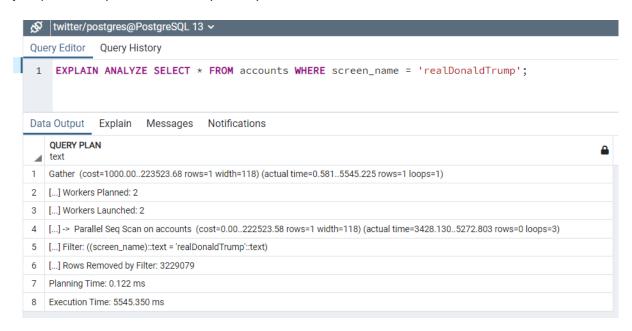
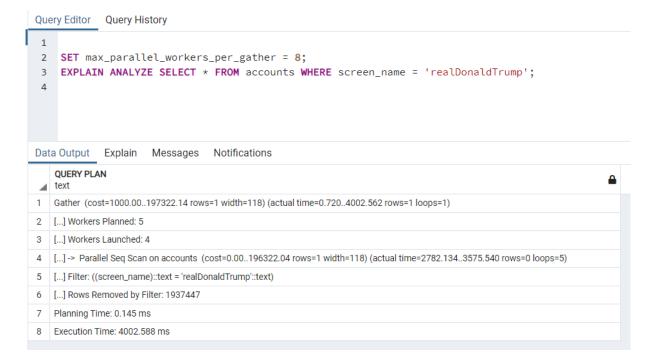
1. Plánovač vybral metódu "Parallel Seq Scan" – paralelné sekvenčné skenovanie. Pravdepodobne túto metódu vybral z dôvodu, že je rýchlejšia ako keby sme pozerali na konkrétny blok v úložisku, čo je zapríčinené tým, že nemáme vytvorený index.



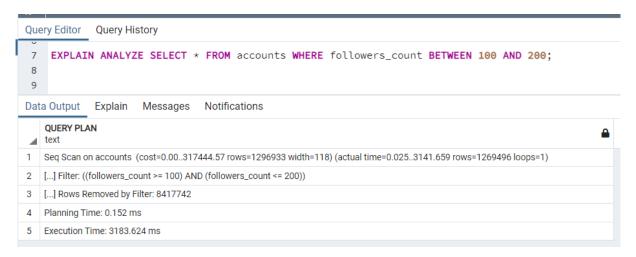
2. Na selecte z predošlej úlohy pracovali dvaja workery. Zdá sa, že zvýšenie počtu workerov neovplyvňuje čas – v mojom prípade trvalo vykonávanie query pred zvýšením počtu zväčša 4-5 sekúnd. Niekedy sa však query vykonalo podstatne rýchlejšie (1.7 – 2.1 sekundy). Podobné časy a podobné skoky vo vykonávaní nastávali aj pri zvýšení počtu workerov. Počet workerov je obmedzený podľa nastavenia tabuľky, respektíve servera.



3. Plánovač využije v tomto prípade jedného workera, pretože nad daným stĺpcom je vytvorený index a teda sa zmení aj metóda na "index scan" – tento prístup nie je paralelný a teda nevyužíva viac workerov. Zásadnú zmenu času ovplyvnilo vytvorenie indexu.

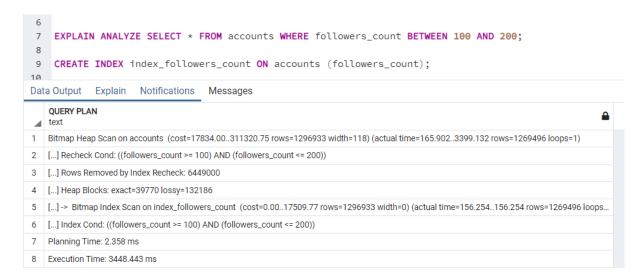


4. Správanie je iné ako v prvej úlohe. V tomto prípade bolo použité sekvenčné skenovanie. V prípade prvej úlohy išlo o **paralelné** sekvenčné skenovanie. Rozdiel oproti tretej úlohe je vo využití indexu – v tomto prípade nemáme vytvorený index a teda bolo použité sekvenčné skenovanie. Paralelné sa využíva v prípade malého počtu záznamov (resp. keď podmienka WHERE odfiltruje veľa záznamov). V takomto prípade sa využije viac workerov. V prípade veľkého počtu záznamov je zložité prenášať tieto záznamy medzi workermi. Nakoľko v tomto prípade pracujeme s veľkým počtom záznamov a nemáme vytvorený index, plánovač použil sekvenčné skenovanie.

