Baza danych medium społecznościowego

Łukasz Fabia 272724
Mikołaj Kubś 272662
Martyna Łopianiak 272682
Piotr Schubert 272659
Projektowanie baz danych wt 18:55

24 listopada 2024

Spis treści

1 Etap 1: Faza konceptualna

1.1 Analiza świata rzeczywistego

1.1.1 Streszczenie - Zarys wymagań projektu

Celem projektu jest stworzenie bazy danych do obsługi medium społecznościowego. Ma ona przechowywać informacje o użytkownikach, ich treściach, relacjach i aktywnościach. Baza powinna być zaprojektowana w sposób wydajny i skalowalny.

1.1.2 Potrzeby informacyjne

- Rejestracja i logowanie użytkowników z różnymi poziomami dostępu.
- Publikowanie i interakcje z treściami użytkowników (posty, polubienia, komentarze).
- Zarządzanie relacjami społecznymi (znajomości).

- Przechowywanie wiadomości prywatnych i historii aktywności.
- Tworzenie konwersacji z innymi użytkownikami
- Reportowanie postów z nieodpowiednimi treśćmi
- Tworzenie i zarządzanie stronami organizacji, firm, fanpage itd.
- Tworzenie i zarządzanie wydarzeniami

1.1.3 Czynności, wyszukiwania

- Wyszukiwanie użytkowników.
- Wyszukiwanie treści po hasztagach lub słowach kluczowych.
- Filtrowanie aktywności użytkownika, np. przeglądanie polubień i komentarzy.
- Wyszukiwanie relacji (np. znajomi użytkownika, osoby obserwujące daną osobę).
- Dodawanie innych użytkowników do znajomych i interakcja z treściami - dodawanie komentarzy i reakcji
- Tworzenie postów, wydarzeń, grup
- Konwersacja grupowa, pisanie wiadomości

1.1.4 Cele projektu

S (Specific): Zaprojektowanie bazy danych dla medium społecznościowego.

M (Measurable): Baza musi być wydajna, tzn. musi być w stanie obsługiwać dużą ilość użytkowników, co najmniej 20 000.

A (Achievable): Projekt zostanie zrealizowany przy użyciu PostgreSQL. Do stworzenia struktury tabel wykorzystany zostanie mechanizm ORM (Object Relational Mapping). Na koniec baza zostanie wypełniona danymi, aby przetestować jej wydajność.

R (Relevant): Przechowywanie profili użytkowników oraz interakcje między nimi są kluczowe dla funkcjonowania medium społecznościowego.

T (Time-bound): Praca nad projektem powinna zająć 2 miesiące.

1.1.5 Zakres projektu

Multimedia: W bazie przechowywane będą wyłącznie linki do plików na zewnętrznym serwerze.

Obsługa haseł: Wszystkie hasła w bazie będą hashowane.

1.2 Wymagania funkcjonalne

Streszczenie

Użytkownikom przypisany jest jeden z tych poziomów dostępu: admin, user lub guest.

Guest(Gość)

- Może przeglądać wybrane dane.

