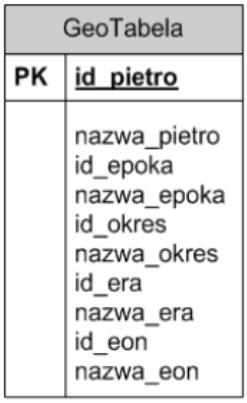
```
CREATE TABLE GeoEon (
      id_eon INTEGER NOT NULL PRIMARY KEY,
    nazwa_eon VARCHAR(15) NOT NULL
);
CREATE TABLE GeoEra (
      id_era INTEGER NOT NULL PRIMARY KEY,
      id_eon INTEGER NOT NULL,
      nazwa_era VARCHAR(15) NOT NULL,
      FOREIGN KEY (id_eon) REFERENCES geo.GeoEon(id_eon)
);
CREATE TABLE GeoOkres(
      id_okres INTEGER NOT NULL PRIMARY KEY,
      id_era INTEGER NOT NULL,
            FOREIGN KEY (id_era) REFERENCES geo.GeoEra(id_era),
    nazwa_okres VARCHAR(15) NOT NULL
);
CREATE TABLE GeoEpoka(
      id_epoka INTEGER NOT NULL PRIMARY KEY,
      id_okres INTEGER NOT NULL,
            FOREIGN KEY (id_okres) REFERENCES GeoOkres(id_okres),
    nazwa_epoka VARCHAR(15) NOT NULL
);

CREATE TABLE GeoPietro(
      id_pietro INTEGER NOT NULL PRIMARY KEY,
      id_epoka INTEGER NOT NULL,
            FOREIGN KEY (id_epoka) references GeoEpoka(id_epoka),
    nazwa_pietro VARCHAR(15) NOT NULL
);
```

Następnie uzupełniono schemat danymi z tabeli ***Błąd! Nie można odnaleźć źródła odwołania.*** i wyświetlono jako jedną tabelę, w celu sprawdzenia poprawności wykonania schematu.

	Eon	Era	Okres	Epoka
1	Fanerozoik	Kenozoik	Czwartorzęd	Halocen
2	Fanerozoik	Kenozoik	Czwartorzęd	Plejstocen
3	Fanerozoik	Kenozoik	Neogen	Pliocen
4	Fanerozoik	Kenozoik	Neogen	Miocen
5	Fanerozoik	Kenozoik	Paleogen	Oligocen
6	Fanerozoik	Kenozoik	Paleogen	Eocen
7	Fanerozoik	Kenozoik	Paleogen	Paleocen
8	Fanerozoik	Mezozoik	Kreda	Górna
9	Fanerozoik	Mezozoik	Kreda	Dolna
10	Fanerozoik	Mezozoik	Jura	Górna
11	Fanerozoik	Mezozoik	Jura	Środkowa
12	Fanerozoik	Mezozoik	Jura	Dolna
13	Fanerozoik	Mezozoik	Trias	Górna
14	Fanerozoik	Mezozoik	Trias	Środkowa
15	Fanerozoik	Mezozoik	Trias	Dolna
16	Fanerozoik	Paleozoik	Perm	Górny
17	Fanerozoik	Paleozoik	Perm	Dolny
18	Fanerozoik	Paleozoik	Karbon	Górny
19	Fanerozoik	Paleozoik	Karbon	Dolny
20	Fanerozoik	Paleozoik	Dewon	Górny
21	Fanerozoik	Paleozoik	Dewon	Środkowy
22	Fanerozoik	Paleozoik	Dewon	Dolny

Tabela 2. Tabela znormalizowana

Następnie zdecydowano się stworzyć tabelę opartą na schemacie gwiazdy (schemat zdenormalizowany). W tym przypadku również zrezygnowano z kolumny poświęconej piętrom. Klucz główny określono na *id_epoka.*



Rysunek 2. Zdenormalizowany schemat tabeli geochronologicznej

W celu osiągnięcia formy zdenormalizowanej, utworzono jedną tabelę GeoTabela, zawierającą wszystkie dane z powyższych tabel. Dokonano tego za pomocą złączenia naturalnego.

