Leonard Puškáč

PDT Zadanie 1 – Dokumenácia/Protokol

Import Tweetov do Postgres Databázy

link na github classroom: https://github.com/FIIT-DBS/zadanie-pdt-Dewflio

Opis algoritmu pri importe

1. Tweety z kompresovaných jsonl súborov som importoval nasledujúcim spôsobom:
2. Otvoril som authors.jsonl.gz a po riadkoch som prešiel po jeho záznamoch
3. Pri každom zázname som sa pýtal, č isa id autora nachádza v hash table ktorý som predtým vytvoril
4. Ak sa záznam nenachádzal v hash table, pridal som jeho relevantné údaje do tuple ktorú som vložil do pola tuples
5. Keď sa toto pole naplnilo (bolo tam 10,000 záznamov), zavolal som funkciu insert_authors
6. Táto funkcia pomocou psycopg2.extras.execute_values vykoná query pre insertovanie tohto poľa do tabuľky authors v batchoch (blokoch)
7. Veľkosť bloku som pre celý program nastavil na 10,000 a každé query sa vykonáva s veľkosťou stránky takisto 10,000

Po spracovaní autorov program v separátnej funkcii spustí čítanie zo súboru, ktorý obsahuje konverzácie. Tento súbor sa takisto číta iba v jednej iterácii. Rozhodol som sa to robiť takto, aby som ušetril čas strávený importovaním, s tým že referencie na konverzácie, ktoré v databáze neexistujú vyriešim po importe, tým že odstránim každý záznam, ktorý obsahuje kľúč, ktorý neexistuje v tabuľke autorov. Takisto pri importe, respektíve pred ním neriešim foreign keys, To, že nevkladám duplikáty, čo sa týka stĺpcov, ktoré obsahujú tieto kľúče riešim v programe pomocou hash table. Retrospektívne, mohol som to možno riešiť aj pomocou ON CONFLICT DO NOTHING. Respektíve som si to mohol týmto aspoň zaistiť.

Konverzácie a iné entity vpísané v ich json záznamoch sa spracúvajú podobne ako autori:

1. Cyklus iteruje cez riadky v jsonl súbore
2. Pre každý riadok sa najprv spýta, či existuje konverzácia s id aktuálnej v hash table
3. Ak nie, zapíše jej id do hash table
4. Ďalej kontroluje, či sa v json objekte nachádzajú jednotlivé potrebné entitity a pre každú z nich robí to isté
5. Ich tuple (list v ktorom sú vložené jednotlivé atribúty vkladané do tabuľky) je appendovaný do ich vlastného poľa - pomenované “hashtags_insert_vals” v prípade, že sa jedná o hashtagy.
6. Popritom sa počíta koľko takýchto záznamov je vložených do polí, a keď ich počet dosiahne veľkosť bloku (10,000), pole je vložené do funkcie pre insert
7. vo funkcii insert sa pre každý typ entity zavolá pre ich špecifickú tabuľku execute_vals(INSERT….)
8. na konci cyklu (po prečítaní celého súboru) sa zvyšné nevyprázdnené polia znovu posunú do insert funkcie

Veľkosť bloku 10,000 platila pre väčšinu entít, okrem context_domains, a context_entities, keďže ich výsledný počet bol výrazne menší od iných entít.

Zvolené technológie

Na import do databázy som si zvolil použitie knižnice psycopg2 pomocou, ktorej som vedel vykonávať vloženia do tabuľky pomocou batchov. Spravil som to preto, aby som nemusel napríklad volať dopyt pre vloženie pre každý záznam.

Pôvodne som na spracovanie záznamov chcel použiť pyspark, a na študovaní tejto technológie som strávil väčšinu času, ktorý som mal na zadanie. Bohužial som nakoniec pyspark, ktorý by mi dovolil spracovať viacero vecí distribuovane, nedokázal použiť, lebo som zistil, že ak používam windows, musím pri pysparku používať dockery, s čím som sa nechcel v rámci tohto časového okna zaoberať.

Okrem toho som použil pythonovské knižnice gzip a json na prácu so vstupnými súbormi, a csv s výstupnými.

Možné zlepšenia

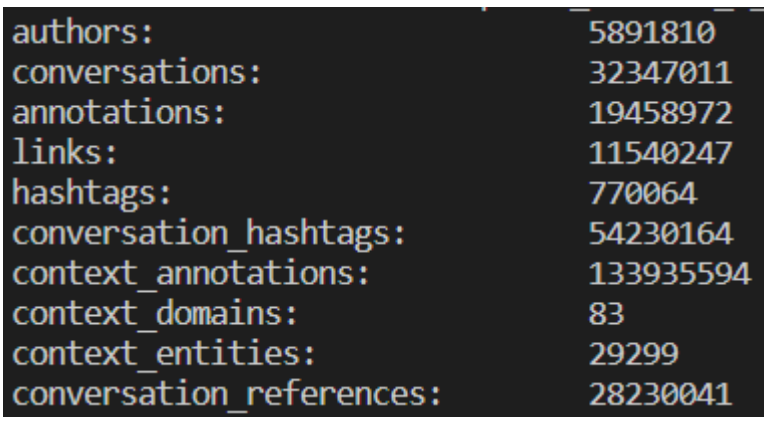
Pravdepodobne som mohol na kontrolu duplicity používať obyčajný python dictionary, namiesto hashtable, ktorý zbytočne zvyšoval počet operácií, pri vyhľadávaní, ktoré sa dialo viacerokrát pre každý záznam.

