

Projekt

Agnieszka Pawicka

1 Jak używać

W archiwum znajduje się dołączony plik run.sh. Aby otworzyć program jako użytkownik init należy wpisać komendę „./run.sh -init” . Następnie aby połączyć się z bazą:

{ "open": { "database": "student", "login": "init", "password": "qwerty" } }. Później poprawne są jedynie wywołania funkcji leader, np.:

{ "leader": { "timestamp": 1557473000, "password": "abc", "member": 1 } }.

Aby otworzyć program jako użytkownik app należy wpisać komendę „./run.sh -app ”. Następnie aby połączyć się z bazą: { "open": { "database": "student", "login": "app", "password": "qwerty" } }. Od teraz poprawne są wywołania definiowane w poleceniu projektu, np.

{ "support": { "timestamp": 1557475701, "password": "123", "member": 3, "action": 600, "project": 5000 } }

{ "trolls": { "timestamp": 1557477055 } }

2 Opis

Baza składa się z pięciu tabel:

1. members(id, cryptpwd, leader, lastactivity, upvotes, downvotes), gdzie:
 - id - identyfikator użytkownika
 - cryptpwd - hasło zakodowane przy pomocy funkcji crypt z modułu pgcrypto (md5)
 - leader - true lub false
 - lastactivity - timestamp
 - upvotes, downvotes - wartości opisane w funkcji trolls
2. actions(id, memberid, projectid, action, time) - action przyjmuje wartości 's' (support), 'p' (protest)
3. votes(memberid, actionid, vote, time) - vote przyjmuje wartości 'u' (upvote), 'd' (downvote)
4. project(id, authority, creationtime)

5. ID - zawierające wszystkie identyfikatory użytkowników, akcji, projektów, organów władzy

Użytkownik init jest na prawach SUPERUSER, użytkownik app może wykonywać działania typu SELECT, INSERT, UPDATE.

Implementacja poszczególnych funkcji API:







- support/protest *timestamp member password action project [authority]*:
 1. sprawdzenie, czy *member* jest istniejącym ID członka, jeśli nie, dodawana jest nowa krotka do tabeli Member
 2. sprawdzenie, czy hasło jest poprawne i czy członek jest aktywny, jeśli nie: zwrócenie błędu
 3. sprawdzenie, czy podany projekt pojawił się wcześniej, jeśli nie pojawił się i nie podano authority lub pojawił się i podano inne authority niż poprzednio zwracany jest błąd
 4. jeśli projekt nie pojawił się wcześniej dodawana jest krotka (project, authority) do tabeli Projects
 5. dodawana jest nowa krotka do tabeli Actions o odpowiednich wartościach
- upvote/downvote *timestamp member password action* :
 1. sprawdzenie, czy *member* jest istniejącym ID członka, jeśli nie, dodawana jest nowa krotka do tabeli Member
 2. sprawdzenie, czy hasło jest poprawne i czy członek jest aktywny, jeśli nie: zwrócenie błędu
 3. sprawdzenie, czy istnieje akcja o podanym id, jeśli nie: zwracany jest błąd
 4. dodawana jest nowa krotka do tabeli Votes (jeśli jest to drugie głosowanie danego członka w tej samej akcji, to nowa krotka złamie więzy klucza głównego tabeli Votes i nie zostanie dodana)
- actions *timestamp member password [type] [project [authority]*
votes *timestamp member password [action , project]*
projects *timestamp member password [authority]*:
 1. sprawdzenie, czy *member* jest istniejącym ID członka będącego liderem, jeśli nie, zwracany jest błąd,
 2. sprawdzenie, czy hasło jest poprawne i czy członek jest aktywny, jeśli nie: zwrócenie błędu,
 3. wywoływana jest odpowiednia funkcja SELECT
- trolls *timestamp* : Wywoływana jest odpowiednia funkcja SELECT operująca na tabeli members.





Dodatkowe funkcje „postgresowe”:




- wyzwalacze `id_trigger()`, `id_trigger_authority()` wstawiające nowe krotki do tabeli `id`
- wyzwalacz `count_votes()` zliczający głosowania na akcje danego członka partii
- funkcja `active()` sprawdzająca, czy członek partii wciąż jest aktywny
- funkcja `action_type()` konwertująca zawartość kolumny `actions.action` na wartości „protest”, „support”
- funkcja `timestamp_cast()` konwertująca unix timestamp






Pomocnicze metody w pythonie:


- `open_connection(...)` - łączy się z bazą
- `close_connection(...)` - zamyka połączenie z bazą
- `is_leader(...)` - sprawdza, czy dany użytkownik jest liderem
- `is_member(...)` - sprawdza, czy w bazie jest podany użytkownik, jeśli nie ma, próbuje dodać takiego
- `is_active(...)` - sprawdza, czy podany użytkownik jest aktywny
- `is_action(...)` - sprawdza, czy podana akcja jest w bazie
- `is_projet(...)` - sprawdza, czy podany projekt jest w bazie, jeśli nie, próbuje dodać
- `create_leader(...)` - dodaje nowego lidera
- `new_action(...)` - dodaje nową akcję
- `new_vote(...)` - dodaje nowy głos
- `trolls(...)`, `votes(...)`, `actions(...)`, `projects(...)` - wywołują odpowiednie `SELECT`y
- `taken_args_str(...)`, `taken_args(...)` - pomocnicze funkcje do wykrywania opcjonalnych argumentów

Member	
 id	numeric(19, 0)
 cryptpwd	varchar(128)
 lastActivity	timestamp
 leader	int4
 upvotes	int4
 downvotes	int4

Votes	
 actionId	numeric(19, 0)
 memberId	numeric(19, 0)
 time	timestamp
 vote	char(1)

Project	
 id	numeric(19, 0)
 authority	numeric(19, 0)
 creationDate	timestamp

Actions	
 Id	numeric(19, 0)
 action	char(1)
 memberId	numeric(19, 0)
 projectId	numeric(19, 0)
 time	timestamp

ID	
 id	numeric...

