Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Vyhľadávanie a indexovanie

Martin Schön, Bc.

AIS ID: 103121

E-mail: xschon@stuba.sk

GitHub repozitár: https://github.com/FIIT-DBS/zadanie-pdt-xSchon

Predmet: Pokročilé databázové technológie

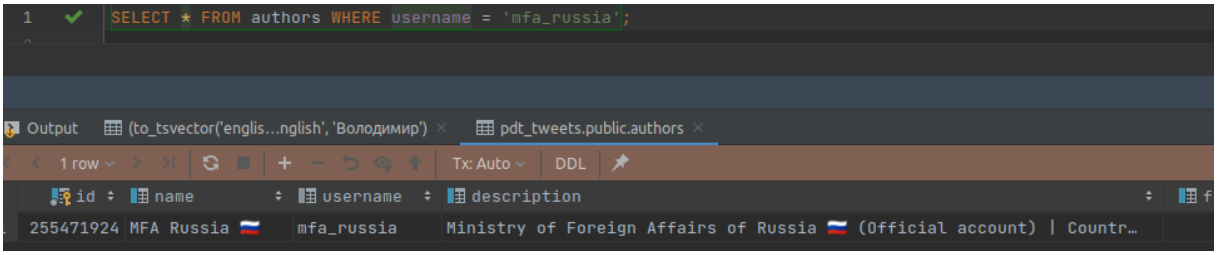
Zimný semester 2022/2023

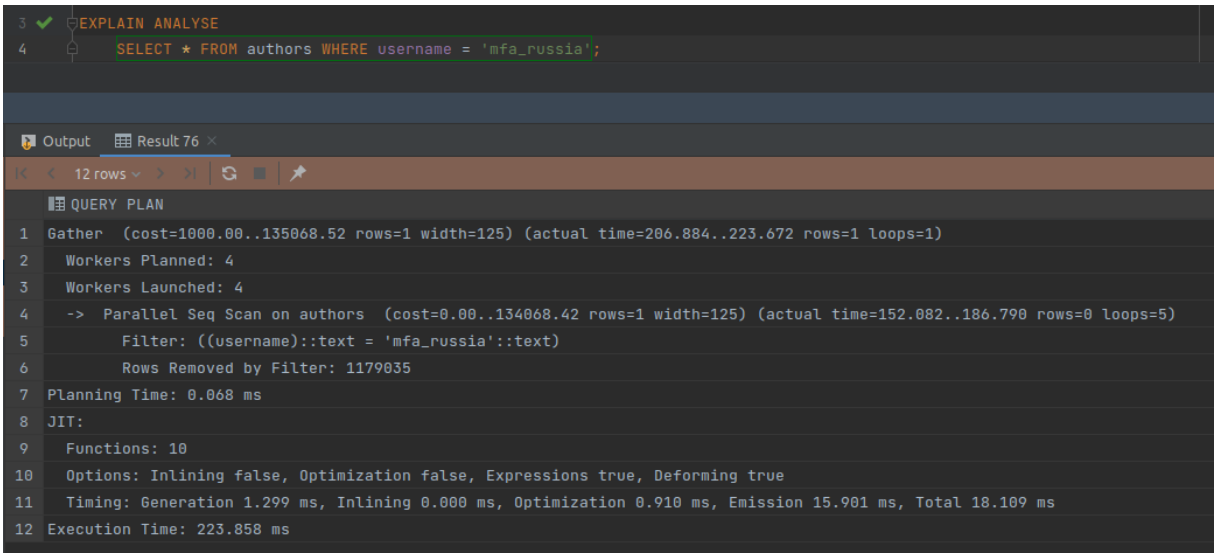
Martin Schön

Vyhľadávanie a indexovanie

1. Vyhľadanie mfa_russia 2
2. Workeri 2
3. Btree index autori 4
4. Followers count 4
5. Index nad followers countom 5
6. 3 Btree indexy insert 9
7. Dĺžka vytvárania indexu 10
8. Porovnanie indexov 10
9. Conversations content meno “Gates” 13
10. “There are no excuses…” tweet 15
11. Index na vyhľadávanie podľa konca reťazca 18
12. Jednoduché indexy nad countami 19
13. Zložený index nad countami 20
14. Index pre Putin a New World Order cez GiST a GIN 21
15. Linky darujme.sk 23
16. Query pomocou FTS 24
Záver 26

1. Vyhľadanie mfa_russia



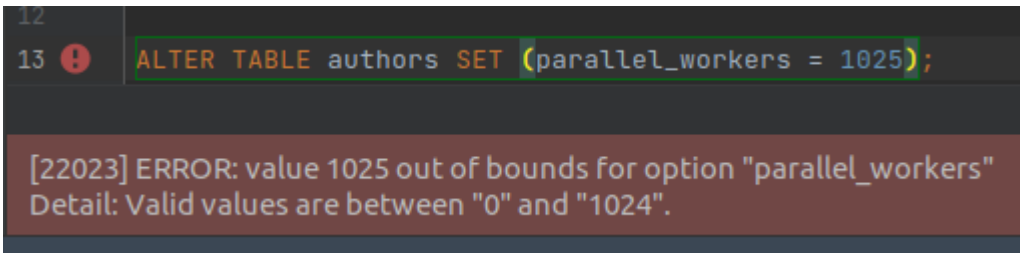


Plánovač vybral paralelný sekvenčný sken. Bez indexov nemá pomôcku a musí tak prehľadať všetky riadky tabuľky - čo sa vykonáva sekvenčným skenom. Ideálne je, ak je možné prehľadávať viacero častí DB na disku súčasne - preto paralelný. DB je rozdelená na bloky, ktoré následne jednotliví paralelizovaní workeri prehľadávajú sekvenčne.

