Pokročilé databázové technológie

Zadanie 1 – import tweetov do PostgreSQL

Marek Adamovič

Cvičenie: Piatok 11:00 – 12:40

Ing. Ján Balažia, PhD. 2022/2023

Obsah

[Opis algoritmu 3](#_Toc115554944)

[Database setup 3](#_Toc115554945)

[First reading 3](#_Toc115554946)

[Second reading 4](#_Toc115554947)

[main\_migration\_conversations.py 4](#_Toc115554948)

[conversations\_migration.py 4](#_Toc115554949)

[hashtags\_migration.py 4](#_Toc115554950)

[context\_migration.py 5](#_Toc115554951)

[annotations\_migration.py 5](#_Toc115554952)

[links\_migration.py 5](#_Toc115554953)

[Použité technológie 5](#_Toc115554954)

[Použité SQL 5](#_Toc115554955)

[Počet a veľkosť záznamov 5](#_Toc115554956)

# Opis algoritmu

Program sa spúšťa cez súbor „main.py“. Tento súbor si následne importuje ďalšie časti protokolu, ktoré sú umiestnené v adresári „code“. Hlavný súbor („main.py“) je rozdelený na 5 nasledovných častí:

* Database setup
* First reading
* Second reading
* Third reading
* Constraints

Pre vyššiu rýchlosť využíva riešenie 3 hash tabuľky pre nasledujúce dáta:

* existujúci autori
* existujúce konverzácie
* existujúce hashtagy a ich tagy

## Database setup

Táto časť, obsiahnutá v súbore „database\_init.py” sa stará o pripojenie na databázu a vytvorenie tabuliek. Tabuľky sú vytvorené podľa schémy zo zadania. Výnimkou sú obmedzenia (constraints), ktoré sa kvôli efektivite riešenia doplnia do tabuliek až v rámci poslednej časti riešenia -> Constraints.

## First reading

Táto časť ohraničuje prečítanie a spracovanie súboru „authors.jsonl.gz“. Funkcionalita sa nachádza v súbore „main\_migration\_authors.py“.

Najskôr sa vytvorí prázdna hash tabuľka (vyplníme ju prázdnymi poľami), do ktorej si postupne zapisujeme id autorov, aby sme mali rýchly prístup k informácií, či daný autor existuje alebo nie. Miesto v tabuľke pre id autora vyberáme pomocou zmodulovania id autora s veľkosťou hash tabuľky. Nevadí nám, ak sa na danom mieste už nejaké id nachádza, pretože každý prvok v našej hash tabuľke je tvorený poľom, vďaka čomu nám stačí nový prvok pridať na koniec poľa. Ak potrebujeme zistiť, či autor existuje, stačí nám zmodulovať jeho id a pozrieť sa, či sa na danej pozícií nachádza jeho id v poli.

Po vytvorení (zatiaľ) prázdnej hash tabuľky začneme čítať súbor s autormi. Pri každom riadku súboru vyberieme potrebné informácie a pridáme autora do dávky („batchu“), ktorého veľkosť si vieme nastaviť navrchu súboru pomocou konštanty „BATCH\_SIZE“. Keď batch dosiahne veľkosť konštanty „BATCH\_SIZE“, tak sa so všetkými dátami pošle do databázy a následne sa vyprázdni. Keď prečítame celý súbor, pošlú sa zvyšné dáta (teda neúplný batch) a program sa vráti do hlavnej časti (main súbor). Funkcia vráti hash tabuľku s autormi, ktorú sme si počas čítania naplnili dátami o existujúcich autoroch.

## Second reading

Najväčšia časť protokolu, ktorá číta a spracováva súbor „conversations.jsonl.gz“. Je rozdelená do nasledujúcich súborov:

* main\_migration\_conversations.py
  + conversations\_migration.py
  + hashtags\_migration.py
  + context\_migration.py
  + annotations\_migration.py
  + links\_migration.py

### main\_migration\_conversations.py

V tomto súbore postupne načítavame dáta o konverzáciách. Následne ich posielame do funkcií, kde sa spracúvajú do batchov (podobne ako pri autoroch, aj tu máme konštantu „BATCH\_SIZE“, ktorou si vieme nastaviť maximálnu veľkosť batchov). Keď sa batch s konverzáciami naplní na BATCH\_SIZE, všetky batche sa pošlú do databázy a následne sa vyprázdnia. Funkcia v tomto súbore vracia hash tabuľku s konverzáciami do hlavnej časti programu, ktorú potrebujeme pre efektívne spracovanie dát v časti Third reading.

