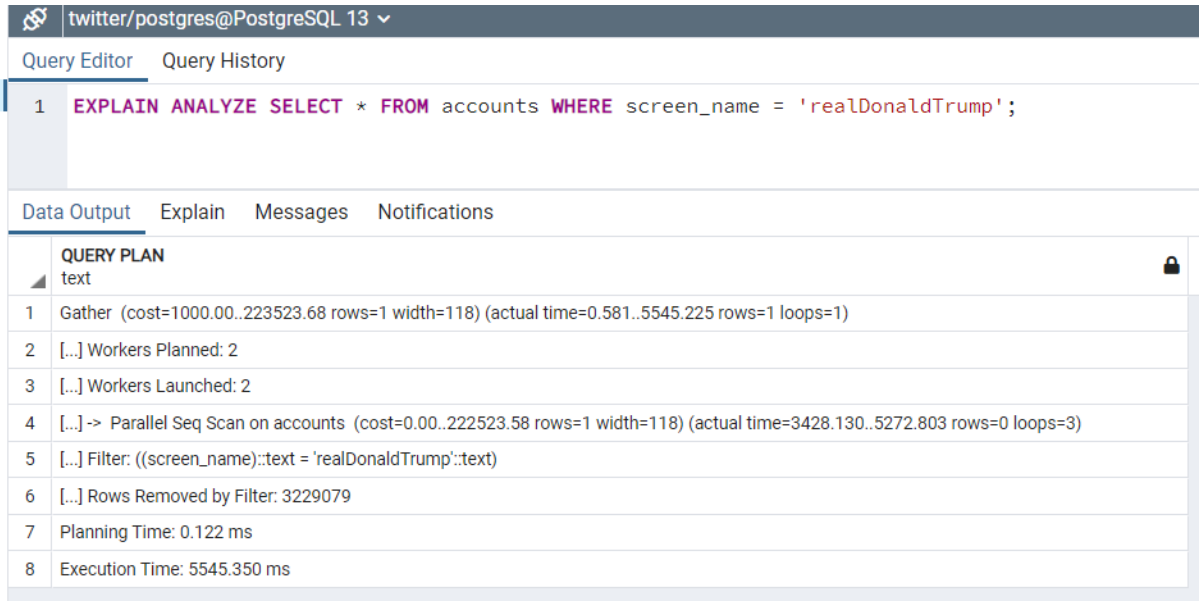
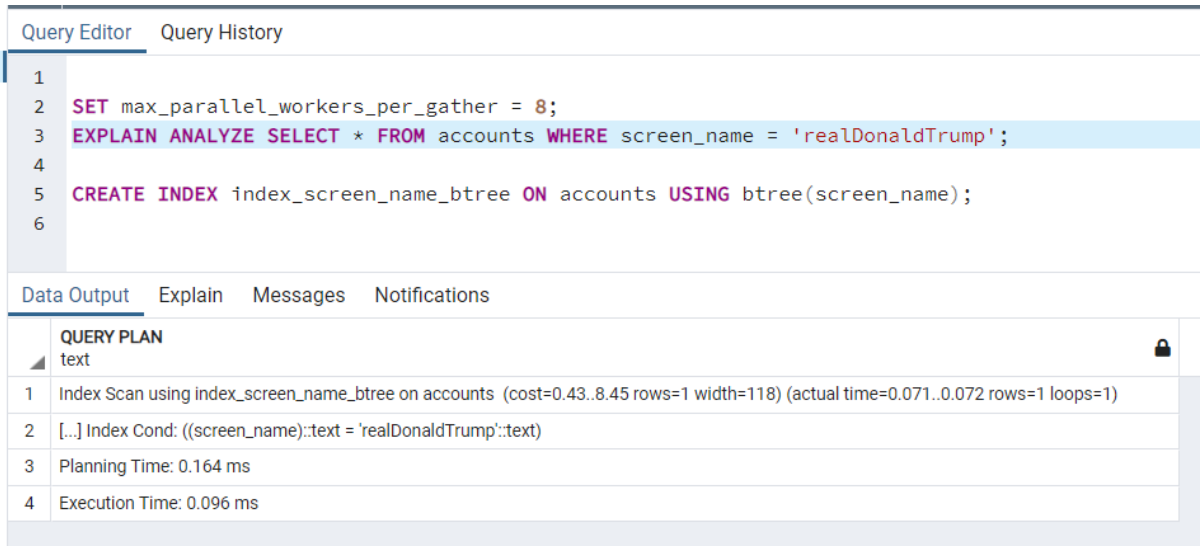
1. Plánovač vybral metódu „Parallel Seq Scan“ – paralelné sekvenčné skenovanie. Pravdepodobne túto metódu vybral z dôvodu, že je rýchlejšia ako keby sme pozerali na konkrétny blok v úložisku, čo je zapríčinené tým, že nemáme vytvorený index.