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

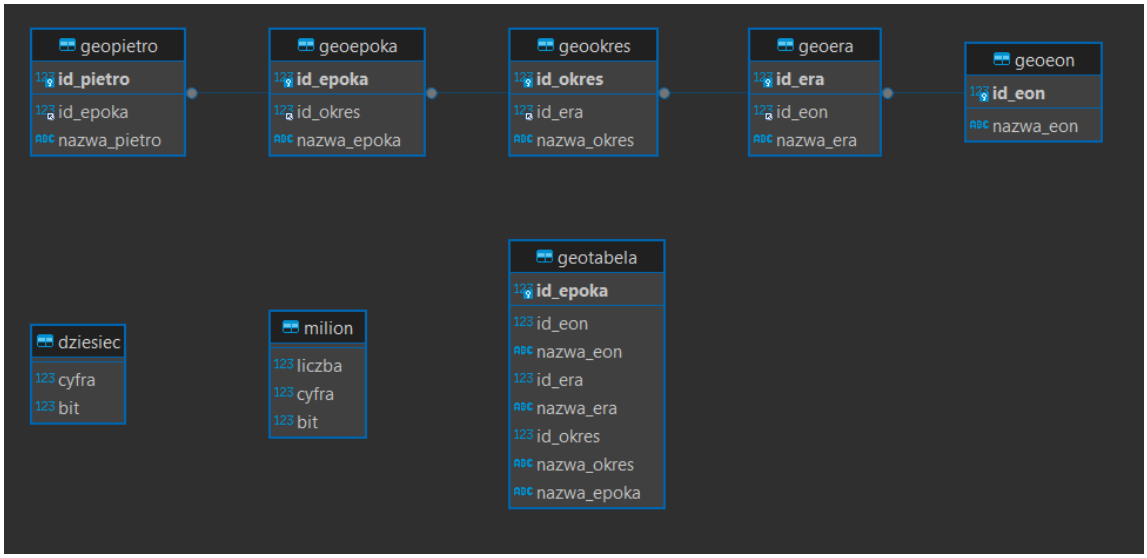
	id_eon	id_era	id_okres	id_epoka	nazwa_epoka	nazwa_okres	nazwa_era	nazwa_eon
1	1	1	1	1	Halocen	Czwartorzęd	Kenozoik	Fanerozoik
2	1	1	1	2	Plejstocen	Czwartorzęd	Kenozoik	Fanerozoik
3	1	1	2	3	Pliocen	Neogen	Kenozoik	Fanerozoik
4	1	1	2	4	Miocen	Neogen	Kenozoik	Fanerozoik
5	1	1	3	5	Oligocen	Paleogen	Kenozoik	Fanerozoik
6	1	1	3	6	Eocen	Paleogen	Kenozoik	Fanerozoik
7	1	1	3	7	Paleocen	Paleogen	Kenozoik	Fanerozoik
8	1	2	4	8	Górna	Kreda	Mezozoik	Fanerozoik
9	1	2	4	9	Dolna	Kreda	Mezozoik	Fanerozoik
10	1	2	5	10	Górna	Jura	Mezozoik	Fanerozoik
11	1	2	5	11	Środkowa	Jura	Mezozoik	Fanerozoik
12	1	2	5	12	Dolna	Jura	Mezozoik	Fanerozoik
13	1	2	6	13	Górna	Trias	Mezozoik	Fanerozoik
14	1	2	6	14	Środkowa	Trias	Mezozoik	Fanerozoik
15	1	2	6	15	Dolna	Trias	Mezozoik	Fanerozoik
16	1	3	7	16	Górny	Perm	Paleozoik	Fanerozoik
17	1	3	7	17	Dolny	Perm	Paleozoik	Fanerozoik
18	1	3	8	18	Górny	Karbon	Paleozoik	Fanerozoik
19	1	3	8	19	Dolny	Karbon	Paleozoik	Fanerozoik
20	1	3	9	20	Górny	Dewon	Paleozoik	Fanerozoik
21	1	3	9	21	Środkowy	Dewon	Paleozoik	Fanerozoik
22	1	3	9	22	Dolny	Dewon	Paleozoik	Fanerozoik

Tabela 3. GeoTabela

W celu przeprowadzenia testów wydajności złączeń oraz zapytań zagnieżdżonych, potrzebne były tabele o dużej ilości danych. Postanowiono utworzyć tabelę *Milion*, zawierającą wartości cyfra, bit i liczba, wypełnione kolejnymi liczbami naturalnymi od 0 do 999 999. Wykorzystano metodę autozłączenia tabeli *Dziesiec* wypełnionej liczbami od 0 do 9.

```
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,
```

Podsumowując, w celu wykonania testów utworzono następujące tabele:



Rysunek 3. Schemat utworzonych tabel

Wszystkie testy wykonano na komputerze o następujących parametrach:

- CPU: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz
- RAM: 32,0 GB
- SSD: SAMSUNG MZVLB512HAJQ-000H1 475GB
- S.O.: Windows 11 Pro 23H2

Jako systemy zarządzania bazami danych wybrano oprogramowanie wolnodostępne uruchomione w programie DBeaver 24.0.2:

- MySQL, wersja 8.0.37
- PostgreSQL, wersja 16.3

Rozdział 3. Metody

Celem ćwiczenia było przeprowadzenie testów, sprawdzających wydajność złączeń i zagnieżdżeń z tabelą geochronologiczną w wersji zdenormalizowanej i znormalizowanej. Procedura została przeprowadzona w dwóch etapach: bez nałożonych indeksów na kolumny danych oraz z nałożonymi indeksami na wszystkie kolumny biorące udział w złączeniu.

W celu oceny wpływu normalizacji na zapytania złożone (złączenia i zagnieżdżenia) zaproponowano cztery zapytania:

1) **Zapytanie 1 (1 ZL)** – złączenie syntetycznej tablicy *Milion* z tabelą geochronologiczną w postaci zdenormalizowanej. Dopasowano zakresy wartości złączanych kolumn dodając operację modulo.

```
SELECT COUNT(*) FROM Milion INNER JOIN GeoTabela ON
(mod(Milion.liczba,22)=(GeoTabela.id_epoka));
```

2) **Zapytanie 2 (2 ZL)** – złączenie syntetycznej tablicy *Milion* z tabelą geochronologiczną w postaci znormalizowanej, reprezentowaną przez złączenie czterech tabel:

```
SELECT COUNT(*) FROM Milion INNER JOIN GeoEpoka ON
(mod(Milion.liczba,22)= GeoEpoka.id_epoka) NATURAL JOIN
GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon;
```

3) **Zapytanie 3 (3 ZG)** – złączenie syntetycznej tablicy *Milion* z tabelą geochronologiczną w postaci zdenormalizowanej, wykorzystując złączenie poprzez zagnieżdżenie skorelowane.

```
SELECT COUNT(*) FROM Milion WHERE mod(Milion.liczba,22)=
(SELECT id_epoka FROM GeoTabela WHERE
mod(Milion.liczba,22)=(id_epoka));
```

4) **Zapytanie 4 (4 ZG)** – złączenie syntetycznej tablicy *Milion* z tabelą geochronologiczną w postaci znormalizowanej, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane, a zapytanie wewnętrzne jest złączeniem tabel poszczególnych jednostek geochronologicznych. W tym przypadku należało nieco zmodyfikować kod zamieszczony w artykule poprzez zmianę znaku '=' na 'IN', ponieważ w przeciwnym razie pojawiał się błąd spowodowany zwróceniem więcej niż 1 wiersza.

```
SELECT COUNT(*) FROM Milion WHERE mod(Milion.liczba,22) IN
(SELECT GeoEpoka.id_epoka FROM GeoEpoka NATURAL JOIN GeoOkres
NATURAL JOIN GeoEra NATURAL JOIN GeoEon);
```

Każde zapytanie dla każdego z etapów wykonano wielokrotnie – przeprowadzono po 10 prób. W celu uzyskania informacji o czasie wykonania poszczególnych zapytać, w MySQL skorzystano z funkcji:

```
SET profiling = 1

SHOW PROFILES;
```

Natomiast w PostgreSQL po wcześniejszym skonfigurowaniu, użyto:

```
SELECT * FROM pg_stat_statements;
```

Aby przeprowadzić badania dla etapu drugiego nałożono indeksy na wszystkie kolumny biorące udział w złączeniach.

```
CREATE INDEX idx_milion_liczba ON Milion(liczba);
```

Rozdział 4. Wyniki

Uzyskane wyniki poddano analizie, w celu omówienia następujących zagadnień:

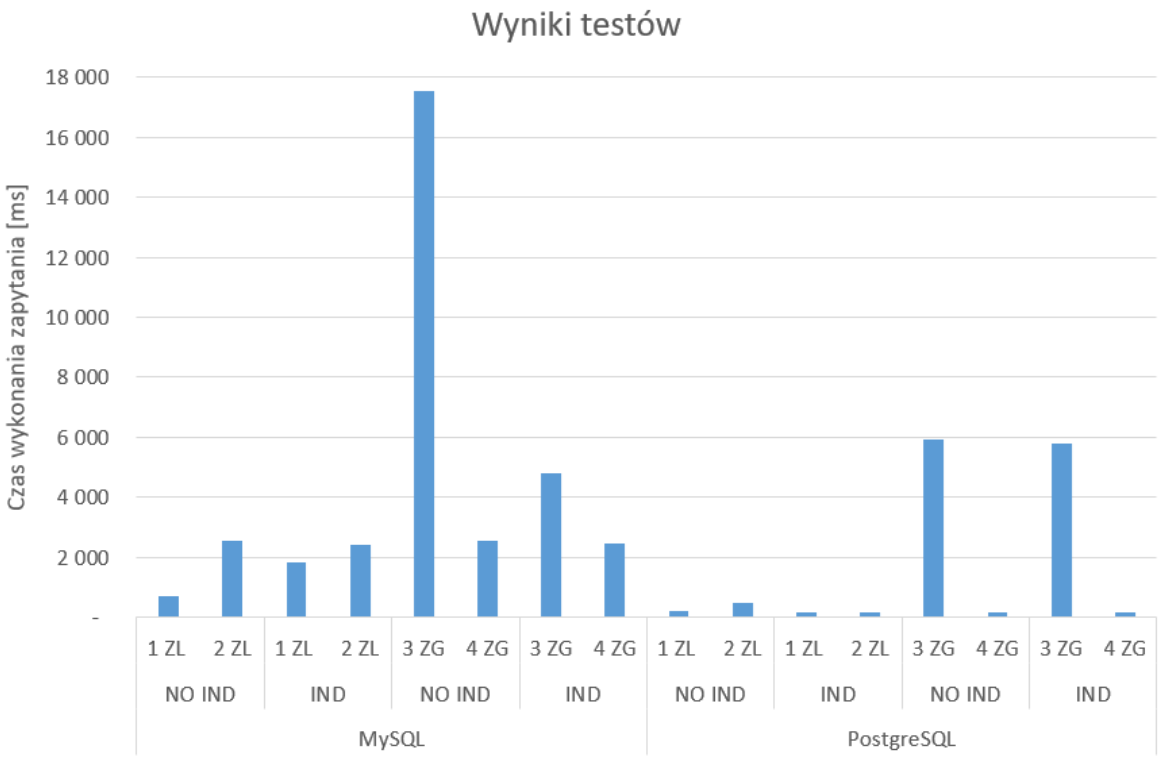
1) tempo wykonywania zapytań dla wersji znormalizowanej oraz zdenormalizowanej,
2) wpływ użycia indeksów na tempo wykonania zapytań,
3) porównanie tempa wykonywania zapytań dla złączeń oraz zagnieżdżeń skorelowanych,
4) tempo wykonywania zapytań w zależności od wykorzystywanego systemu zarządzania bazami danych: MySQL i PostgreSQL.

Wyniki testów zamieszczono w poniższej tabeli.

	1 ZL		2 ZL		3 ZG		4 ZG	
BEZ INDEKSÓW	**MIN**	**ŚR**	**MIN**	**ŚR**	**MIN**	**ŚR**	**MIN**	**ŚR**
MySQL	634	700,8	2504	2534,1	15755	17549	2519	2549,2
PostgreSQL	182	196	458	475	5747	5918	179	187
Z INDEKSAMI								
MySQL	1777	1843,3	1840	2420,8	4683	4812	2453	2483,6
PostgreSQL	180	186	159	169	5762	5797	162	172

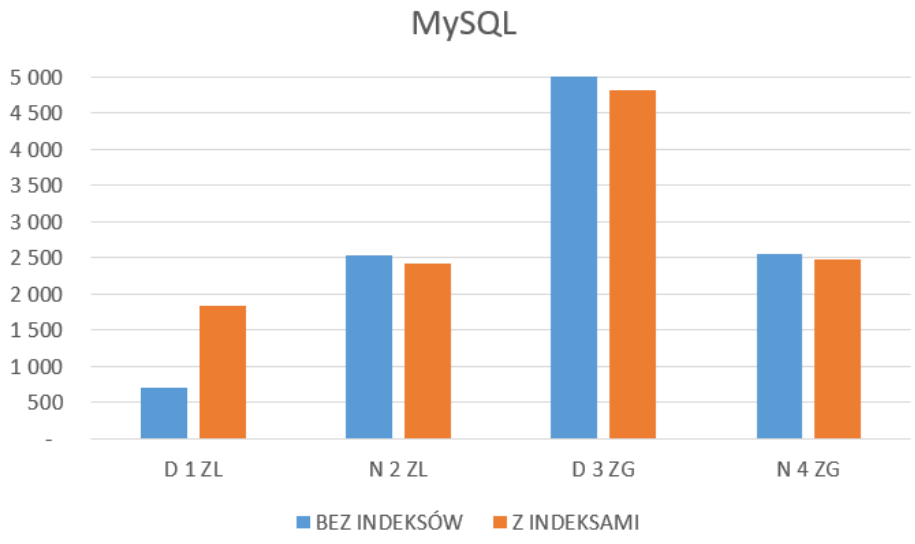
Tabela 4. Czas wykonywania zapytań [ms]

Aby ułatwić analizę stworzono wykresy. Pierwszy z nich obrazuje zbiorcze wyniki testów w ujęciu celu normalizacji.