2. Workeri

Výsledok selectu je rovnaký, ako bol pri úlohe 1 - počet workerov nič nemení. Na selecte pracovali 4 workeri, je to spôsobené tým, že v súbore postgresql.conf mám pre túto DB nastavený parameter *max_parallel_workers_per_gather* = 4. Toto číslo som si nastavoval pri zrýchľovaní databázy v zadaní 1, defaultne je nastavené na hodnotu 2. Využili sa všetci pracovníci na vhodný paralelný sken (spomínaný v bode 1).

Maximálne nastaviteľné číslo paralelných pracovníkov na query je 1024 - na základe obmedzení postgreSQL. Minimálne sa dá vybrať 0 workerov, kedy query nebude parlelizovaná - bude to sekvenčný scan.



Parameter *parallel_workers* (počet workerkov spustiteľných pre procesy danej tabuľky) tabuľky a *max_parallel_workers_per_gather* spoločne s *max_parallel_workers* (počet workerov, ktoré DB podporí) databázy spolu úzko súvisia. *parallel_workers* je

podradené databázovému parametru. Query teda v praxi využije maximálne taký počet workerov, ktorý predstavuje menšie číslo z dvoch spomínaných parametrov. Ak chcem zmeniť *max_parallel_workers_per_gather,* dá sa to pomocou:
`ALTER SYSTEM SET max_parallel_workers_per_gather = 64;` avšak následne je potrebné reštartovať DB. Toto všetko mi teda určí **maximálny** počet workerov, čo ale neznamená, že ich query planner vždy všetkých využije. Proces určovania poćtov je taktiež ovplyvnený cez *max_worker_processes* (default 8), určuje počet background procesov, ktoré zvládne systém podporiť. Ideálne je nastaviť ho na hodnotu *max_parallel_workers.* Rovnako tak je dobré nastaviť aj *max_parallel_maintenance_workers*, ktoré určuje maximálny počet workerov, ktoré môžu byť spustené jedným utility príkazom.

Porovnanie rýchlostí podľa počtu workerov

12 rows

	QUERY PLAN
1	Gather (cost=1000.00..116718.06 rows=1 width=125) (actual time=375.984..4243.039 rows=1 loops=1)
2	Workers Planned: 1024
3	Workers Launched: 1023
4	-> Parallel Seq Scan on authors (cost=0.00..115717.96 rows=1 width=125) (actual time=10.219..10.499 rows=0 loops=1024)
5	Filter: ((username)::text = 'mfa_russia'::text)
6	Rows Removed by Filter: 5757
7	Planning Time: 0.061 ms
8	JIT:
9	Functions: 2048
10	Options: Inlining false, Optimization false, Expressions true, Deforming true
11	Timing: Generation 1076.040 ms, Inlining 0.000 ms, Optimization 33.943 ms, Emission 996.222 ms, Total 2106.204 ms
12	Execution Time: 4243.213 ms

12 rows

	QUERY PLAN
1	Gather (cost=1000.00..125857.31 rows=1 width=125) (actual time=168.257..184.050 rows=1 loops=1)
2	Workers Planned: 8
3	Workers Launched: 8
4	-> Parallel Seq Scan on authors (cost=0.00..124857.21 rows=1 width=125) (actual time=121.377..135.808 rows=0 loops=9)
5	Filter: ((username)::text = 'mfa_russia'::text)
6	Rows Removed by Filter: 655019
7	Planning Time: 0.090 ms
8	JIT:
9	Functions: 18
10	Options: Inlining false, Optimization false, Expressions true, Deforming true
11	Timing: Generation 2.870 ms, Inlining 0.000 ms, Optimization 1.896 ms, Emission 61.555 ms, Total 66.321 ms
12	Execution Time: 184.283 ms

Väčší počet workerov nemusí nevyhnutne znamenať lepší (rýchlejší) beh dopytu. Pri 8 workeroch som dosiahol lepší čas, ako pri 4, ale aj 16. Viac workerov viac zaťažuje CPU, teda ich efektívne využitie závisí od typu a počet jadier CPU. Okrem toho - úzke hrdlo práce s databázami spravidla býva nedostatočne rýchly prístup na disk (úložisko dát), čo viac workerov nevyrieši.

3. Btree index autori

```
CREATE INDEX authors_usernames_btree_index ON authors
    USING btree
    (username);
EXPLAIN ANALYSE
    SELECT * FROM authors WHERE username = 'mfa_russia';
```

Output | Result 105 — 4 rows

```
QUERY PLAN
Index Scan using authors_usernames_btree_index on authors  (cost=0.43..2.65 rows=1 width=125) (actual time=0.022..0.023 rows=1 loops=1)
  Index Cond: ((username)::text = 'mfa_russia'::text)
Planning Time: 0.060 ms
Execution Time: 0.037 ms
```

Po vytvoreni indexu je vyhľadávanie násobne rýchlejšie, vďaka využitiu index scanu. Ten nepoužíva viacero workerov, index scan neprechádza cez všetky dáta uložené v DB, ale namiesto toho používa prechádzanie cez indexy uložené b B-tree štruktúre. Vďaka tomu, že bol vyhľadávaný iba jeden záznam, bolo toto prechádzanie veľmi rýchle - strom bolo potrebné prejsť iba raz. Vyhľadávanie v Btree strome má časovú náročnosť O(log n) - teda máme veľmi veľkú rýchlosť výmenou za pamäť potrebnú na uloženie indexu.

