Slovenská Technická Univerzita v Bratislave
Fakulta Informatiky a Informačných Technológií

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

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

Pre formátovanie dát používame v celom programe funkciu mogrify(), ktorá nám zabezpečí, aby boli dáta vo formáte prijateľnom pre databázu.

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“, ktorá je nastavená na 100 000. 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. Záznamy spracúvame spôsobom, že riešime všetky konverzácie, ktoré máme poznačené v hash tabuľke. To znamená, že ak máme viacero záznamov konverzácií s rovnakým id, tak ich tu riešime všetky.

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
);
```

SQL pre vytvorenie tabuliek v databáz. Dátové typy som použil podľa schémy zo zadania. Jedinou výnimkou je dátový typ BIGSERIAL, čo je BIGINT, ktoré sa automaticky inkrementuje. Do vytvárania tabuliek som nedával obmedzenia (constraints) pre cudzie kľúče a unikátne záznamy (pri hashtagoch), keďže by to značne spomalilo vkladanie do týchto tabuliek. Obmedzenia pridávam až na záver pomocou ALTER TABLE <meno> ADD CONSTRAINT. To skontroluje všetky riadky naraz, vďaka čomu program ušetrí množstvo času.

Vloženie dát do tabuľky authors

```
INSERT INTO authors
VALUES <formated_data>
ON CONFLICT (id) DO NOTHING
```

Vkladanie záznamov do tabuľky authors. Keďže nemáme špecifikované, ktoré riadky sa majú do tabuľky vložiť, INSERT sa bude snažiť naplniť všetky, ktoré v tabuľke existujú. Namiesto <formated_data> sa doplnia samotné dáta z batchu (veľkosť batchu je nastavená na 100 000 riadkov), ktoré sa pošlú v jednom inserte, čo je oveľa efektívnejšie ako keby ich posielame samostatne jeden riadok na jeden insert. Posledný riadok SQL nám zabezpečí, aby nám program nepadol, keby sa snažíme vložiť duplikát autora do databázy.

Vloženie dát do tabuľky conversations