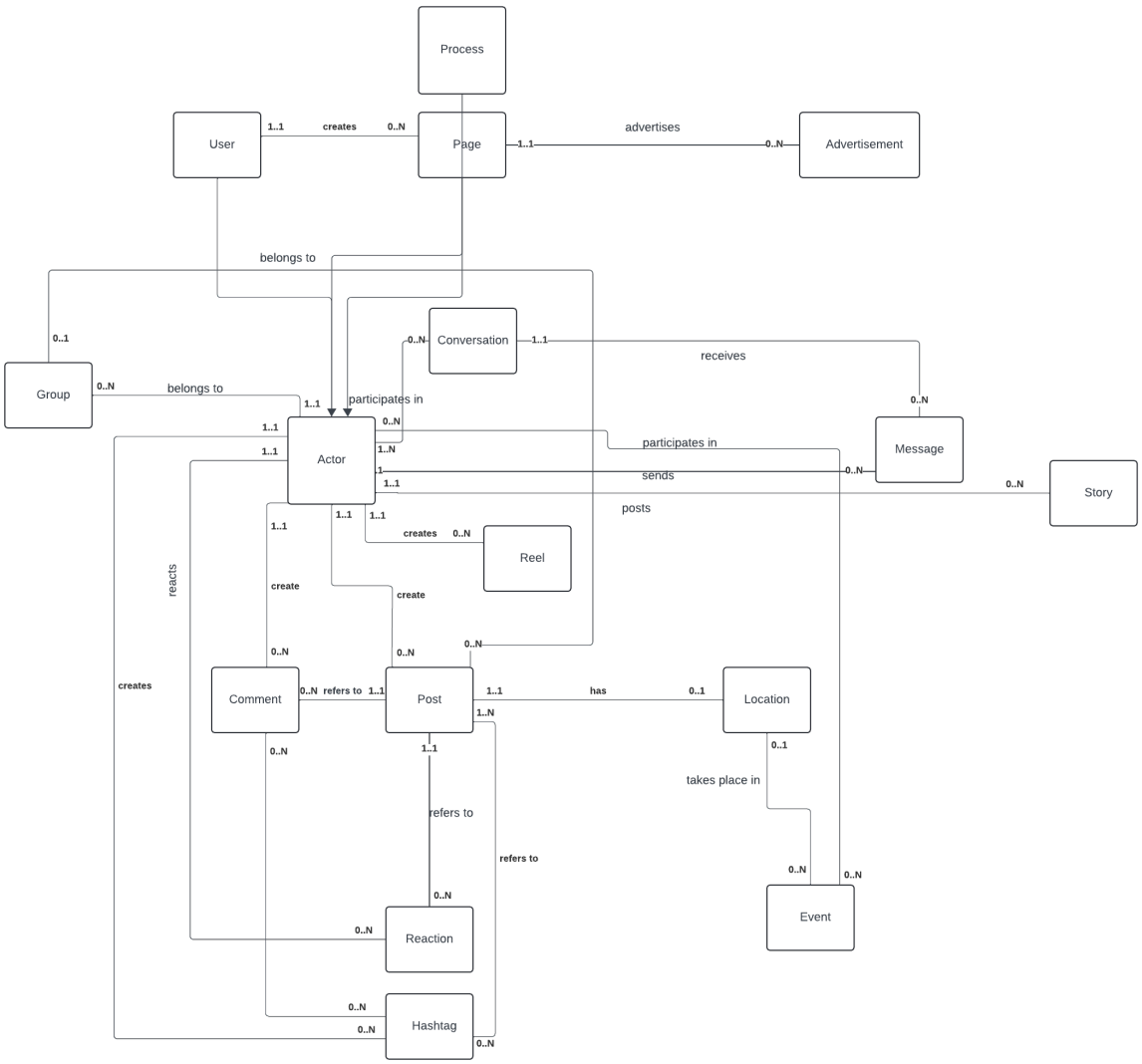
Admin

- Może przeglądać, edytować, usuwać, dodawać i przeglądać wszystkie treści, zrządza bazą i nadaje uprawnienia

User(Użytkownik)

- System umożliwia rejestrację oraz logowanie.
- Rejestracja wymaga imienia, nazwiska, daty urodzenia, hasła oraz maila.
- Logowanie wymaga maila i hasła.