Výsledok zostáva nemenný, keďže sa mení prístup k databáze, ale nie obsah požiadavky, ani tabuľky v databáze:

```
SELECT * FROM authors WHERE username = 'mfa_russia';
```

Output | pdt_tweets.public.authors — 1 row — Tx: Auto — DDL

	id	name	username	description	followers_count	following_count	tweet_count	listed_count
1	255471924	MFA Russia 🇷🇺	mfa_russia	Ministry of Foreign Affairs of Ru…	556912	1443	69213	5300

4. Followers count

```
EXPLAIN ANALYZE
    SELECT * FROM authors WHERE (followers_count >= 100) AND (followers_count <= 200);
```

Output | Result 121 — 12 rows

```
QUERY PLAN
Gather  (cost=1000.00..201791.08 rows=740885 width=125) (actual time=4.088..260.776 rows=760088 loops=1)
  Workers Planned: 8
  Workers Launched: 8
  ->  Parallel Seq Scan on authors  (cost=0.00..126702.58 rows=92611 width=125) (actual time=7.512..173.337 rows=84454 loops=9)
        Filter: ((followers_count >= 100) AND (followers_count <= 200))
        Rows Removed by Filter: 570565
Planning Time: 0.082 ms
JIT:
  Functions: 18
  Options: Inlining false, Optimization false, Expressions true, Deforming true
  Timing: Generation 4.569 ms, Inlining 0.000 ms, Optimization 2.564 ms, Emission 64.755 ms, Total 71.889 ms
Execution Time: 298.284 ms
```

```
EXPLAIN ANALYZE
SELECT * FROM authors WHERE (followers_count >= 100) AND (followers_count <= 120);
```

Output | Result 122

12 rows

```
QUERY PLAN
Gather  (cost=1000.00..145660.38 rows=179578 width=125) (actual time=3.857..241.020 rows=199937 loops=1)
  Workers Planned: 8
  Workers Launched: 8
  ->  Parallel Seq Scan on authors  (cost=0.00..126702.58 rows=22447 width=125) (actual time=5.603..163.693 rows=22215 loops=9)
        Filter: ((followers_count >= 100) AND (followers_count <= 120))
        Rows Removed by Filter: 632804
Planning Time: 0.050 ms
JIT:
  Functions: 18
  Options: Inlining false, Optimization false, Expressions true, Deforming true
  Timing: Generation 4.245 ms, Inlining 0.000 ms, Optimization 2.396 ms, Emission 47.737 ms, Total 54.377 ms
Execution Time: 251.045 ms
```

Pri zvýšenom počte workerov nie je rozdiel - pri 8 prebehne pre oboch paralelný sekvenčný sken, akurát s rôznou rýchlosťou.

Tipujem ale, že pointou tohto zadania bola ukázať fakt, že pri rozptyle 100 až 200 bude použitý sekvenčný sken. Ak nastavím workerov na 4 (prípadne rozsah na 100 až 250), tak sa ukáže sekvenčný sken. To sa deje preto, že query chce vrátiť veľký počet záznamov - odhaduje približne 740000 riadkov. Pri takomto množstve by získavanie stránok daných záznamov bolo najefektívnejšie, ak pristupujem sekvenčným skenom - má zmysel prehľadávať celú databázu a tak získať všetky vyhovujúce záznamy.

```
ALTER TABLE authors SET (parallel_workers = 4);
EXPLAIN ANALYZE
    SELECT * FROM authors WHERE (followers_count >= 100) AND (followers_count <= 200);
```

Output | Result 140

9 rows

```
QUERY PLAN
Seq Scan on authors  (cost=0.00..204098.66 rows=740885 width=125) (actual time=2.459..616.114 rows=760088 loops=1)
  Filter: ((followers_count >= 100) AND (followers_count <= 200))
  Rows Removed by Filter: 5135088
Planning Time: 0.048 ms
JIT:
  Functions: 2
  Options: Inlining false, Optimization false, Expressions true, Deforming true
  Timing: Generation 0.172 ms, Inlining 0.000 ms, Optimization 0.143 ms, Emission 2.303 ms, Total 2.617 ms
Execution Time: 638.067 ms
```

5. Index nad followers countom

```
DROP INDEX authors_followers_count_index;
CREATE INDEX authors_followers_count_index ON authors
    USING btree
    (followers_count);

EXPLAIN ANALYZE
    SELECT * FROM authors WHERE followers_count BETWEEN 100 AND 200;
```