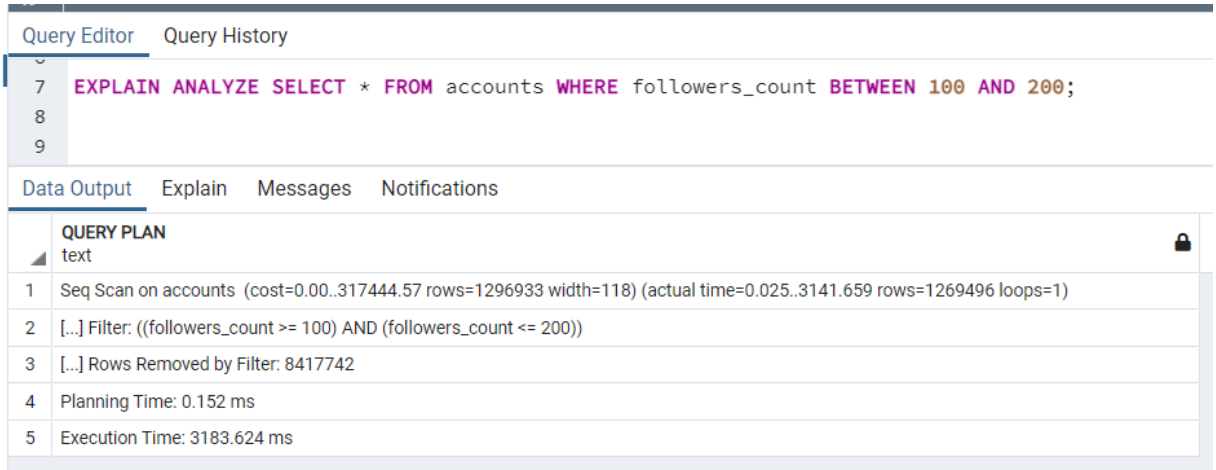


2. Na selecte z predošlej úlohy pracovali dvaja workery. Zdá sa, že zvýšenie počtu workerov neovplyvňuje čas – v mojom prípade trvalo vykonávanie query pred zvýšením počtu zväčša 4-5 sekúnd. Niekedy sa však query vykonalo podstatne rýchlejšie (1.7 – 2.1 sekundy). Podobné časy a podobné skoky vo vykonávaní 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
7  EXPLAIN ANALYZE SELECT * FROM accounts WHERE followers_count BETWEEN 100 AND 200;
8
9  CREATE INDEX index_followers_count ON accounts (followers_count);
10
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Bitmap Heap Scan on accounts (cost=17834.00..311320.75 rows=1296933 width=118) (actual time=165.902..3399.132 rows=1269496 loops=1)
2	[...] Recheck Cond: ((followers_count >= 100) AND (followers_count <= 200))
3	[...] Rows Removed by Index Recheck: 6449000
4	[...] Heap Blocks: exact=39770 lossy=132186
5	[...] -> Bitmap Index Scan on index_followers_count (cost=0.00..17509.77 rows=1296933 width=0) (actual time=156.254..156.254 rows=1269496 loops...
6	[...] Index Cond: ((followers_count >= 100) AND (followers_count <= 200))
7	Planning Time: 2.358 ms
8	Execution Time: 3448.443 ms

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.

```
6
7  EXPLAIN ANALYZE SELECT * FROM accounts WHERE followers_count BETWEEN 100 AND 1000;
8
9  CREATE INDEX index_followers_count ON accounts (followers_count);
10
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Seq Scan on accounts (cost=0.00..317444.57 rows=4411742 width=118) (actual time=0.106..8348.820 rows=4382646 loops=1)
2	[...] Filter: ((followers_count >= 100) AND (followers_count <= 1000))
3	[...] Rows Removed by Filter: 5304592
4	Planning Time: 0.230 ms
5	Execution Time: 8603.712 ms

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.