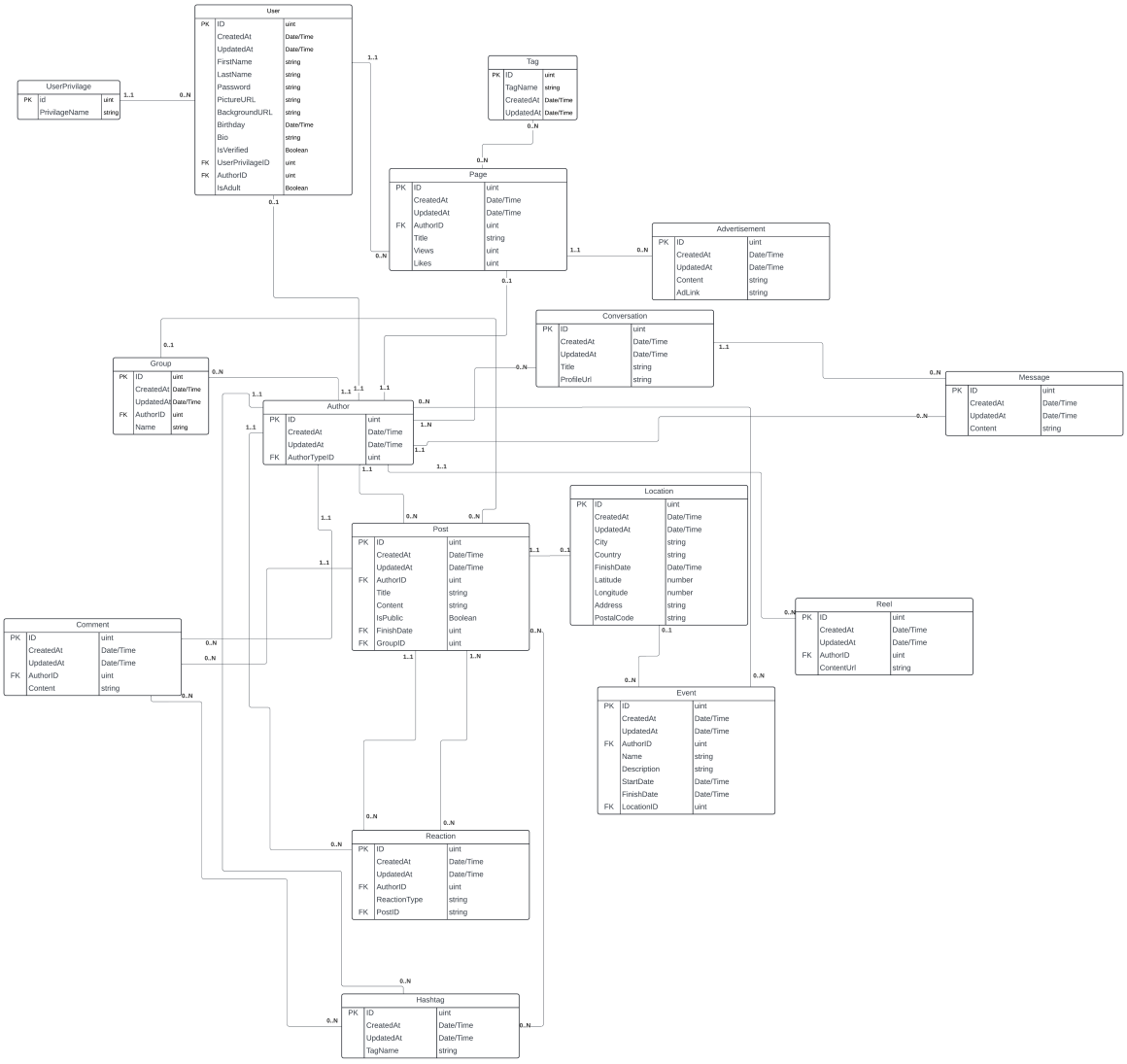
1.3 ERD



Rysunek 1: Diagram obiektowo-relacyjny

2 Etap 2: Faza logiczna

Do tabel zostały dodane atrybuty, tabele **Page** oraz **User** zostały uogólnione przez **Author**, który bierze udział w innych czynnościach. Baza została także sprowadzona do *III postaci normalniej*, przez co wydzielono klika nowych tabel.