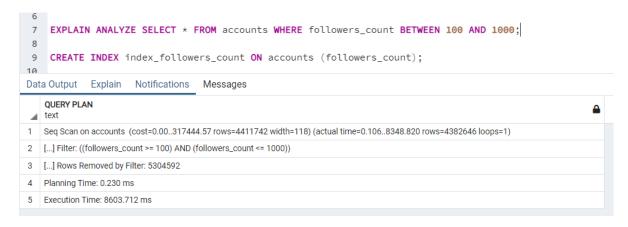


5. "Bitmap Index Scan" sa využíva, ak pracujeme s veľkým počtom záznamov. V takomto prípade sa neukladajú priamo dáta (konkrétne záznamy tabuľky), ale poloha potenciálnych záznamov (respektíve celé stránky). Následne "Bitmap Heap Scan" dekóduje tieto údaje, aby našiel konkrétne záznamy. "Recheck Cond" znamená, že tieto záznamy je potrebné znovu prekontrolovať.

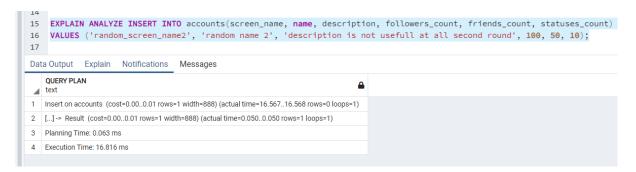
Za normálnych okolností si "work\_mem" ukladá hľadane záznamy. Ak nie je dostatočne veľká, namiesto záznamu si uloží celú stránku (page). Následne "Recheck Cond" znamená, že tieto stránky je potrebné znovu prekontrolovať, pretože sa v nich pochopiteľne nachádzajú aj také záznamy, ktoré nespĺňajú pôvodnú podmienku.

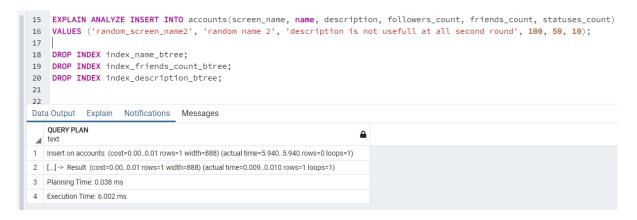


6. Rozdiel je v skenovaní – v tomto prípade bolo použité sekvenčné skenovanie a nie skenovanie pomocou indexu. Je to z dôvodu, že podmienku spĺňa veľa záznamov, teda je lepšie skenovať úložisko sekvenčne a nie podľa indexu, nakoľko skákanie po pamäti pomocou indexu je pomalšia (respektíve drahšia/náročnejšia) operácia. Z toho dôvodu je v konečnom dôsledku sekvenčné skenovanie rýchlejšie ako skenovanie s využitím indexu. Ak by podmienka selectu vyfiltrovala "dostatočný" počet záznamov, index by sa použil.



7. S použitím indexov trval insert 265ms. Bez indexov to trvalo 250ms. Proces som opakoval ešte raz a v tomto prípade trval insert s indexami 16 ms a bez indexov 6 ms. Výrazný časový rozdiel tu teda nevidím – môže sa jednať čisto o výkon môjho zariadenia. Avšak vo všeobecnosti by to pri indexoch malo trvať dlhšie, nakoľko sa okrem zápisu do tabuľky vykonáva aj zápis (respektíve určitá zmena) v indexoch a teda by to s indexami malo trvať dlhšie.





8. Vytvorenie indexu nad tweets.content je omnoho dlhšie, pretože zaindexovať integer je jednoduchšie ako zaindexovať varchar, respektíve text. Aj čo sa týka hlavičiek, hlavičky indexov integerov sú kratšie ako hlavičky indexov textu, čo sa všetko odráža na celkovom čase potrebnom pre vytvorenie indexu.

```
22
     CREATE INDEX index_retweet_count_btree ON tweets USING btree(retweet_count);
 23
     CREATE INDEX index_content_btree ON tweets USING btree(content);
 25
 Data Output
                   Notifications
           Explain
                                 Messages
 CREATE INDEX
 Query returned successfully in 1 min 15 secs.
23
    CREATE INDEX index_retweet_count_btree ON tweets USING btree(retweet_count);
24
    CREATE INDEX index_content_btree ON tweets USING btree(content);
25
Data Output
          Explain
                    Notifications
                                Messages
CREATE INDEX
Query returned successfully in 10 min 48 secs.
```

9.