```
14
15 EXPLAIN ANALYZE INSERT INTO accounts(screen_name, name, description, followers_count, friends_count, statuses_count)
16 VALUES ('random_screen_name2', 'random name 2', 'description is not usefull at all second round', 100, 50, 10);
17
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Insert on accounts (cost=0.00..0.01 rows=1 width=888) (actual time=16.567..16.568 rows=0 loops=1)
2	[...] -> Result (cost=0.00..0.01 rows=1 width=888) (actual time=0.050..0.050 rows=1 loops=1)
3	Planning Time: 0.063 ms
4	Execution Time: 16.816 ms

```
15 EXPLAIN ANALYZE INSERT INTO accounts(screen_name, name, description, followers_count, friends_count, statuses_count)
16 VALUES ('random_screen_name2', 'random name 2', 'description is not usefull at all second round', 100, 50, 10);
17
18 DROP INDEX index_name_btree;
19 DROP INDEX index_friends_count_btree;
20 DROP INDEX index_description_btree;
21
22
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Insert on accounts (cost=0.00..0.01 rows=1 width=888) (actual time=5.940..5.940 rows=0 loops=1)
2	[...] -> Result (cost=0.00..0.01 rows=1 width=888) (actual time=0.009..0.010 rows=1 loops=1)
3	Planning Time: 0.038 ms
4	Execution Time: 6.002 ms

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
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
26
```

Data Output | Explain | Notifications | 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
26
```

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:

```
28
29  EXPLAIN ANALYZE SELECT t.*  FROM tweets t WHERE t.content LIKE '%Gates%';
30
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Gather (cost=1000.00..1152214.39 rows=3033 width=385) (actual time=3.001..46288.147 rows=111886 loops=1)
2	[...] Workers Planned: 7
3	[...] Workers Launched: 4
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)
5	[...] Filter: (content ~~ '%Gates%'::text)
6	[...] Rows Removed by Filter: 6372833
7	Planning Time: 8.341 ms
8	Execution Time: 46304.032 ms

Po vytvorení indexu:

Query Editor | Query History

```
29  EXPLAIN ANALYZE SELECT t.*  FROM tweets t WHERE t.content LIKE '%Gates%';
30
31
32
33
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Gather (cost=1000.00..1152214.39 rows=3033 width=385) (actual time=3.474..46375.081 rows=111886 loops=1)
2	[...] Workers Planned: 7
3	[...] Workers Launched: 4
4	[...] -> Parallel Seq Scan on tweets t (cost=0.00..1150911.09 rows=433 width=385) (actual time=5.194..46076.703 rows=22377 loops=5)
5	[...] Filter: (content ~~ '%Gates%'::text)
6	[...] Rows Removed by Filter: 6372833
7	Planning Time: 0.312 ms
8	Execution Time: 46391.002 ms

11. Nie, index sa nepoužil.

```
25
26  EXPLAIN ANALYZE SELECT t.*  FROM tweets t WHERE t.content LIKE 'The Cabel and Deep State%';
27
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Gather (cost=1000.00..1152214.39 rows=3033 width=385) (actual time=19955.093..44121.699 rows=1 loops=1)
2	[...] Workers Planned: 7
3	[...] Workers Launched: 4
4	[...] -> Parallel Seq Scan on tweets t (cost=0.00..1150911.09 rows=433 width=385) (actual time=39087.012..43917.341 rows=0 loops=5)
5	[...] Filter: (content ~~ 'The Cabel and Deep State%'::text)
6	[...] Rows Removed by Filter: 6395210
7	Planning Time: 22.320 ms
8	Execution Time: 44121.749 ms

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“.

```
1  CREATE INDEX index_content_btree_with_parameter ON tweets USING btree(content text_pattern_ops);
2
3  EXPLAIN ANALYZE SELECT * FROM tweets t WHERE t.content LIKE 'The Cable and Deep State%';
4
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Index Scan using index_content_btree_with_parameter on tweets t (cost=0.81..8.83 rows=3033 width=385) (actual time=0.019..0.020 row...
2	[...] Index Cond: ((content ~>=~ 'The Cable and Deep State'::text) AND (content ~<~ 'The Cable and Deep Statf'::text))
3	[...] Filter: (content ~~ 'The Cable and Deep State%'::text)
4	Planning Time: 0.580 ms
5	Execution Time: 0.045 ms

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.