```
INSERT INTO conversations
```

VALUES <formated_data>

ON CONFLICT (id) DO NOTHING

Vkladanie záznamov do tabuľky conversations. Namiesto <formated_data> sa doplnia samotné dáta z batchu, ktoré sa pošlú v jednom inserte, čo je oveľa efektívnejšie ako keby ich posielame samostatne jeden riadok na jeden insert. Posledný riadok SQL nám zabezpečí, aby nám program nepadol, keby sa snažíme vložiť duplikát konverzácie do databázy.

Vloženie dát do tabuľky hashtags

INSERT INTO hashtags

VALUES <formated_data>

Vkladanie záznamov do tabuľky hashtags. Namiesto <formated_data> sa doplnia samotné dáta z batchu, ktoré sa pošlú v jednom inserte, čo je oveľa efektívnejšie ako keby ich posielame samostatne jeden riadok na jeden insert.

Vloženie dát do tabuľky conversation_hashtags

INSERT INTO conversation_hashtags(conversation_id, hashtag_id)

VALUES <formated_data>

Vkladanie záznamov do tabuľky conversation_hashtags. Namiesto <formated_data> sa doplnia samotné dáta z batchu, ktoré sa pošlú v jednom inserte, čo je oveľa efektívnejšie ako keby ich posielame samostatne jeden riadok na jeden insert. Máme špecifikované, ktoré riadky ideme vkladať. Keďže tabuľka používa datatyp BIGSERIAL pre id, nemusíme ho pridávať my, lebo sa vyplní automaticky.

Vloženie dát do tabuľky context_domains

INSERT INTO context_domains

VALUES <formated_data>

ON CONFLICT (id) DO NOTHING

Vkladanie záznamov do tabuľky context_domains. Namiesto <formated_data> sa doplnia samotné dáta z batchu, ktoré sa pošlú v jednom inserte, čo je oveľa efektívnejšie ako keby ich posielame samostatne jeden riadok na jeden insert. Posledný riadok SQL nám zabezpečí, aby nám program nepadol, keby sa snažíme vložiť duplikát domény do databázy.

Vloženie dát do tabuľky context_entities

INSERT INTO context_entities

VALUES <formated_data>

ON CONFLICT (id) DO NOTHING

Vkladanie záznamov do tabuľky context_entities. Namiesto <formated_data> sa doplnia samotné dáta z batchu, ktoré sa pošlú v jednom inserte, čo je oveľa efektívnejšie ako keby ich posielame samostatne jeden riadok na jeden insert. Posledný riadok SQL nám zabezpečí, aby nám program nepadol, keby sa snažíme vložiť duplikát entity do databázy.

Vloženie dát do tabuľky context_annotations

INSERT INTO context_annotations(conversation_id, context_domain_id, context_entity_id)

VALUES <formated_data>

Vkladanie záznamov do tabuľky context_annotations. Namiesto <formated_data> sa doplnia samotné dáta z batchu, ktoré sa pošlú v jednom inserte, čo je oveľa efektívnejšie ako keby ich posielame samostatne jeden riadok na jeden insert.

Vloženie dát do tabuľky annotations

INSERT INTO annotations(conversation_id, value, type, probability)

VALUES <formated_data>

Vkladanie záznamov do tabuľky annotations. Namiesto <formated_data> sa doplnia samotné dáta z batchu, ktoré sa pošlú v jednom inserte, čo je oveľa efektívnejšie ako keby ich posielame samostatne jeden riadok na jeden insert.

Vloženie dát do tabuľky links

INSERT INTO links(conversation_id, url, title, description)

VALUES <formated_data>

Vkladanie záznamov do tabuľky links. Namiesto <formated_data> sa doplnia samotné dáta z batchu, ktoré sa pošlú v jednom inserte, čo je oveľa efektívnejšie ako keby ich posielame samostatne jeden riadok na jeden insert.

Vloženie dát do tabuľky conversation_references

INSERT INTO conversation_references(conversation_id, parent_id, type)

VALUES <formated_data>

Vkladanie záznamov do tabuľky conversation_references. Namiesto <formated_data> sa doplnia samotné dáta z batchu, ktoré sa pošlú v jednom inserte, čo je oveľa efektívnejšie ako keby ich posielame samostatne jeden riadok na jeden insert.

Pridanie obmedzení (constraints)

ALTER TABLE conversations

ADD CONSTRAINT fk_user FOREIGN KEY(author_id)

REFERENCES authors(id);

ALTER TABLE hashtags

ADD CONSTRAINT unq_key UNIQUE(tag);

ALTER TABLE conversation_hashtags

ADD CONSTRAINT fk_conversation FOREIGN KEY(conversation_id)

REFERENCES conversations(id);

ALTER TABLE conversation_hashtags

ADD CONSTRAINT fk_hashtag FOREIGN KEY(hashtag_id)

REFERENCES hashtags(id);

ALTER TABLE context_annotations

ADD CONSTRAINT fk_conversation FOREIGN KEY(conversation_id)

REFERENCES conversations(id);

ALTER TABLE context_annotations

ADD CONSTRAINT fk_context_domain FOREIGN KEY(context_domain_id)

REFERENCES context_domains(id);

```
ALTER TABLE context_annotations
ADD CONSTRAINT fk_context_entity FOREIGN KEY(context_entity_id)
REFERENCES context_entities(id);

ALTER TABLE annotations
ADD CONSTRAINT fk_conversation FOREIGN KEY(conversation_id)
REFERENCES conversations(id);

ALTER TABLE links
ADD CONSTRAINT fk_conversation FOREIGN KEY(conversation_id)
REFERENCES conversations(id);