Output | Result 159

8 rows

```
QUERY PLAN
Index Scan using authors_followers_count_index on authors  (cost=0.43..142749.57 rows=748676 width=125) (actual time=0.021..464.313 rows=760088 loops=1)
  Index Cond: ((followers_count >= 100) AND (followers_count <= 200))
Planning Time: 0.060 ms
JIT:
  Functions: 2
  Options: Inlining false, Optimization false, Expressions true, Deforming true
  Timing: Generation 0.338 ms, Inlining 0.000 ms, Optimization 0.000 ms, Emission 0.000 ms, Total 0.338 ms
Execution Time: 482.505 ms
```

```
EXPLAIN ANALYZE
    SELECT * FROM authors WHERE followers_count BETWEEN 100 AND 120;
```

Output | Result 160

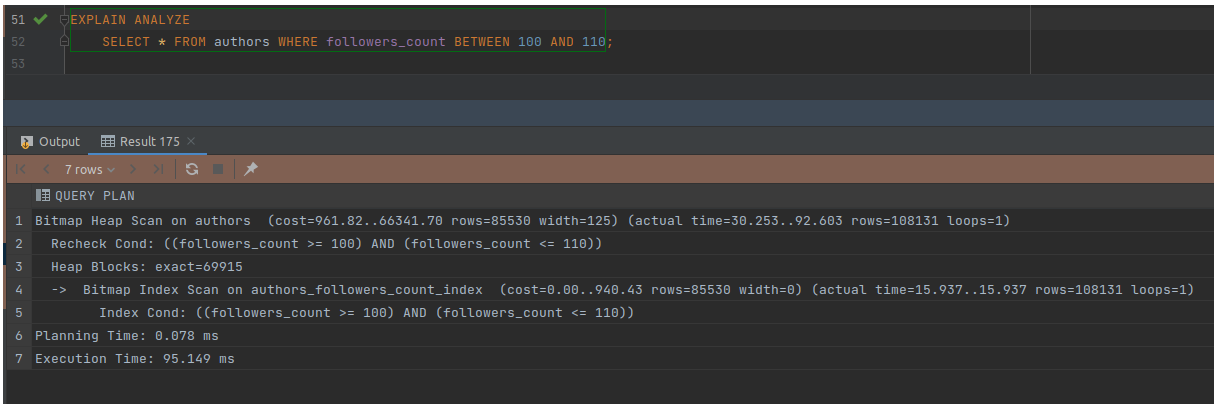
8 rows

```
QUERY PLAN
Index Scan using authors_followers_count_index on authors  (cost=0.43..114944.38 rows=179527 width=125) (actual time=0.020..133.780 rows=199937 loops=1)
  Index Cond: ((followers_count >= 100) AND (followers_count <= 120))
Planning Time: 0.058 ms
JIT:
  Functions: 2
  Options: Inlining false, Optimization false, Expressions true, Deforming true
  Timing: Generation 0.229 ms, Inlining 0.000 ms, Optimization 0.000 ms, Emission 0.000 ms, Total 0.229 ms
Execution Time: 138.847 ms
```

Bitmap Index Scan je vhodný na použitie pre výber takého počtu prvkov, ktorý je priveľký pre index, ale zároveň primalý pre sekvenčný sken. Každá stránka je zobrazená iba raz. Bitmap Heap sken funguje podobne ako sekvenčný sken, ale navštívi len tie stránky, ktoré obsahujú potrebné dáta - čo sa hodí pre zrýchlenie procesu. Klasický index prechádza riadok po riadku a navštevuje všetky stránky, ktoré potrebuje pre daný riadok, koľkokrát potrebuje za beh programu. Bitmap index scan získa zoznam stránok a z každej vytiahne vhodné riadky- teda vzniká recheck cond.

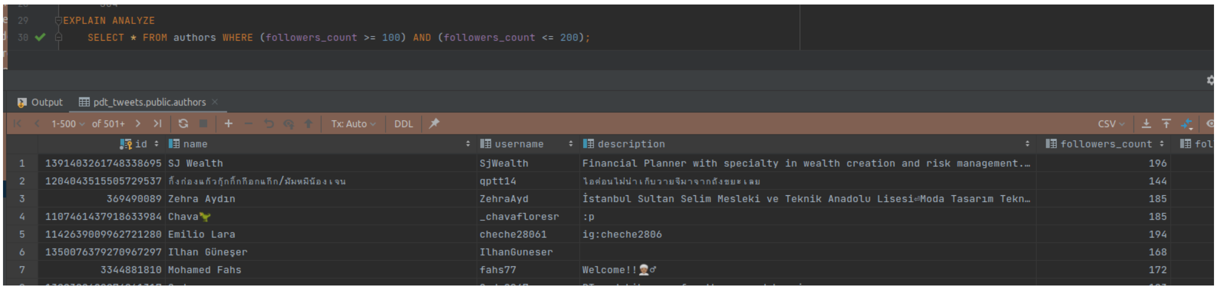
Recheck cond sa dostáva do funkcie, keď by bitmapa bola priveľká - bitmapa si začne pamätať, aké stránky obsahujú potrebné hodnoty, namiesto toho aby si pamätala hodnoty samotné. Keď sa to stane, recheck condition kontroluje všetky hodnoty na zapamätanej stránke a vytiahne z nej vhodné hodnoty.
Inými slovami - sken vyberie stránky na ktorých si myslí, že budú správne hodnoty a recheck condition potom skontroluje, či táto domnienka platí a či sú hodnoty naozaj správne.

Na dosiahnutie Bitmap Heap scan som potreboval znížiť rozsah na <100, 110>, pretože pri väčšom rozsahu nefungoval.

