Dokumentacja projektu z Baz Danych

Prowadzący : Jan Otop

Łukasz Klasiński

Wrocław, 12 czerwca 2018

1. Diagram E-R - model konceptualny

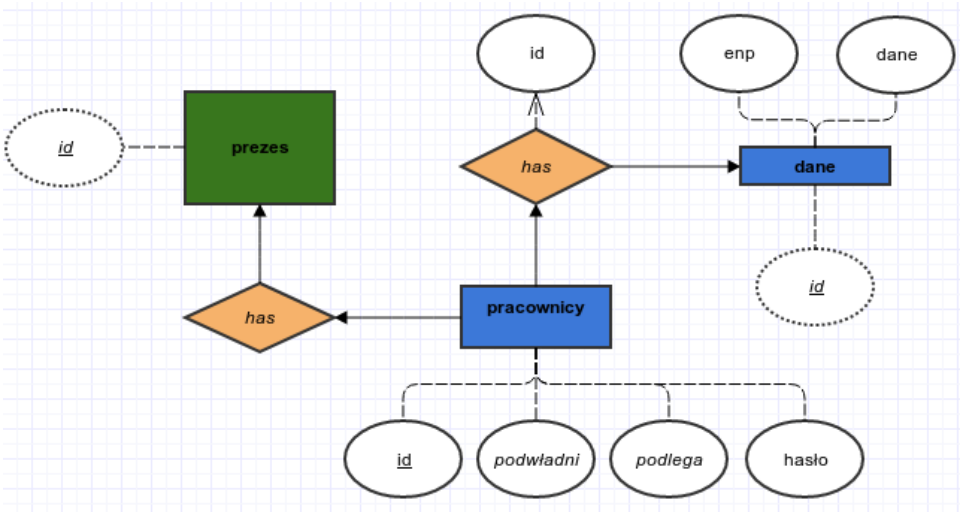

Diagram pokazuje koncept bazy dla danego problemu. Atrybut *podwładni* ze zbioru **pracownicy**, oznaczaja wszystkich bezpośrednio podległych pracowników. Z kolei *podlega* pracowników, którym bezpośrednio lub pośrednio podlega. Encja **prezes** zawiera pojedyńczy wpis, zawierający *id* pracownika, który jest prezesem.

Użytkownik ***app*** ma możliwość odczytu oraz zapisu encji pokolorowanych na niebiesko, tylko odczytu dla zielonych tabel. Z kolei ***init*** ma dostęp do całej bazy danych.

2. Implemetacja

Funkcje dodające/modyfikujące użytkowników będą wykonywać standardowe funkcje *SELECT* oraz *UPDATE* języka sql, ze sprawdzaniem, czy dany użytkownik posiada odpowiednie prawa.

Funkcje szukające pracowników, którym dana osoba podlega będą zwracać wartości zawarte w liście *podlega*. Z kolei przy szukaniu podległych pracowników, rekurencyjnie zwracają wartości *podwładni*.

Podobnie usuwanie pracownika, rekurencyjnie usuwa dane kolejnych pracowników.

Tworzenie wpisu prezesa dodatkowo dodaje odpowiedni wpis do tabeli *prezes*.

3. Wymagania

Api wymaga zainstalowanego interpretera języka **Python** w wersji co najmniej **2.7** oraz zainstalowanego modułu **psycopg2** w wersji przynajmniej **2.7.2**. Poza tym api korzysta z silnika baz danych **Postgresql** w wersji $\geq$ **9.1.23** z zainstalowanym modułem **pgcrypto**.

4. Uruchomienie

Spośród zawartych plików należy uruchomić za pomocą interpretera python plik **api.py**, który po uruchomieniu będzie czekać na standardowym wejściu na zgodne ze specyfikacją danych do napotkania znaku końca lini. Dla każdego zapytania program zwraca dwie możliwe wartości w formacie JSON:

- ```
  {"status" : "OK" , "data" : [ "v1", "v2",...]}
  ```

  W przypadku powodzenia, gdzie zawartość pole *data* jest opcjonalne i występuje tylko przy zapytaniach oczekujących na wynik. Poszczególne dane wynikowe są zawsze w tablicy. Oznacza to, że dla wyników jednoelementowych zwrócona zostanie tablica jednoelementowa. Ponadto kolejne wartości v_i są zawsze ciągami tekstowymi, nawet w przypadku wartości boolowskich, np: *"data" : ["false"]* może być wynikiem dla zapytania *ancestor*. Jest to zastosowane w celu unifikacji danych wyjściowych. Ponieważ wszystkie wartości na wejściu są traktowane oraz
  ```

zapisywane jako stringi, zwracane dane także nimi są. W szczególności dzięki temu unikatowe identyfikatory pracowników *emp* mogą być dowolnymi typami danych.

- ```
  {"status" : "ERROR", "debug" : "opis"}
  ```

  W przypadku błędu, gdzie *debug* zawiera opcjonalne informacje o przyczynie błędu. Przykładowo *Wrong admin password* oznacza, że podane zostało nieprawidłowe hasło dostępu do aplikacji. Dodatkowo jeśli błąd nie był spowodowany błędnymi danymi, tylko błędem bazy danych, to jego wynik zostanie umieszczony w polu *debug*. W szczególności, jeśli przypadkowo dwa razy poda zapytanie z parametrem *init*, to zwrócony zostanie błąd silnika bazy danych informujacy o tym, że tablice, które próbuje utworzyć api już istnieją.

Przykładowe uruchomienie:

```
python api.py
```

Dodatkowo w systemach uniksowych, można po nadanu odpowiednich praw uruchomić plik *api.py* jako plik wykonywalny:

```
chmod +x api.py
./api.py
```
```