Rysunek 2: Diagram relacji

3 Etap 3: Faza fizyczna

Encje, które uległy zmianie:

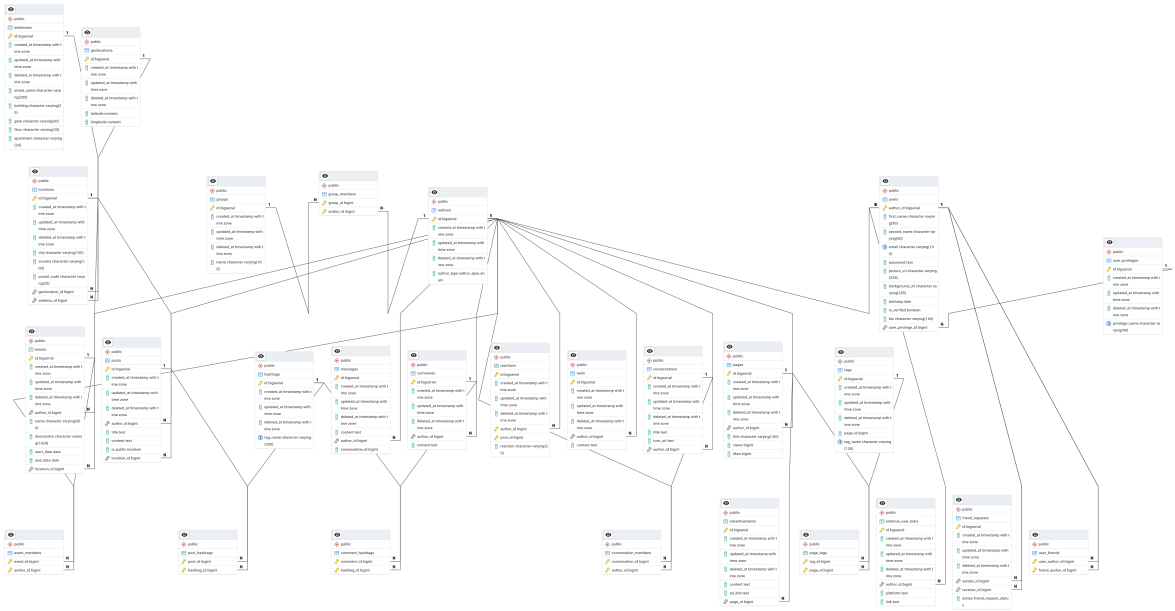
- Rozbicie lokalizacji na pomniejsze tabele
- Zamiana na enumy: typu autora oraz statusu zaproszenia do znajomych
- Dodanie do encji Conversation pole Members, które przechowuje id użytkowników biorących udział w konwersacji

Wykorzystano instrukcję CHECK do potwierdzenia poprawności danych w paru encjach. Przykładowo:

- 'A' nie moze być przyjacielem 'A'
- 'A' nie moze wysłać zaproszenia do znajomych do 'A'
- Czas rozpoczęcia wydarzenia musi być wcześniejszy niż czas zakończenia wydarzenia
- Grupa musi mieć od 1 do 10000000 członków.

Do stworzenia struktury bazy wykorzystano mechanizm ORM - (gorm). Sama baza wymagała dopracowania jeśli chodzi o enumeracje oraz reguły usuwania już bezpośrednio w systemie PostgreSQL (pgadmin).

Usuwanie: Każdy model posiada pole *DeletedAt* z indeksem. Podczas usuwania danego wiersza pole *DeletedAt* jest ustawiane na znacznik czasu (timestamp) wskazujący moment, w którym dane zostały usunięte. Dzięki temu baza obsługuje soft deleting. Oznacza to, że gdy użytkownik usunie swoje konto, będzie można je przywrócić. Taką operację obsługują na przykład serwisy takie jak Facebook.



Rysunek 3: Diagram relacji z PostgreSQL

4 Etap 4: Faza fizyczna 2

Standardowo kod można zobaczyć w repoztorium na gicie: social media db.