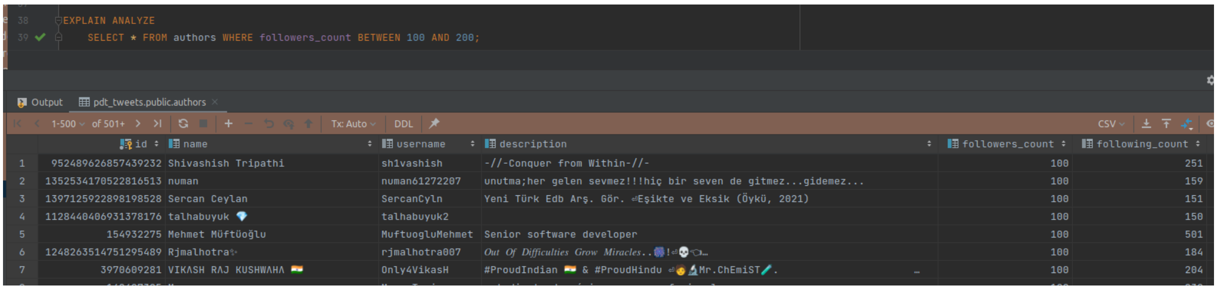


Výsledky queries pre bod 4 a 5:

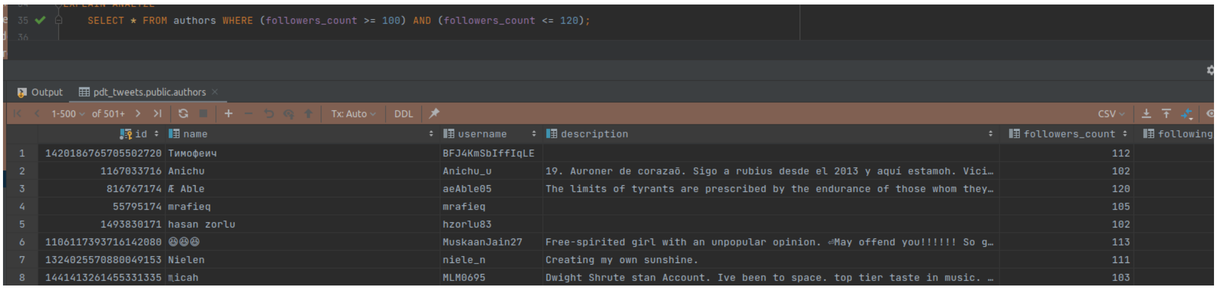
<100, 200> bez indexu



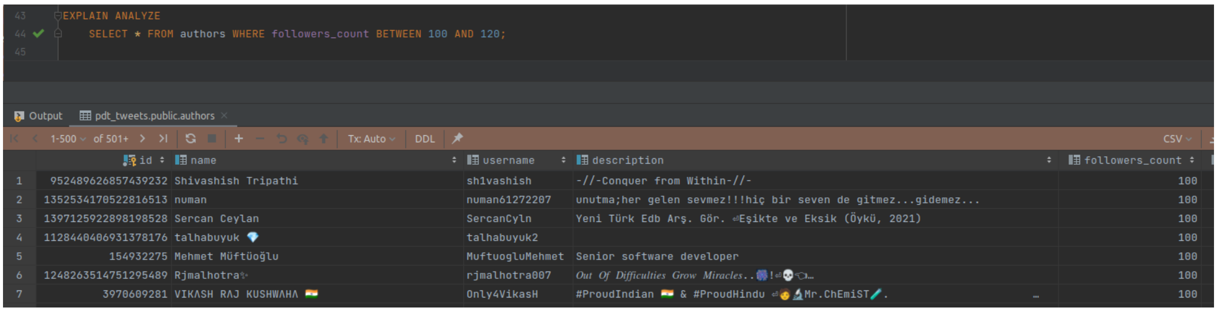
<100, 200> s indexom



<100, 120> bez indexu



<100, 200> s indexom



Pri výsledkoch queries je možné vidieť, že výsledky sú v rôznom poradí - výsledky samotné sa nemenia, ale ich poradie môže byť rôzne, nakoľko každý systém pristupuje k pamäti iným spôsobom a teda získa dáta v inom poradí.

Na dôkaz tohto tvrdenia sú v odovzdaní priložené aj dva csv súbory obsahujúce výstupy rozmedzia <100, 120> pre indexovanú aj neindexovanú query: *z02_04_index_100_120_schon.csv* a *z02_04_noindex_100_120_schon.csv.*

6. 3 Btree indexy insert

Vkladanie s existujúcimi indexami:

```
69 ✓ EXPLAIN ANALYZE
70   INSERT INTO authors (id, name, username, description,
71                        followers_count, following_count, tweet_count, listed_count)
72   VALUES (8, 'Martin Schon', 'xschon',
73           'Ziak ktory dostane Acko z PDT <3', 7, 3, 358, 2);
```

i authors_names_btree_index

Output | Result 29

4 rows

	QUERY PLAN
1	Insert on authors (cost=0.00..0.01 rows=0 width=0) (actual time=1.103..1.104 rows=0 loops=1)
2	-> Result (cost=0.00..0.01 rows=1 width=1088) (actual time=0.001..0.002 rows=1 loops=1)
3	Planning Time: 0.033 ms
4	Execution Time: 1.126 ms

