System zarządzania partią polityczną - dokumentacja

Patryk Kumor

10 czerwca 2019

Spis treści

1 Potrzebne pakiety i uruchomienie programu

1.1 Potrzebne pakiety

Program został napisany w Pythonie 2.7, z użyciem następujących bibliotek:

- argparse
- json
- sys
- psycopg2

Wszystkie powyższe moduły, oprócz ostatniego - psycopg2 - powinny być dostarczone wraz z podstawową dystrybucją pythona.
Aby zainstalować psycopg2 należy skorzystać z PIP - package managera do języka python, w tym celu należy wykonać polecenie:

```
\$ pip install psycopg2
```

1.2 Uruchomienie programu

Program do zarządzania partią przyjmuje na wejściu obiekty json, które są odczytywane jako ciąg wywołań funkcji API.
Program rozróżnia kilka typów wywołań:

Aby wywołać program z odczytem linii zawierających obiekty json (jeden obiekt na linię) ze standardowego wejścia w pętli:

- dla pierwszego wywołania wraz z flagą init:

  ```
  \$ python main.py --init
  ```

- dla kolejnych wowołań:

  ```
  \$ python main.py
  ```

Aby wywołać program wraz z odczytem standardowego wejścia zawartego w pliku (każda linia w pliku jest obiektem json):

- dla pierwszego wywołania wraz z flagą init:

  ```
  \$ python main.py --init < <input_file>
  ```

- dla kolejnych wowołań:

  ```
  \$ python main.py < <input_file>
  ```

Aby wywołać program wraz z odczytem zawartości pliku podanego jako argument (każda linia w pliku jest obiektem json) należy użyć flagi --f
(Program po skończoeniu odczytywania zawartości pliku przechodzi w tryb ciągłego czytania standardowego wejścia)

- dla pierwszego wywołania wraz z flagą init:

```
\$ python main.py --init --f <input_file>
```

- dla kolejnych wowołań:

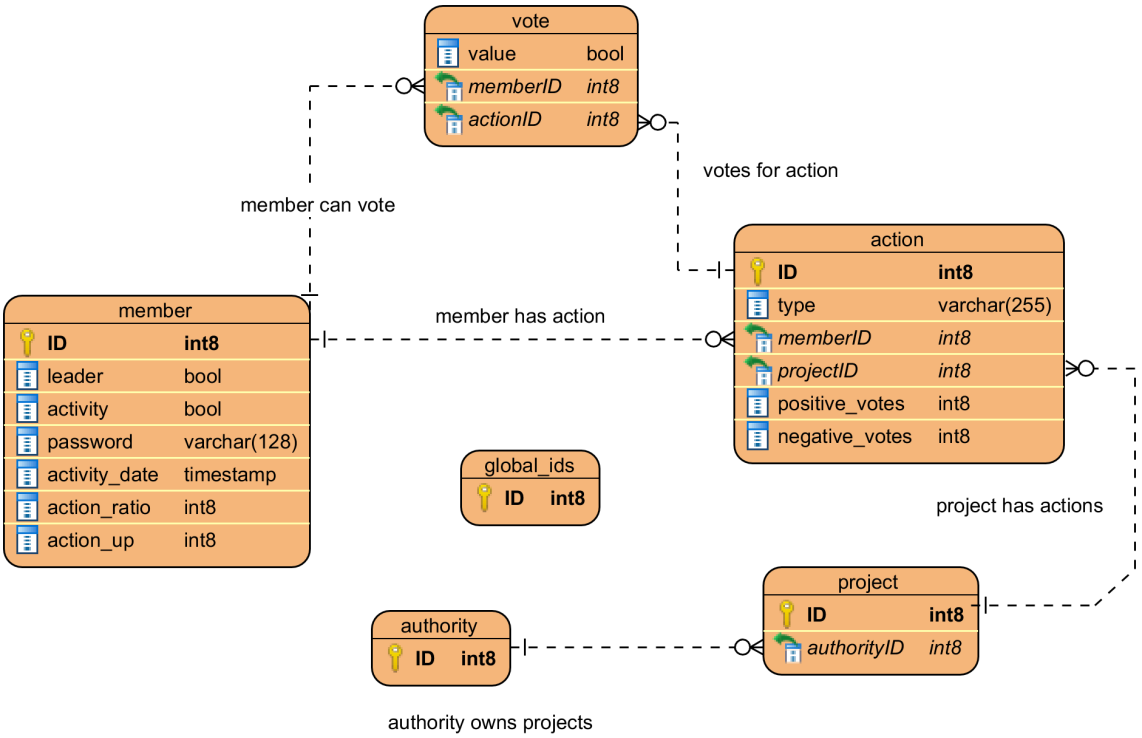
```
\$ python main.py --f <input_file>
```