10. S indexom alebo bez neho, v oboch prípadoch bol čas vykonania približne rovnaký. Dokonca bolo použité paralelné sekvenčné skenovanie. Jediný rozdiel bol v čase plánovania. V prípade pred vytvorením indexu bol čas plánovania niekoľkonásobne väčší.

Pred vytvorením indexu:

```
EXPLAIN ANALYZE SELECT t.* FROM tweets t WHERE t.content LIKE '%Gates%';

Data Output Explain Notifications Messages

QUERY PLAN
text

Gather (cost=1000.00..1152214.39 rows=3033 width=385) (actual time=3.001..46288.147 rows=111886 loops=1)

[...] Workers Planned: 7

[...] Workers Planned: 4

[...] -> Parallel Seq Scan on tweets t (cost=0.00..1150911.09 rows=433 width=385) (actual time=5.921..45981.093 rows=22377 loops=5)

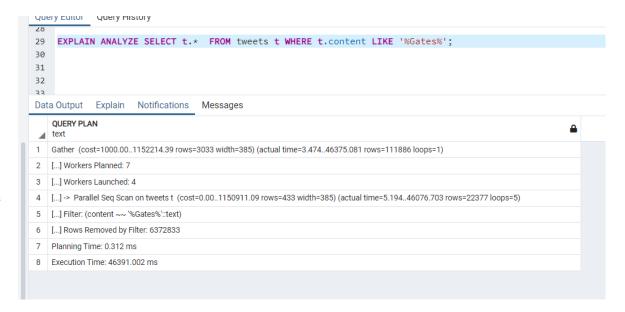
[...] Filter: (content ~~ '%Gates%'::text)

[...] Rows Removed by Filter: 6372833

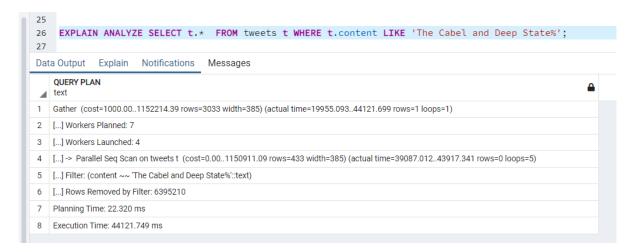
Planning Time: 8.341 ms

Execution Time: 46304.032 ms
```

## Po vytvorení indexu:



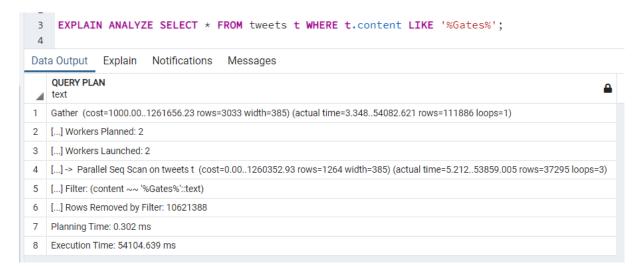
## 11. Nie, index sa nepoužil.



12. Použitie indexu ovplyvnilo pridanie parametru "text\_pattern\_ops". Vďaka tomuto parametru sa dáta voči vyhľadávanému výrazu porovnávajú po znakoch, čo je výhodou pri vyhľadávaní s použitím "LIKE".



Pri vyhľadávaní "%Gates%" sa index už nepoužil. Je to pravdepodobne pre to, že v predošlom prípade sme jasne definovali čím sa hľadaný výraz začína. V tomto prípade na hľadaný výraz z počiatku sedia všetky záznamy, tým pádom sa neoplatí používať index, ale plánovač porovnáva záznamy po poradí.