Vkladanie po dropnutí indexov:

```
83 ✓ EXPLAIN ANALYZE
84   INSERT INTO authors (id, name, username, description,
85                        followers_count, following_count, tweet_count, listed_count)
86   VALUES (10, 'Martin Schon', 'xschon',
87           'Ziak ktory dostane Acko z PDT <3', 7, 3, 358, 2);
```

Output | Result 36

4 rows

	QUERY PLAN
1	Insert on authors (cost=0.00..0.01 rows=0 width=0) (actual time=0.022..0.022 rows=0 loops=1)
2	-> Result (cost=0.00..0.01 rows=1 width=1088) (actual time=0.001..0.001 rows=1 loops=1)
3	Planning Time: 0.021 ms
4	Execution Time: 0.031 ms

Vkladanie do tabuľky v ktorej neexistuju indexy je násobne rýchlejšie. Akonáhle totižto vkladám dáta do indexovanej tabuľky, je potrebné zaindexovať aj vložené hodnoty - čo zaberie čas. Pokiaľ vkladám záznam do tabuľky bez indexov, stačí nájsť na disku miesto iba pre takýto záznam - teda kľudne prvé vyhovujúce. Pre každý index je potom tiež potrebné nájsť miesto. A vo finále - pre btree indexy je potrebné správne uložiť index. To znamená nájsť správne miesto v správnom liste - ak nie je, je potrebné rozdeliť listy a vytvoriť nové. Následne je potrebné tiež overiť vybalancovanie stromu, keďže btree strom musí byť vybalancovaný podľa jeho definície.

7. Dĺžka vytvárania indexu

```
pdt_tweets.public> CREATE INDEX conversations_retweet_count_index ON conversations
                   USING btree
                       (retweet_count)
[2022-10-17 10:55:47] completed in 43 s 920 ms
pdt_tweets.public> CREATE INDEX conversations_content_index ON conversations
                   USING btree
                       (content)
[2022-10-17 10:59:24] completed in 3 m 9 s 6 ms
```

Ako vidno, index nad kontentom konverzácií sa vytvára 4-násobne pomalšie. Dĺžka vytvárania indexu je ovplyvnená viacerými faktormi - softvérom, konfiguráciou DB, hardvérom, počtom riadkov tabuľky, vyťaženosťou zariadenia,... V tomto prípade je ale väčšina týchto "externých" parametrov rovnaká.

Rozdiel medzi contentom a retweet_countom je ten, že retweet_count obsahuje jedno číslo pre každý riadok tabuľky. Pri contente ale máme dlhé vety, ktoré obsahujú desiatky slov, ktoré je celé potrebné uložiť do listov stromu a je potrebné k ním spraviť cestu. Je potrebné nájsť väčšie bloky voľnej pamäte, nakoľko by sa dlhá veta nezmestila na menšie miesto. Preto je takéto vytváranie násobne pomalšie a veľkosť indexu kontentu je väčšia, ako pre retweet_county - preto, že dlhý text zaberie viac miesta ako číslo. Taktiež sú čísla na porovnanie jednoduchšie, ako porovnávanie stringov, kde treba porovnať viacero znakov na správne zaradenie do stromu.

8. Porovnanie indexov

Conversation content:

```
111 ✓ SELECT tree_level, index_size, root_block_no, internal_pages, leaf_pages FROM pgstatindex( relname: 'conversations_content_index');
```

Output | Result 120

1 row

	tree_level	index_size	root_block_no	internal_pages	leaf_pages
1	5	2517655552	245302	12253	295077

```
112 ✓ SELECT type, avg_item_size, page_size FROM bt_page_stats( relname: 'conversations_content_index', blkno: 3);
```

Output | Result 155

1 row

	type	avg_item_size	page_size
1	i	204	8192

```
113 ✓ SELECT type, avg_item_size, page_size FROM bt_page_stats( relname: 'conversations_content_index', blkno: 10003);
```

Output | Result 156

1 row

	type	avg_item_size	page_size
1	i	233	8192

```
SELECT type, avg_item_size, page_size FROM bt_page_stats( relname: 'conversations_content_index', blkno: 1);
```

Output | Result 157

1 row

	type	avg_item_size	page_size
1	l	214	8192

```
SELECT type, avg_item_size, page_size FROM bt_page_stats( relname: 'conversations_content_index', blkno: 1515);
```

Output | Result 158

1 row

	type	avg_item_size	page_size
1	l	312	8192

Conversation retweet count:

```
SELECT tree_level, index_size, root_block_no, internal_pages, leaf_pages FROM pgstatindex( relname: 'conversations_retweet_count_index');
```

Output | Result 159

1 row

	tree_level	index_size	root_block_no	internal_pages	leaf_pages
1	2	225804288	209	136	27427

```
SELECT type, avg_item_size, page_size FROM bt_page_stats( relname: 'conversations_retweet_count_index', blkno: 3);
```

Output | Result 167 | Result 166

1 row

	type	avg_item_size	page_size
1	i	23	8192

```
SELECT type, avg_item_size, page_size FROM bt_page_stats( relname: 'conversations_retweet_count_index', blkno: 10003);
```