```
3  EXPLAIN ANALYZE SELECT * FROM tweets t WHERE t.content LIKE '%Gates%';
4
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Gather (cost=1000.00..1261656.23 rows=3033 width=385) (actual time=3.348..54082.621 rows=111886 loops=1)
2	[...] Workers Planned: 2
3	[...] Workers Launched: 2
4	[...] -> Parallel Seq Scan on tweets t (cost=0.00..1260352.93 rows=1264 width=385) (actual time=5.212..53859.005 rows=37295 loops=3)
5	[...] Filter: (content ~~ '%Gates%'::text)
6	[...] Rows Removed by Filter: 10621388
7	Planning Time: 0.302 ms
8	Execution Time: 54104.639 ms

13. Nerozumiem aké funkcionality mám opísať. 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';
35
```

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:

```
1  EXPLAIN ANALYZE SELECT * FROM accounts
2  WHERE followers_count < 10 AND friends_count > 1000
3  ORDER BY statuses_count;
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Gather Merge (cost=239568.55..249592.08 rows=85910 width=119) (actual time=2171.785..2178.862 rows=719 loops=1)
2	[...] Workers Planned: 2
3	[...] Workers Launched: 2
4	[...] -> Sort (cost=238568.53..238675.92 rows=42955 width=119) (actual time=2032.963..2032.989 rows=240 loops=3)
5	[...] Sort Key: statuses_count
6	[...] Sort Method: quicksort Memory: 67kB
7	[...] Worker 0: Sort Method: quicksort Memory: 66kB
8	[...] Worker 1: Sort Method: quicksort Memory: 63kB
9	[...] -> Parallel Seq Scan on accounts (cost=0.00..232617.02 rows=42955 width=119) (actual time=12.207..2031.515 rows=240 loops=3)
10	[...] Filter: ((followers_count < 10) AND (friends_count > 1000))
11	[...] Rows Removed by Filter: 3228841
12	Planning Time: 0.204 ms
13	Execution Time: 2178.939 ms

Po vytvorení indexov:

```
5  CREATE INDEX index_followers_count ON accounts(followers_count);
6  CREATE INDEX index_friends_count ON accounts(friends_count);
7  CREATE INDEX index_statuses_count ON accounts(statuses_count);
```

```
1  EXPLAIN ANALYZE SELECT * FROM accounts
2  WHERE followers_count < 10 AND friends_count > 1000
3  ORDER BY statuses_count;
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Gather Merge (cost=225981.48..236134.76 rows=87022 width=119) (actual time=1753.267..1775.435 rows=719 loops=1)
2	[...] Workers Planned: 2
3	[...] Workers Launched: 2
4	[...] -> Sort (cost=224981.46..225090.24 rows=43511 width=119) (actual time=1657.040..1657.072 rows=240 loops=3)
5	[...] Sort Key: statuses_count
6	[...] Sort Method: quicksort Memory: 62kB
7	[...] Worker 0: Sort Method: quicksort Memory: 69kB
8	[...] Worker 1: Sort Method: quicksort Memory: 65kB
9	[...] -> Parallel Bitmap Heap Scan on accounts (cost=26400.23..218951.63 rows=43511 width=119) (actual time=212.150..1656.233 rows...
10	[...] Recheck Cond: ((followers_count < 10) AND (friends_count > 1000))
11	[...] Rows Removed by Index Recheck: 1917708
12	[...] Heap Blocks: exact=15714 lossy=33145
13	[...] -> BitmapAnd (cost=26400.23..26400.23 rows=104426 width=0) (actual time=287.714..287.715 rows=0 loops=1)
14	[...] -> Bitmap Index Scan on index_followers_count (cost=0.00..5993.32 rows=544651 width=0) (actual time=115.334..115.334 rows=51...
15	[...] Index Cond: (followers_count < 10)
16	[...] -> Bitmap Index Scan on index_friends_count (cost=0.00..20354.45 rows=1857335 width=0) (actual time=165.556..165.556 rows=18...
17	[...] Index Cond: (friends_count > 1000)
18	Planning Time: 6.835 ms
19	Execution Time: 1775.781 ms

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:

```
13  CREATE INDEX index_all_counts ON accounts(followers_count, friends_count, statuses_count);
```