Zakończenie pracy programu

Program kończy się automatycznie w przypadku odczytywania standardowego wejścia za pomocą $<$, w przypadku ciągłego odczytywania kolejnych linii program zakończy się gdy zostanie użyty skrót **ctrl**+**c**

2 Model fizyczny

Program przy swoim pierwszym uruchomieniu (wraz z argumentem --init) po pomyślnym połączeniu z bazą danych tworzy w niej wszystkie potrzebne mu elementy, które są przedstawione na poniższym rysunku.
(Cały schemat użyty podczas tworzenia zawartości bazy danych zawarty jest w pliku **base.sql**)



Użytkownik init, za pomocą którego łączymy się z bazą, posiada wszystkie uprawnienia potrzebne w programie w tym musi mieć uprawienia do: CREATE TABLE, ADD CONSTRAINT, CREATE user, GRANT przywilej. Użytkownik init również tworzy nową rolę **app** z uprawieniami UPDATE, INSERT i SELECT na powyższych tabelach

Tabela **global_ids** przechowuje wszystkie id, które zostały użyte do tej pory w programie, dzięki temu jesteśmy w stanie kontrolować globalną unikalność id we wszystkich tabelach
Wytłumaczenie zawartości kolumn, które mogą być niejasne:

- value w **vote** - przechowuje informację o tym czy oddany głos jest za (True) czy przeciw (False)
- positive_votes i negative_votes w **action** - przechowuje informację o oddanej sumarycznej liczbie głosów za/przeciw wobec danej akcji
- action_up w **member** - przechowuje wartość wszystkich głosów upvotes wobec projektów które dany członek utworzył
- action_ratio w **member**- przechowuje wartość (downvotes - upvotes), która potrzebna jest do uznania członka za trolla (jeśli jest dodatnia)

3 Implementacja

3.1 Dokładny opis wejścia

Obiekty json podawane na wejściu mają strukturę:

```
{
    <function> : <value>        // nazwa funkcji
    {
        <arg1> : <value>        // argumenty funkcji
        <arg2>  : <value>
        . . .
        <argn>  : <value>
    }
}
```

Na przykład funkcja **open** z argumentami <**database**> <**login**> oraz <**password**> może zostać przekazana na wejściu jako:

```
{ "open": { "database": "student", "login": "app", "password": "qwerty"}}
```

Skutkuje to wywołaniem przez program odpowiedniej funkcji przetwarzającej taki obiekt (mającej na celu ustanowienie połączenia z bazą danych wraz z danymi dostępowymi podanymi w argumentach)

3.2 INIT i kolejne wywołania programu

Podczas pierwszego uruchomienia programu należy użyć flagi --init. Podczas uruchomienia --init program pobiera wyłącznie jsony z funkcją open i leader. Jako pierwszy json pobrany powinien być ten zawierający <open> który definiuje elementy bazy i nawiązuje z nią połączenie, w przypadku niepowodzenia funkcji obsługującej <open> program zakończy działanie.
Kolejne wywoałania funkcji obsługującej <leader> będą definiować nowe krotki członków, którzy są leaderami.

Kolejne wywołania programu (bez flagi --init) pobierają i obsługują tylko jsony z funkcjami: open, support/protest, upvote/downvote, actions, projects, votes i trolls, które zostaną szczegółowo opisane w dalszej części dokumentacji. Również tutaj w przypadku niepowodzenia <open> program zakończy pracę.

Program przetwarza wejście parserem json, po czym przekazuje je do funkcji **if_case(dic, case)**, której argumentem jest nowo utworzony słownik powstały na wskutek działania parsera, oraz nazwa funkcji zawarta w jsonie

W dalszej części dokumentacji opisane będą sposoby działania funkcji do których zostanie przekierowany program

3.3