Output | Result 167 | Result 168

1 row

	type	avg_item_size	page_size
1	l	729	8192

```
SELECT type, avg_item_size, page_size FROM bt_page_stats( relname: 'conversations_retweet_count_index', blkno: 1);
```

Output | Result 169 | Result 168

1 row

	type	avg_item_size	page_size
1	l	729	8192

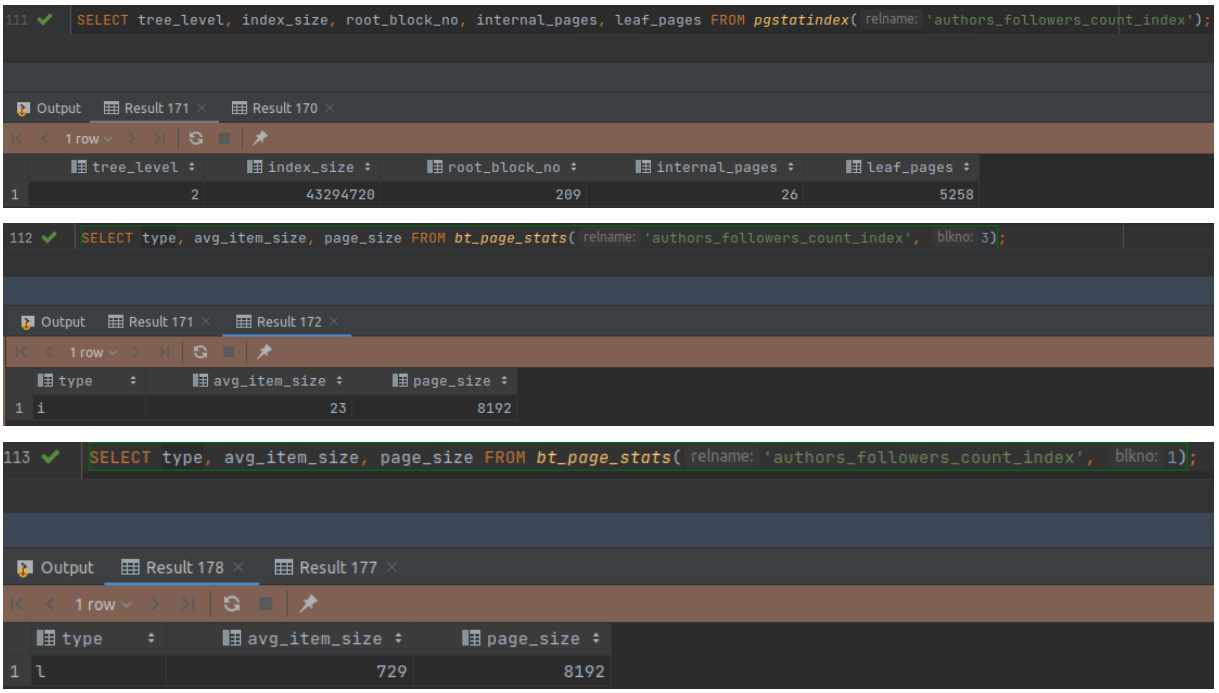
```
SELECT type, avg_item_size, page_size FROM bt_page_stats( relname: 'conversations_retweet_count_index', blkno: 1515);
```