Mając napisany ORM poprzenio, w tym etapie pozostało napisać funkcje generujące dane do bazy. Rozwiązanie można podzielić na 3 cześci (od szczegółu do ogółu):

1. Dekorator, który będzie wykonywać określny blok `count` razy.
2. Funkcja, generująca dany typ danych np. generator użytkowników.
3. Odpowienie ułożenie wywołań.

Zapełnianie bazy danych dużą ilością danych zajmuje stosunkowo dużo czasu, może to być spowodowane przez unikalność niektórych atrybutów(biblioteka "męczy" się z generowaniem unikalnych sensownych danych), ale także przez wywołania które tworzą listy autorów dla np. eventów, ostatnim podejrzeniem może być nieoptymalnie napisay kod.

Język: Go
Generowanie sztucznych danych: gofakeit
Object relational mapping (ORM): gorm

Reszta rzeczy, które zostały wykonane:

- Funkcja usuwająca dane z tabel
- Własne implementacje niektórych danych np. `Title`, `Birthday`

- Konfiguracja loggera
- Funkcja haszująca hasło

5 Etap 5: Faza fizyczna 3

W tej fazie projektu stworzyliśmy projekt interfejsu graficznego w programie Figma oraz napisaliśmy 10 nietrywialnych kwerend w SQL.

5.1 Projekt interfejsu graficznego

Zaprojektowaliśmy następujące 10 widoków do aplikacji mobilnej:

- Strona główna
- Strona komentarzy i rekacji
- Strona profilu
- Strona strony (page)
- Strona grupy
- Strona wydarzenia
- Strona konwersacji
- Strona panelu konwersacji
- Strona rolek (reel)
- Strona ze znajomymi

Podczas projektowania zauważyliśmy parę drobnych rzeczy, które można by było poprawić - np. brak ikony czy grafiki tła dla strony. Tak więc poprawiliśmy takie nieścisłości i zaktualizowaliśmy kod seeder.go, który zajmuje się generowaniem tych danych.

Pełny projekt interfejsu graficznego znajduje się w załączonym pliku "figma.pdf".

Rysunek 4: 1 z widoków interfejsu graficznego

5.2 Kwerendy SQL

Poniżej znajdują się 10 kwerend SQL, które zostały napisane w celu przeszukania i stworzenia raportów na podstawie bazy danych:

- 10 najpopularniejszych hashtagów z ostatnich 7 dni
- Konwersacje autora
- Publiczne posty autora
- Średnia liczba znajomych
- Średnia liczba poszczególnych reakcji na posty w ostatnim miesiącu
- Wydarzenia w danym mieście (np. Saint Petersburg)
- Wydarzenia, w których wezmą udział znajomi autora
- Tagi stron, posortowane po średniej liczbie polubień i wyświetleń
- Użytkownicy, z którymi użytkownik ma wspólnych znajomych
- Użytkownicy z najwiekszą średnią liczbą reakcji na posty

	user_author_id bigint	first_name character varying (50)	second_name character varying (50)	mutualfriendscount bigint
1	2	Jazmin	Donnelly	3
2	247	Tess	Gusikowski	2
3	281	Francisca	Hoppe	2
4	154	Micheal	Jewess	2
5	326	Abby	Miller	2
6	244	Jana	Spinka	2
7	460	Lisette	Bednar	1
8	328	Darrion	Blanda	1
9	107	Danny	Boehm	1
10	441	Bernhard	Bogan	1