Všetky nasledujúce súbory patriace do Second readingu obsahujú (hlavne) funkcie spracúvajúce dáta do ich vlastných batchov a taktiež funkcie pre posielanie týchto batchov do korešpondujúcich tabuliek.

### conversations\_migration.py

Funkcie v tomto súbore sú zodpovedné za viacero vecí. Jednou z nich je tvorba hash tabuľky pre konverzácie, ktorú potrebujeme pre Third reading. Avšak využívame ju aj pri napĺňaní batchov pre zisťovanie duplikátov (čo je rýchlejšie ako dopytovať databázu). Ak zistíme, že riešená konverzácia je duplikát, ktorý už existuje v našej hash tabuľke, nejdeme ju riešiť druhýkrát (tým pádom neriešime ani ostatné dáta pre zbytok tabuliek v tomto zázname) Pri spracovávaní dát konverzácie sa teda najskôr pozrieme do hash tabuľky, či už sme konverzáciu s rovnakým id nespracovávali. Ak nie, tak toto id konverzácie pridáme do hash tabuľky. Následne sa pozrieme, či existuje autor konverzácie podľa author\_id, ktoré vyhľadáme v autorskej hash tabuľke. Ak autor neexistuje, tak vytvoríme anonymného autora (ktorý má len id) a pridáme ho do hash tabuľky autorov a taktiež do batchu nových autorov, ktorý neskôr posielame do databázy. Konečne spracujeme informácie o samotnej kovnerzácií a uložíme ju do batchu konverzácií, ktorý taktiež neskôr posielame do databázy.

### hashtags\_migration.py

Tento súbor obsahuje funkciu na vytvorenie hash tabuľky s hashtagmi. Do tejto tabuľky ukladáme unikátne hashtagy s nami prideleným id. Vďaka tejto hash tabuľke vieme následne veľmi rýchlo nájsť id hashtagu len podľa jeho tagu (čo značne využívame pri plnení tabuľky conversation\_hashtags). Ďalej sa tu nachádza funkcia na naplnenie batchu s dátami, ktorá funguje nasledovne:

* skontroluje sa, či sa v prijatých dátach nachádza hashtag(alebo viacero hashtagov)
* pre každý hashtag vykoná:
  + zahashuje hashtag tag
  + pomocou hashu tagu nájde príslušnú pozíciu v hash tabuľke
  + ak sa na tejto pozícií hashtag už nachádza, zoberie sa jeho id
    - pridá sa záznam do conversation\_hashtag, ktorý hovorí, že ktorý hashtag sa vyskytol v ktorej konverzácií
  + ak sa na tejto pozícií hashtag nenachádza, pridá sa tam s novým id, ktoré sa hneď zapíše do tabuľky conversation\_hashtag spolu s id riešenej konverzácie

### context\_migration.py

Súbor context\_migration.py zabezpečuje spracovanie dát a ich ukladanie do troch batchov pre tri korešpondujúce tabuľky (context\_domains, context\_entities, context\_annotations). Dôležité je sledovať, či sa v aktuálnom zázname konverzácie nachádzajú domény a entity, aby sme ich vedeli spracovať.

### annotations\_migration.py

V tomto súbore sa zabezpečuje správne ukladanie dát do batchu a jeho následné odoslanie do tabuľky annotations. Vždy je potrebné skontrolovať, či sa v aktuálnom zázname konverzácie nachádzajú nejaké anotácie.

### links\_migration.py

V tomto súbore sa zabezpečuje správne ukladanie dát do batchu a jeho následné odoslanie do tabuľky links. Keďže v tabuľke links máme obmedzenie pre url (najviac 2048 znakov), musíme vždy skontrolovať, či aktuálna url nepresahuje daný počet znakov. V prípade, že áno, záznam nepridávame.