ALTER TABLE conversation_references
ADD CONSTRAINT fk_conversation FOREIGN KEY(conversation_id)
REFERENCES conversations(id);
ALTER TABLE conversation_references
ADD CONSTRAINT fk_conversation_parent FOREIGN KEY(parent_id)
REFERENCES conversations(id);
```

SQL slúžiace na pridanie obmedzení do tabuliek. Prvým obmedzením je cudzí kľúč (foreign key), teda ukazovateľ na primárny kľúč (primary key) inej tabuľky. Ďalším obmedzením, ktoré používame pre hashtagy, je unikátnosť (unique), čo zaručí, aby sa tá istá hodnota nenachádzala v tabuľke viackrát. Vďaka tomu, že tieto obmedzenia pridávame až na konci, po vložení všetkých záznamov, ušetríme množstvo času pri vkladaní záznamov.

Počet a veľkosť záznamov

Počet

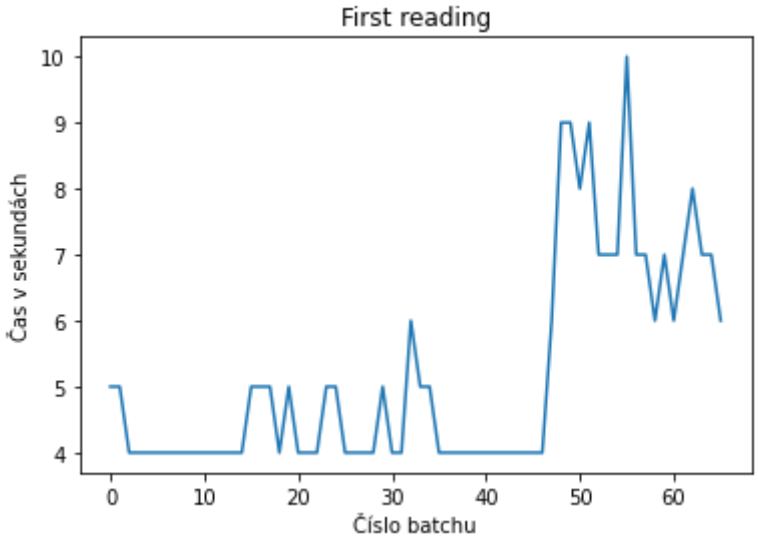
	annotations bigint	authors bigint	context_annotations bigint	context_domains bigint	context_entities bigint	conversation_hashtags bigint	conversation_references bigint	conversations bigint	hashtags bigint	links bigint
1	19458972	5895176	134285948	88	29438	54613745	27950190	32347011	773865	11540704

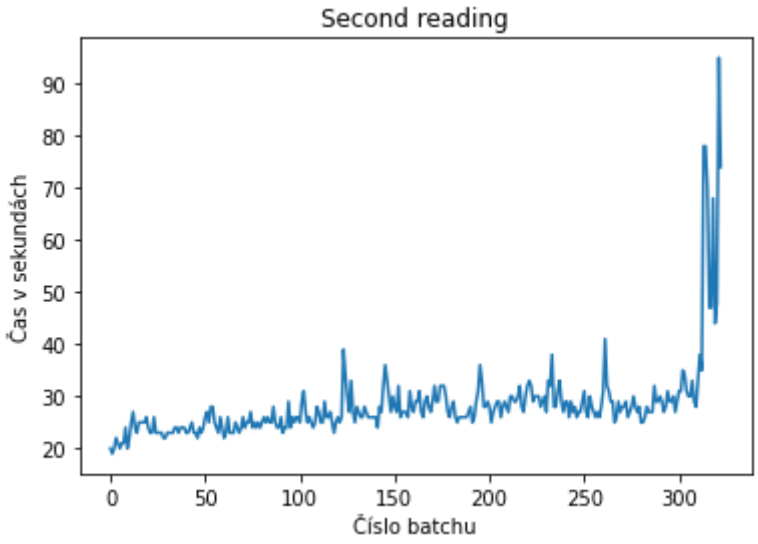
Veľkosť

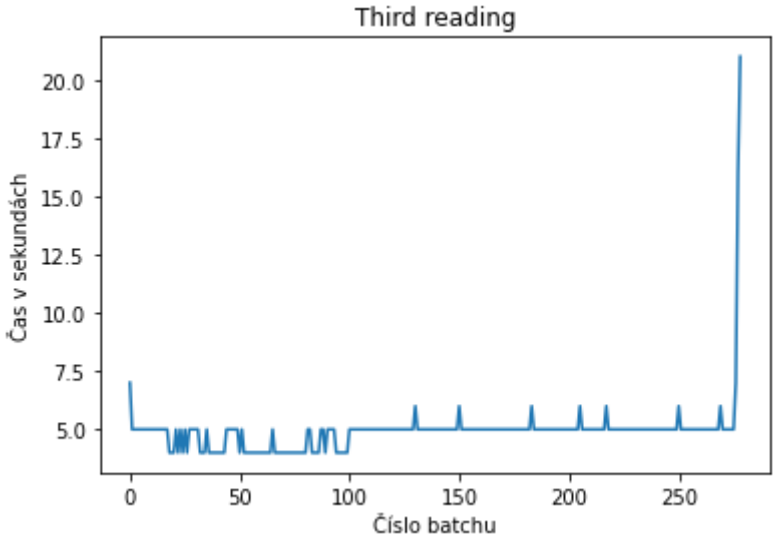
	annotations text	authors text	context_annotations text	context_domains text	context_entities text	conversation_hashtags text	conversation_references text	conversations text	hashtags text	links text
1	1721 MB	1069 MB	10 GB	64 kB	4184 kB	3888 MB	2402 MB	8632 MB	81 MB	2044 MB

Čas importu – grafy

Jednotlivé časy importu (ktoré boli taktiež použité pri tvorbe nasledujúcich grafov) sú priložené v .csv súboroch v priečinku results_times.






Third reading
Čas v sekundách
20.0
17.5
15.0
12.5
10.0
7.5
5.0
0
50
100
150
200
250
Číslo batchu