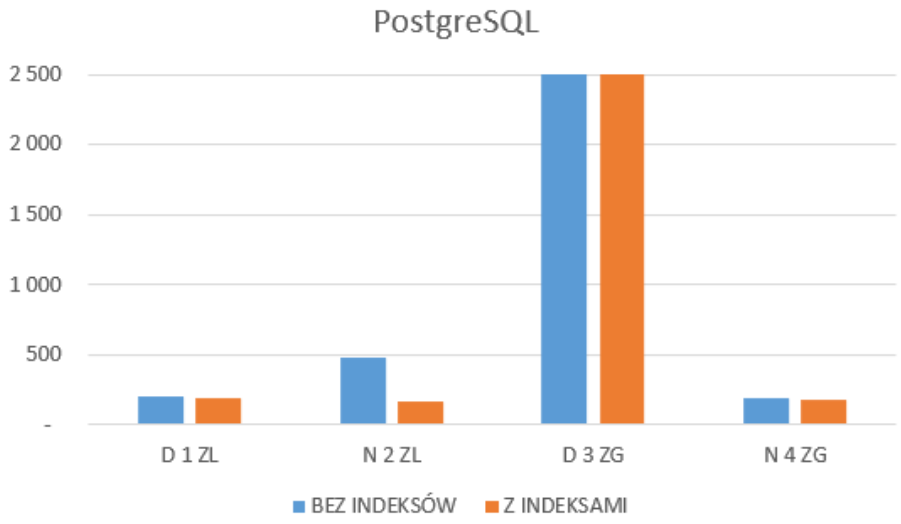


Wykres 1. Wyniki testów w ujęciu celu normalizacji

Z uwagi na rozpiętość danych utworzono dodatkowo dwa wykresy dla testów wykonanych w MySQL i PostgreSQL, zmniejszając zakres wartości na osi y. Literą „D” oznaczono wersję zdenormalizowaną, a „N” znormalizowaną.



Wykres 2. Wyniki testów w MySQL



Wykres 3. Wyniki testów w PostgreSQL

W przypadku złączeń postać znormalizowana spowodowała spadek wydajności. Jedynie w przypadku zapytania 2 ZL w PostgreSQL wprowadzenie indeksów w tabeli znormalizowanej wpłynęło na poprawienie wyniku względem postaci zdenormalizowanej.

Inaczej było dla zagnieżdżeń skorelowanych. Tam, znormalizowanie tabeli okazało się wydajniejszym rozwiązaniem.

Najwolniej wykonywało się zapytanie w postaci zdenormalizowanej z zagnieżdżeniem skorelowanym, a najszybciej postać zdenormalizowana ze złączeniem.

W większości przypadków (za wyjątkiem zapytania 1 ZL w MySQL) wprowadzenie indeksów przyśpieszyło czas wykonywania zapytań.

Porównując dwa systemy widać, że zapytania były o wiele szybciej wykonywane w PostgreSQL. Jedynym wyjątkiem jest zapytanie 3 ZG bez indeksu – MySQL okazał się szybszy.

Rozdział 5. Wnioski

Otrzymane wyniki w dużej mierzenie pokryły się z przedstawionymi w artykule, jednak wystąpiła duża rozbieżność w zapytaniu 4ZG w PostgreSQl.

Wydajność bazy danych zależy silnie od struktury tabeli i typu zapytania. Znormalizowanie tabeli nie zawsze prowadzi do lepszej wydajności, zwłaszcza w przypadku złączeń.

Indeksy dla większości przypadków poprawiają wydajność zapytań, ale może się to różnić w zależności od systemu bazy danych i konkretnego zapytania.

PostgreSQL generalnie oferuje lepszą wydajność niż MySQL w większości przypadków testowanych zapytań, co może sugerować jego przewagę w zastosowaniach wymagających wysokiej wydajności.

Rozdział 6. Spis ilustracji