## Third reading

Ďalšia časť protokolu znovu číta súbor s konverzáciami za účelom naplnenia tabuľky conversation\_references. Použitie hash tabuľky nám značne urýchli napĺňanie, keďže na existenciu konverzácie, ktorá má parent\_id, sa nedopytujeme databázy (ktorá má svoje údaje uložené na disku), ale hash tabuľky, ktorá je uložená v RAM.

## Constraints

V poslednej časti pridávame obmedzenia pre jednotlivé tabuľky (foreign keys a unique constraints). Je oveľa efektívnejšie naraz skontrolovať všetky záznamy, ako ich kontrolovať po jednom pri vkladaní. Ak by sme mali záznam, ktorý by porušoval pridanú podmienku, program by vyhodil error (tým pádom neriskujeme stratu integrity dát bez toho, aby sme o tom vedeli).

# Použité technológie

Python – vybral som si python kvôli širokej škále rôznych funkcionalít či už samotného pythonu alebo naimportovaných knižníc (meranie času, otvorenie a práca s .gzip súborom, ovládač pre postgres databázu...). Medzi jeho nevýhody patrí pomalšia rýchlosť, avšak na druhej strane verím, že tá je vynahradená efektívnejším programovaním.

Hash tabuľky – som si vybral pre markantné zvýšenie rýchlosti behu programu na úkor operačnej pamäte. Je oveľa rýchlejšie získať informácie z operačnej pamäte ako z databázy, ktorá má svoje dáta uložené na pevnom disku.

# Použité SQL

## Vytvorenie tabuliek

CREATE TABLE IF NOT EXISTS authors(

id BIGINT PRIMARY KEY,

name VARCHAR(255),

username VARCHAR(255),

description TEXT,

followers\_count INT,

following\_count INT,

tweet\_count INT,

listed\_count INT

);

CREATE TABLE IF NOT EXISTS conversations(

id BIGINT PRIMARY KEY,

author\_id BIGINT NOT NULL,

content TEXT NOT NULL,

possibly\_sensitive BOOLEAN NOT NULL,

language VARCHAR(3) NOT NULL,

source TEXT NOT NULL,

retweet\_count INT,

reply\_count INT,

like\_count INT,

quote\_count INT,

created\_at TIMESTAMPTZ NOT NULL

);

CREATE TABLE IF NOT EXISTS hashtags(

id BIGINT PRIMARY KEY,

tag TEXT NOT NULL

);

CREATE TABLE IF NOT EXISTS conversation\_hashtags(

id BIGSERIAL PRIMARY KEY,

conversation\_id BIGINT NOT NULL,

hashtag\_id BIGINT NOT NULL

);

CREATE TABLE IF NOT EXISTS context\_domains(

id BIGINT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT

);

CREATE TABLE IF NOT EXISTS context\_entities(

id BIGINT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT

);

CREATE TABLE IF NOT EXISTS context\_annotations(

id BIGSERIAL PRIMARY KEY,

conversation\_id BIGINT NOT NULL,

context\_domain\_id BIGINT NOT NULL,

context\_entity\_id BIGINT NOT NULL

);

CREATE TABLE IF NOT EXISTS annotations(

id BIGSERIAL PRIMARY KEY,

conversation\_id BIGINT NOT NULL,

value TEXT NOT NULL,

type TEXT NOT NULL,

probability NUMERIC(4, 3) NOT NULL

);

CREATE TABLE IF NOT EXISTS links(

id BIGSERIAL PRIMARY KEY,

conversation\_id BIGINT NOT NULL,

url VARCHAR(2048) NOT NULL,

title TEXT,

description TEXT

);

CREATE TABLE IF NOT EXISTS conversation\_references(

id BIGSERIAL PRIMARY KEY,

conversation\_id BIGINT NOT NULL,

parent\_id BIGINT NOT NULL,

type VARCHAR(20) NOT NULL

);

# Počet a veľkosť záznamov