Dokumentacja projektu z Baz Danych: System Zarządzania Partią Polityczną

Filip Marcinek

semestr letni 2019

Specyfikacja projektu jest dostępna pod linkiem: https://github.com/piotrekiiuwr/bd2019-projekt/blob/master/projekt.md.

1 Uruchomienie programu

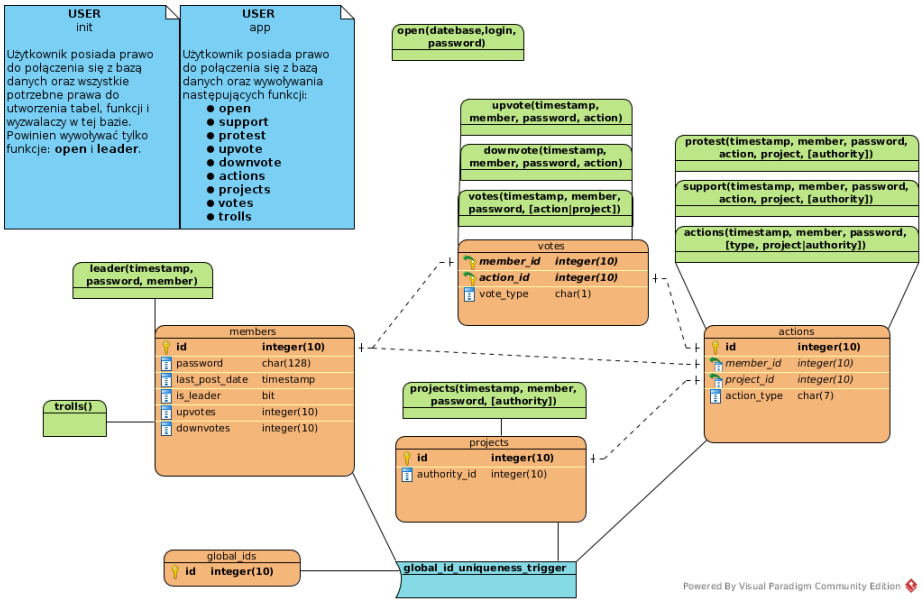
Program uruchamia się poprzez uruchomienie skryptu `run.sh` z odpowiednimi parametrami (jak w dokumentacji) oraz przekierowanie na jego standardowe wejście pliku z instrukcjami API (sformatowanymi jak w dokumentacji). Program musi być uruchomiony w tym samym katalogu, w którym znajduje się baza danych, którą obsługuje (baza danych powinna istnieć przed pierwszym uruchomieniem programu). Flagi inne niż wspomniane w dokumentacji są ignorowane przez program. Jeżeli program dostanie instrukcje API w nieprawidłowym formacie - zwróci błąd.
Przykładowe uruchomienie programu: `./run.sh --init < init-test.json`.

2 Projekt bazy danych

Projekt składa się z dwóch modułów:

- pythonowego programu, który parsuje wejście, wysyła żadania do bazy danych oraz odbiera je, przetwarza i wyświetla,
- bazy danych sql, w której zaimplementowane są wszystkie funkcjonalności API i przechowywane dane przekazywane do programu.

Rysunek 1. przedstawia projekt bazy danych, na którym możemy zobaczyć uprawnienia użytkowników, tabele oraz ogólny model triggerów i funkcji wykorzystywanych przez użytkowników. Warto zauważyć, że użytkownik `app` ma prawo tylko do wywoływania funkcji sql wymienionych w dokumentacji i nie jego kompetencje nie wykraczają poza ten obszar. Trigger `global_id_uniqueness_trigger` używany jest do zachowania warunku unikalności wszystkich id w bazie danych. Funkcje połączone są liniami z tabelami, które wykorzystują w głównej mierze.



Rysunek 1: Projekt bazy danych

3 Implementacja

Pythonowy program wykorzystuje bibliotekę `psycopg2` do łączenia się z bazą danych. Baza danych sql tworzy użytkownika `app` ze wspomnianymi wcześniej prawami oraz tabele, triggery i funkcje przedstawione ogólnie w sekcji *Projekt bazy danych*.

Unikalność wszystkich id w bazie danych jest zachowana z wykorzystaniem tabeli `global_ids`, do której za pomocą triggerów wstawiane jest każde id nowego obiektu w bazie (jeżeli któreś id się powtórzy wystąpi błąd naruszenia własności klucza w `global_ids`).

Hasła użytkowników są haszowane z pomocą modułu `pgcrypto` i przechowywane w tabeli `members`, w której znajdują się także id użytkowników, flaga `is_leader` – czy użytkownik jest liderem partii, `last_post_date` – data ostatniego posta użytkownika, liczniki `upvotes` i `downvotes` – liczby głosów (za i przeciw) oddanych na ich akcje.

Przy każdym poprawnym działaniu w polu `last_post_date` zapisywana jest data tego działania (dla funkcjonowania bazy zgodnie ze specyfikacją wystarczy przechowywanie daty ostatniego działania użytkownika).

Funkcje `support` i `protest` wstawiają akcje do tabeli `actions` z oznaczeniem 'support'/'protest' w kolumnie `action_type`, a projekty wraz z odpowiadającymi im organami władzy do tabeli `projects`.

Funkcje `upvote` i `downvote` zapisują oddane głosy w tabeli `votes` wraz z oznaczeniem 'u'/'d' (upvote/downvote) w kolumnie `vote_type`.

Pozostałe funkcje wymagają praw lidera partii (flaga `is_leader`), co jest sprawdzane przed wszelkimi modyfikacjami czy odczytami z bazy przez użytkownika (podobnie przed wykonaniem zadania funkcji API, najpierw sprawdzane jest, czy użytkownik nie jest zamrożony, poprzez porównanie aktualnej

daty z datą ostatniego działania użytkownika).

Funkcja `actions` wykorzystuje pomocniczą funkcję (dla czytelności kodu) i obsługuje wszystkie przypadki wymienione w specyfikacji. Funkcji `projects` wystarcza korzystanie z samej tylko tabeli `projects`.

Funkcja `trolls` w prosty sposób wykorzystuje kolumny `last_post_date`, `upvotes` i `downvotes` z tabeli `members`.