```
1  EXPLAIN ANALYZE SELECT * FROM accounts
2  WHERE followers_count < 10 AND friends_count > 1000
3  ORDER BY statuses_count;
4
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Gather Merge (cost=212730.31..222883.58 rows=87022 width=119) (actual time=126.041..131.141 rows=719 loops=1)
2	[...] Workers Planned: 2
3	[...] Workers Launched: 2
4	[...] -> Sort (cost=211730.28..211839.06 rows=43511 width=119) (actual time=19.879..19.910 rows=240 loops=3)
5	[...] Sort Key: statuses_count
6	[...] Sort Method: quicksort Memory: 147kB
7	[...] Worker 0: Sort Method: quicksort Memory: 25kB
8	[...] Worker 1: Sort Method: quicksort Memory: 25kB
9	[...] -> Parallel Bitmap Heap Scan on accounts (cost=13149.05..205700.46 rows=43511 width=119) (actual time=15.521..19.609 rows=24...
10	[...] Recheck Cond: ((followers_count < 10) AND (friends_count > 1000))
11	[...] Heap Blocks: exact=718
12	[...] -> Bitmap Index Scan on index_all_counts (cost=0.00..13122.94 rows=104426 width=0) (actual time=46.268..46.268 rows=719 loops...
13	[...] Index Cond: ((followers_count < 10) AND (friends_count > 1000))
14	Planning Time: 5.927 ms
15	Execution Time: 131.226 ms

S použitím zloženého indexu nad atribútmi followers_count a friends_count:

```
17  CREATE INDEX index_followers_and_friends_counts ON accounts(followers_count, friends_count);
```

```
1  EXPLAIN ANALYZE SELECT * FROM accounts
2  WHERE followers_count < 10 AND friends_count > 1000
3  ORDER BY statuses_count;
4
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Gather Merge (cost=208450.31..218603.58 rows=87022 width=119) (actual time=119.998..127.297 rows=719 loops=1)
2	[...] Workers Planned: 2
3	[...] Workers Launched: 2
4	[...] -> Sort (cost=207450.28..207559.06 rows=43511 width=119) (actual time=2.984..3.039 rows=240 loops=3)
5	[...] Sort Key: statuses_count
6	[...] Sort Method: quicksort Memory: 147kB
7	[...] Worker 0: Sort Method: quicksort Memory: 25kB
8	[...] Worker 1: Sort Method: quicksort Memory: 25kB
9	[...] -> Parallel Bitmap Heap Scan on accounts (cost=8869.05..201420.46 rows=43511 width=119) (actual time=2.518..2.806 rows=240 lo...
10	[...] Recheck Cond: ((followers_count < 10) AND (friends_count > 1000))
11	[...] Heap Blocks: exact=718
12	[...] -> Bitmap Index Scan on index_followers_and_friends_counts (cost=0.00..8842.94 rows=104426 width=0) (actual time=7.355..7.355 r...
13	[...] Index Cond: ((followers_count < 10) AND (friends_count > 1000))
14	Planning Time: 4.960 ms
15	Execution Time: 127.488 ms

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.

```
1  EXPLAIN ANALYZE SELECT * FROM accounts
2  WHERE followers_count < 1000 AND friends_count > 1000
3  ORDER BY statuses_count;
4
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Gather Merge (cost=354743.53..484060.06 rows=1108350 width=119) (actual time=2316.997..2779.203 rows=740655 loops=1)
2	[...] Workers Planned: 2
3	[...] Workers Launched: 2
4	[...] -> Sort (cost=353743.51..355128.95 rows=554175 width=119) (actual time=2197.923..2343.245 rows=246885 loops=3)
5	[...] Sort Key: statuses_count
6	[...] Sort Method: external merge Disk: 38104kB
7	[...] Worker 0: Sort Method: external merge Disk: 35048kB
8	[...] Worker 1: Sort Method: external merge Disk: 35120kB
9	[...] -> Parallel Seq Scan on accounts (cost=0.00..232681.26 rows=554175 width=119) (actual time=0.470..1843.458 rows=246885 loops...
10	[...] Filter: ((followers_count < 1000) AND (friends_count > 1000))
11	[...] Rows Removed by Filter: 2982196
12	Planning Time: 0.169 ms
13	Execution Time: 2815.867 ms