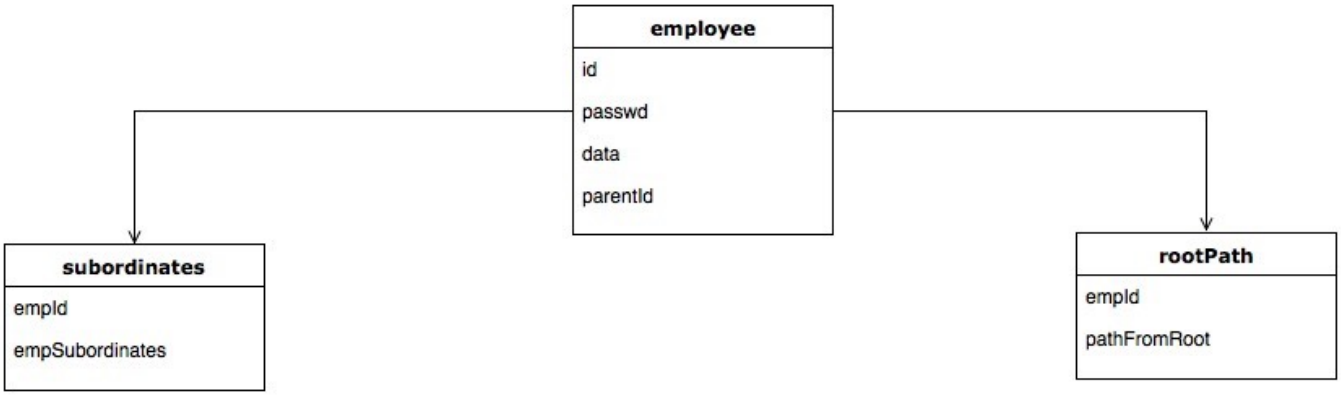
Bazy Danych 2017/18 - projekt

Wojciech Pratkowiecki, nr albumu: 281417

Dokumentacja

1. Model konceptualny



Baza danych składa się z trzech tabel:

1. employee - zawiera podstawowe informacje o pracowniku, które są podawane w trakcie tworzenia użytkownika przy pomocy instrukcji **new**
2. rootPath - zawiera klucz obcy *empId* wskazujący na wpis w tabeli *employee*, a także, dla każdego z pracowników, tablicę z id pracowników będących na ścieżce w hierarchii firmy - pierwszym elementem tablicy jest root, ostatnim - pracownik o numerze *empId*. Przy każdym wywołaniu funkcji **new** tworzony jest wpis w tabeli dla nowego użytkownika.
3. subordinates - zawiera klucz obcy *empId* wskazujący na wpis w tabeli *employee* oraz tablicę z bezpośrednimi podwładnymi pracownika o identyfikatorze *empId.* Przy każdym wywołaniu funkcji **new** tworzony jest wpis w tabeli dla nowego użytkownika.

2. Implementacja

Projekt zawiera następujące pliki:

1. dbAPI.py - kod będący głównym mechanizmem programu, zawiera funkcje wywoływane przy łączeniu się z bazą danych, obsługujące wejście i wyjście oraz odpowiednie funkcje dla każdego z zapytań. Zawiera następujące funkcje:
    1. loadQueries(): dla zadanej nazwy pliku z zapytaniami json zwraca listę tych zapytań
    2. connnectToDB(): używa modułu psycopg2 do utworzenia połączenia z zadaną bazą danych, zwracając niezbędne do obługi bazy handlery
    3. checkPassword(): sprawdza poprawność hasła dla zadanego użytkownika z bazy
    4. checkIfEmpExists(): Sprawdza, czy użytkownik o danym id istnieje w bazie
    5. createStatus(): tworzy wynikowy obiekt json
    6. createErrorStatus(): tworzy status błędu
    7. createRoot(): przy uruchomieniu z parametrem init tworzy użytkownika w korzeniu hierarchii o zadanych danych
    8. createEmp(): przy instrukcji **new** tworzy nowego użytkownika
    9. removeEmp(): przy instrukcji **remove** usuwa zadanego użytkownika
    10. getChild(): przy instrukcji **child** wyznacza bezpośrednich podwładnych użytkownika
    11. getParent(): przy instrukcji **parent** wyznacza bezpośredniego przełożonego użytkownika
    12. getAncestors(): przy instrukcji **ancestors** wyznacza wszystkich przełożonych użytkownika
    13. getDescendants(): przy instrukcji **descendants** wyznacza wszystkich podwładnych użytkownika
    14. getAncestor(): przy instrukcji **ancestor**, sprawdza, czy zadany użytkownik jest przełożonym drugiego
    15. getData(): przy instrukcji **read** zwraca dane użytkownika
    16. updateData(): przy instrukcji **update** uaktualnia dane użytkownika
    17. openDB(): przy instrukcji **open** tworzy połączenie z bazą zwracając handlery
    18. execQueries(): odpowiada za obsługę zapytań przy uruchomieniu programu bez parametrów
    19. initDB(): odpowiada za obsługę zapytań przy uruchomieniu programu z parametrem init

Ponadto funkcje dbają o zachowanie warunków dla każdej instrukcji zadanych w treści pracowni

2. getUpdateQueries.py - zawiera funkcje wysyłające zapytania do bazy danych i zwracające odpowiedzi.
3. createQueries.py - zawiera funkcje wysyłające zapytania równoważne plikowi createDB.sql, które tworzą użytkownika app oraz tabele w bazie
4. createDB.sql - model fizyczny bazy, zapytania z tego pliku są wywoływane przy każdym uruchomieniu projektu z parametrem init
5. clear.sql - pomocnicze zapytania usuwające użytkownika app oraz wszystkie tabele z bazy, umożliwiając przeprowadzenie kolejnego uruchomienia z parametrem init
6. tests/ - folder z przykładowymi testami
7. Makefile - prosty plik umożliwiający uruchomienie projektu na 3 sposoby - z parametrem init i domyślnym modelem fizycznym, z parametrem init i własnym modelem fizycznym oraz bez parametrów

Uruchomienie aplikacji

Głównym silnikiem jest plik dbAPI.py. Aplikację należy uruchomić przy użyciu python3. Szczegółowe informacje można uzyskać wpisując komendę
python3 dbAPI.py --help
Każde uruchomienie programu wymaga podania w argumentach nazwy pliku zawierającego zapytania. Ponadto wyróżnione są uruchomienia z parametrem init. Można wówczas podać własny model fizyczny bazy przy pomocy argumentu --db, domyślnie baza jest budowana na podstawie createDB.sql. Przykładowe uruchomienia z parametrem init:
python3 dbAPI.py --init testinit.json
python3 dbAPI.py --init --db createDB.sql testinit.json
W przypadku uruchomienia aplikacji bez parametru obsługiwane są zapytania z jedynego argumentu:
python3 dbAPI.py test.json

W katalogu znajduje się plik Makefile umożliwiający równoważne uruchomienie programu przy użyciu jednej z trzech komend:
make init input=testinit.json
make initCustrom db=createDB.sql input=testinit.json
make run input=test.json