13. Nerozumiem aké funkcie mám opísať. EXPLAIN ANALYZE nad CREATE INDEX použiť nejde. Takisto ak použijem EXPLAIN ANALYZE nad selectom, žiadne funkcie to nevypíše. Pri vytváraní indexu je možné definovať aj obsah, respektíve hodnotu, ktorú chceme zaindexovať, pomocou podmienky WHERE:

```
34 CREATE INDEX index_content_btree_specific_qanon ON tweets USING btree(content) WHERE content ILIKE '%idiot #QAnon';

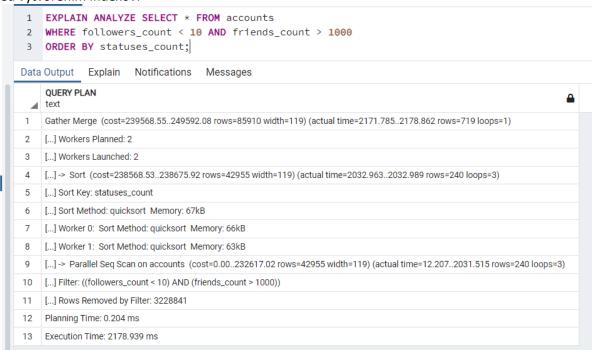
Data Output Explain Notifications Messages

CREATE INDEX

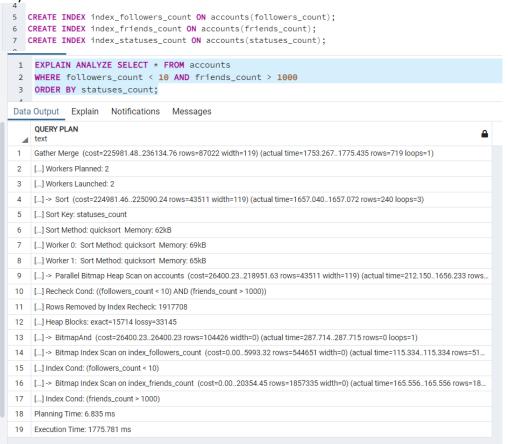
Query returned successfully in 8 min 52 secs.
```

14. Z obrázkov nižšie môžeme vidieť, že po vytvorení indexov sa použili indexy nad atribútmi followers\_count a friends\_count. Vytvárať index nad statuses\_count nemá zmysel, pretože tento atribút nijako nesúvisí s podmienkou WHERE, zatiaľ čo atribúty followers\_count a friends\_count súvisia a teda sa pre ne oplatí vytvárať index.

## Pred vytvorením indexov:



## Po vytvorení indexov:

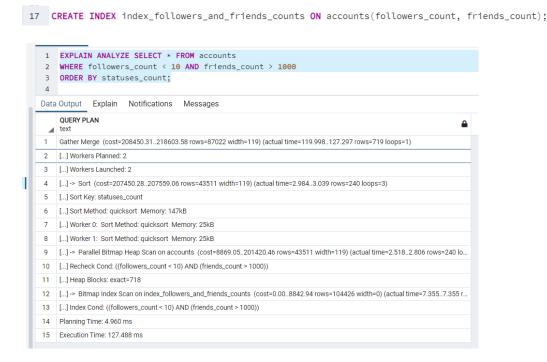


15. Použitie zloženého indexu výrazne zrýchlilo vyhľadávanie. Je to z dôvodu, že v podmienke sledujeme dva atribúty, pričom ak sú vytvorené dva indexy zvlášť na každý atribút, vyhľadávanie je pomalšie, ako keď vytvoríme zložený index nad oboma atribútmi. V prípade zloženého indexu sa indexujú rôzne "dvojice" hodnôt v týchto atribútoch. Ak teda vyhľadávame konkrétne čísla, vďaka zloženému indexu vieme rýchlo zistiť, ktoré záznamy spĺňajú podmienku. V prípade jednoduchých indexov musím pri vyhľadávaní využiť oba indexy a na záver spraviť prienik medzi vyhľadanými záznamami, aby sme zistili, ktoré zo záznamov spĺňajú obe podmienky.

S použitím zloženého indexu nad všetkými atribútmi:



S použitím zloženého indexu nad atribútmi followers\_count a friends\_count:



16. V prípade zmeny podmienky sa opäť použilo sekvenčné vyhľadávanie (paralelné), pretože podmienku spĺňa omnoho viac záznamov a teda sa neoplatí skákať v pamäti podľa indexu. Takisto aj celový čas vyhľadávania je výrazne dlhší.