Rysunek 5: Wynik kwerendy wyszukującej użytkowników ze wspólnymi znajomymi

	id [PK] bigint	name character varying (300)	start_date date	end_date date	numberoffriendsattending bigint
1	5507	Seminar Civis Analytics	2024-11-18	2024-11-28	3
2	7711	Summit Robinson + Yu	2024-11-18	2024-11-24	2
3	4220	Festival Embark	2024-11-18	2024-11-21	2
4	10376	Party TransUnion	2024-11-18	2024-11-28	2
5	6989	Seminar Bekins	2024-11-18	2024-11-22	2
6	10096	Party Morgan Stanley	2024-11-18	2024-11-26	2
7	7596	Summit Weather Decision Technologies	2024-11-18	2024-11-28	2
8	11376	Party iRecycle	2024-11-18	2024-11-27	2
9	8206	Summit StreetCred Software, Inc	2024-11-18	2024-11-27	2
10	5778	Summit xDayta	2024-11-18	2024-11-28	2
11	4122	Webinar KPMG	2024-11-18	2024-11-25	2
12	153	Summit SmartProcure	2024-11-18	2024-11-21	2
13	5392	Summit Ensco	2024-11-18	2024-11-24	2
14	11168	Party StreamLink Software	2024-11-18	2024-11-26	2
15	2615	Seminar CitySourced	2024-11-18	2024-11-27	2
16	9197	Festival American Red Ball Movers	2024-11-18	2024-11-22	2
17	1941	Party Recargo	2024-11-18	2024-11-28	2
18	11513	Workshop SAS	2024-11-18	2024-11-22	2

Rysunek 6: Wynik kwerendy wyszukującej wydarzenia, na które zapisali się znajomi użytkownika

6 Etap 5: Faza fizyczna 4

Przykładowy wynik działania `EXPLAIN`

```
Seq Scan on foo  (cost=0.00..155.00 rows=10000 width=4)
```

Ogólnie można powiedzieć, że komenda do dostarcza bardzo dokładne informacje w porównaniu np. do **MySQL**.

Powyższa komenda to sposób w jaki zapytanie może zostać wykonane. W tym wypadku mamy wykonanie sekwencyjne, czyli skanujemy całą tabelę w celu znalezienia wierszy spełniających warunki w kwerendzie.

Koszt(cost) - wartość po lewej to koszt początkowy zapytania tutaj jest on równy 0.00, kolejna wartość to szacowany koszt operacji, wartość ta jest obliczana przez system bazodanowy.
Wiersze(rows) - ilość wierszy spełniająca dane kryteria.
Szerokość(width) - waga wiersza w bajtach

Zatem optymalizacja zapytań będzie polegała na minimalizacji tych "kosztów". W tym celu wykorzystuje się **indeksy**.

6.1 Różne typy indeksów i ich zastosowanie

Typ indeksu	Zastosowanie
B-Tree	Wyszukiwanie, sortowanie, zakresy, klucze główne, indeksy unikalne.
Hash	Szybkie porównania równości (=).
GiST	Dane przestrzenne, zakresy, hierarchie.
BRIN	Duże tabele z posortowanymi danymi, dane archiwalne.

6.2 Obserwacje

Indeksy nie pomogą w każdej operacji - na przykład dla kwerend, gdzie największym kosztem jest grupowanie, mogą niekoniecznie pomóc.

Hash nie przynosi znaczącej poprawy wydajności w naszych zapytaniach w porównaniu z B-tree.

Zapytania wykorzystujące funkcje agregujące są trudne do optymalizacji, a dodanie indeksów w większości przypadków nie przyniosło zauważalnych korzyści.

Wyjątek stanowi zapytanie **show_events_in_st_petersburg**, gdzie zastosowanie indeksu złożonego przyniosło wyraźną poprawę:

- Zastosowanie indeksu typu **B-tree** zmniejszyło koszt zapytania o około 10%.
- Dodanie indeksu typu **BRIN** dodatkowo obniżyło koszt o kolejne 10%.

Dla zapytań niewykorzystujących funkcji agregujących, dodanie indeksów pozwoliło na kilkukrotne zmniejszenie kosztów ich wykonania.

Rysunek 7: Przykładowe czasy zapytań bez i z indeksami.

Rysunek 8: Przykładowe czasy zapytań bez i z indeksami.

Rysunek 9: Optymalizacja zapytania z wydarzeniami w Piotrogrodzie