Mohol som použiť nejakú úroveň paralelizmu, pre operácie, ktoré by to povoľovali, ako napríklad spracovanie jednotlivých entít paralelne v rámci jedného záznamu. Alebo aj spracovanie entít, ktoré si vyžadujú kontrolovať duplicitu, s tým, že hashtable (alebo slovník), ktorý si o tom drží záznam by bol chránenou premennou.

Ďalšia vec je, že ešte nemám vyriešené conversation_references - V programe jednoducho vkladám, každú referenciu bez ohľadu na to, či tweet, na ktorý referuje existuje alebo nie. S tým, že po importe mám v pláne odstrániť tie záznamy, ktoré obsahujú parent_id, ktoré sa nenachádza v tabuľke autorov. Dalo by sa to pomocou vytvorenia hash indexu nad autormi, podla ich id. Ak tento krok nestihnem spraviť do odovzdania, spravím ho po.

Výsledky

Toto sú výsledky z importu, ktorý nanešťastie ľahol pri posledných vkladaniach (počet záznamov menej ako jeden blok), kvôli syntaxovej chybe. Volal som funkciu pre insert konverzácií namiesto linkov, a keď sa program snažil parsnúť input do query z tuples tak nastalo index out of range, pretože sa jedná o tuple inej dĺžky, nehovoriac o tom, že ide o inú tabuľku aj keby sa to podarilo. Túto chybu som opravil, a v čase písania tohto odstavca čakám, kým sa databáza znova importne, tento raz snáď správne.



S tým, že tento import trval takto dlho:

```
3783    2022-10-07T19:44Z;170:50;00:03
3784    2022-10-07T19:44Z;170:53;00:03
3785    2022-10-07T19:44Z;170:57;00:03
3786    2022-10-07T19:44Z;171:01;00:03
3787    2022-10-07T19:44Z;171:05;00:03
3788    2022-10-07T19:44Z;171:08;00:03
3789
```

Výsledky s úspešným koncom:
Výpis toho čo spočítal python:

```
('2022-10-07T23:06Z', '188:15', '00:03')
('2022-10-07T23:06Z', '188:19', '00:04')
('2022-10-07T23:06Z', '188:23', '00:03')
('2022-10-07T23:06Z', '188:27', '00:03')
conversations parsed with count: 32347011
annotations parsed with count: 19458972
hashtags parsed with count: 773853
conversation_hashtags parsed with count: 54234876
links parsed with count: 11540704
context_annotations parsed with count: 133941462
context_domains parsed with count: 88
context_entities parsed with count: 29386
conversation_references parsed with count: 28231290
new_authors parsed with count: 23366
```

Počty záznamov získané sql dopytmi:

```
PS C:\Users\Leonard\Desktop\PDT_zadanie_1_code> python -u "c:\Users\Leonard\Desktop\PDT_zadanie_1_code\pdt_get_row_counts.py"
authors:                    5891810
conversations:              32347011
annotations:                19458972
links:                      11540247
hashtags:                   770064
conversation_hashtags:      54230164
context_annotations:        133935594
context_domains:            83
context_entities:           29299
conversation_references:    28230041
```

poznámka:
počet autorov je vyšší ako ten ktorý je reálne v súbore authors.jsonl pretože pri tweetoch ktoré nemali autora v tabuľke som pre nich vytváral nových autorov s prázdnymi stĺpcami, aby existovala relácia

A počet conversation_references je vyšší ako má byť lebo som sa ešte nezbavil tých pre ktoré neexistuje referencia na parent

Obsiahnuté zdrojové súbory a SQL dopyty

Súbory:

pdt_get_row_counts.py	- spočítava záznamy v tabuľkách cez SELECT COUNT(*)
pdt_hashtable.py	- objekt hashtable
PDT_pyspark.ipynb	- povodny jupyet notebook na ktorom som si testoval pyspark
pdt_table_create.py	- skript na vytvorenie každej z tabuliek
pdt_tweet_parser.py	- program ktorý vykonava import

V súbore pdt_get_row_counts.py sa nachádzajú takéto dopyty pre každú tabuľku:

```
sql = "SELECT COUNT(*) FROM authors"
cursor.execute(sql)
result = cursor.fetchone()
print("authors:\t\t\t" + str(result[0]))
```

V pdt_table_create.py sú nasledové typy dopytov:
Tento je pre tabuľku pre ktoru je manualne vkladané id

```
sql_authors ='''CREATE TABLE IF NOT EXISTS authors(
    id int8 PRIMARY KEY,
    name varchar(255),
    username varchar(255),
    description text,
    followers_count int4,
    following_count int4,
    tweet_count int4,
    listed_count int4
)'''
```

Tento je pre tabuľku s auto inkrementom id:

```
sql_links ='''CREATE TABLE IF NOT EXISTS links(
    id BIGSERIAL PRIMARY KEY,
    conversation_id int8 NOT NULL,
    url varchar(2048) NOT NULL,
    title text,
    description text
)'''
```

V súbore su nasledovné typy dopytov:
Dopyt v ktorom sa insertuje do všetkých stĺpcov

```
def insert_context_domains(conn, cursor, page_size, insert_vals):
    psycopg2.extras.execute_values(cursor, """
            INSERT INTO context_domains VALUES %s;
        """, ((
            item[0],
            item[1],
            item[2],
        ) for item in insert_vals), page_size=page_size)
    conn.commit()
```

Dopyt v ktorom sa insertuje do vybraných stĺpcov a id sa auto inkrementuje

```
def insert_conv_hashtags(conn, cursor, page_size, insert_vals):
    psycopg2.extras.execute_values(cursor, """
            INSERT INTO conversation_hashtags(conversation_id, hashtag_id) VALUES %s;
        """, ((
            hasht[0],
            hasht[1],
        ) for hasht in insert_vals), page_size=page_size)
    conn.commit()
```