Bazy Danych 1 – analiza rozwiązania projektu MojaChata.pl

W ramach analizy bazy danych, będziemy przeprowadzać różne testy optymalizacyjne na bazie danych.

1. **Wpływ wyzwalaczy na czas wykonywania poleceń –**
   Będziemy wstawiać 10000 rekordów do tabeli opinii najpierw bez włączonego wyzwalacza tg_review_text, a następnie z i zobaczymy jaki wpływ to ma na szybkość wykonania.

	Czas wstawienia rekordów [s]					AVG
	No.1	No.2	No.3	No.4	No.5	
Wyzwalacz wyłączony	2.652	2.707	2.495	2.513	2.537	2.5808
Wyzwalacz włączony	3.383	3.288	3.5134	3.371	3.386	3.3884

$$\frac{t_{wł}}{t_{wył}} = \frac{3{,}3884}{2{,}5808} = 1{,}3129$$

Podobny test przeprowadzimy dla tabeli requests, z wyzwalaczem tg_calculate_total_price.

	Czas wstawienia rekordów [s]					AVG
	No.1	No.2	No.3	No.4	No.5	
Wyzwalacz wyłączony	2.867	2.946	2.989	2.999	2.973	2.9548
Wyzwalacz włączony	3.837	3.975	3.878	3.96	3.857	3.9014

$$\frac{t_{wł}}{t_{wył}} = \frac{3{,}9014}{2{,}9548} = 1{,}3204$$

- wprowadzenie wyzwalaczy wydłuża czas działania poleceń o około 30%, jest to akceptowalna strata na rzecz funkcjonalności aplikacji

2. **Optymalizacja zapytań poprzez wprowadzenie indexów-**

   Sprawdzamy wpływ indexów na czas wstawienia 10000 rekordów oraz koszt wykonania prostego zapytania na tabeli cottages z warunkami do każdej kolumny.

-Przydzielenie indexu do każdej kolumny tabeli cottages:

	Czas wstawienia rekordów [s]					AVG	Koszt CPU zapytania
	No.1	No.2	No.3	No.4	No.5		
Brak indexów	3,318	3,337	3,363	3,344	3,363	2.5808	413
5 indexów	3,933	4,561	4,441	4,005	3,963	3.3884	294

$$\frac{t_5}{t_0} = \frac{3{,}3884}{2{,}5808} = 1{,}3129 \qquad \frac{K_5}{K_0} = 0{,}71$$

W tym przypadku wprowadzenie indexów daje nam 30% bardziej wydajne zapytania, ale tracimy na tym 30% szybkości wstawiania rekordów.

-Następnie rozważymy zapytanie łączące tabele cottages, addresses oraz countries:

```
SELECT COUNTRIES.NAME, COUNT(*) FROM COTTAGES
JOIN ADDRESSES USING (ADDRESS_ID)
JOIN CITIES USING (CITY_ID)
JOIN COUNTRIES USING (COUNTRY_ID)
GROUP BY COUNTRY_ID, COUNTRIES.NAME
order by count(*) desc;
```

Bez żadnych indexów zapytanie kosztuje procesor 478, natomiast po dodaniu indexu do klucza obcego tabeli adresów w tabeli cottage, liczba ta drastycznie maleje do 193, zwiększając wydajność.

3. **Testy wydajnościowe z dużą liczbą rekordów-**

Testujemy czas wykonania różnych zapytań dla ilości rekordów ~300000, sprawdzimy działanie zapytań z funkcji w naszej aplikacji.

Zapytanie	Czas wykonania [s]
Is_cottage_available	0,04
Calculate_total_tax	0,01
Request_status	0,01
Avg_price_in_country_with_tax	0,04

Wszystkie zapytania funkcyjne oraz zapytania testowe ze skryptu testowego wykonują się w zadowalającym czasie poniżej 2 sekund.

4. **Potencjalne problemy wynikające ze wzrostu liczby użytkowników**
   Wraz z rozwojem aplikacji użytkowników będzie przybywało, a baza będzie się zapełniać, co będzie prowadzić do wolniejszego jej działania. Problem ten może być minimalizowany poprzez dalszą optymalizację zapytań, dodawanie indexów lub partycjonowanie danych. Możliwe jest również zewnętrzne ograniczanie dostępu do bazy, żeby miało do niej dostęp na raz kontrolowana liczba osób.