Output | Result 169 | Result 170

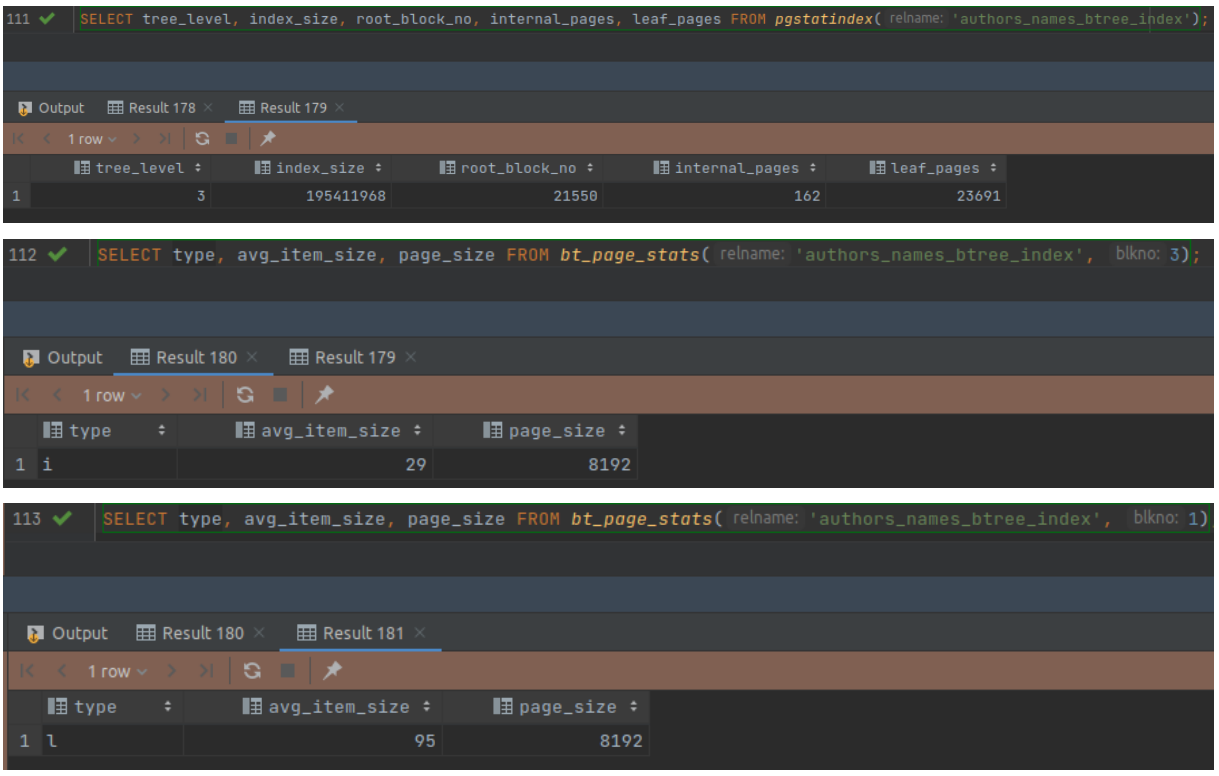
1 row

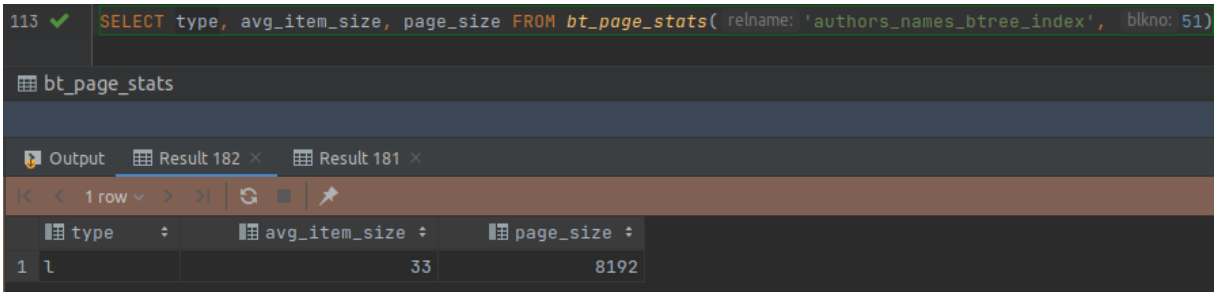
	type	avg_item_size	page_size
1	l	729	8192

Authors followers count:



Authors name:





Vysvetlenie:

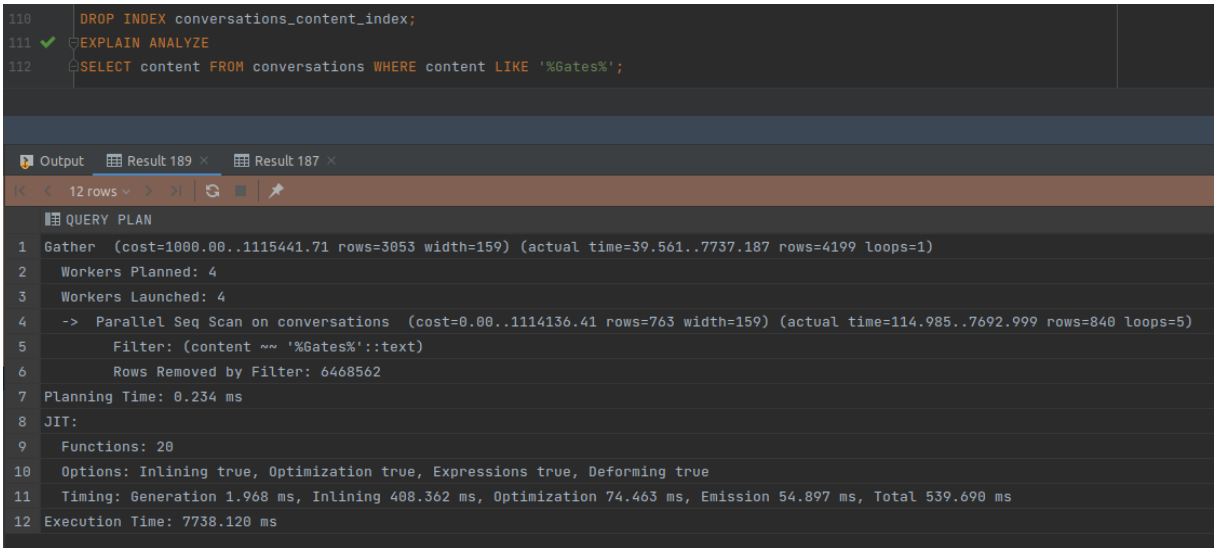
Level stromu sa líši v závislosti od veľkosti indexu - konkrétne závisí od počtu nódov, ktoré sa v strome nachádzajú.Tieto stromy sú veľmi široké, ale nie až tak veľmi hlboké - vidíme, že sa dostaneme na level 5 pri conversations retweets. Pri menších indexoch vznikajú menej hlboké stromy.

V stromoch sa nachádzajú dva rôzne typy nódov - inner nodes (i), ktoré slúžia na prechádzanie stromom - obsahujú rozhodovacie keys, nie teda samotné values. Hodnoty z databázy sa nachádzajú v listoch (i), teda na poslednej vrstve. Počet vnútorných nodes (v screenshotoch ako root_block_no) je tiež závislý od veľkosti vkladaných hodnôt.

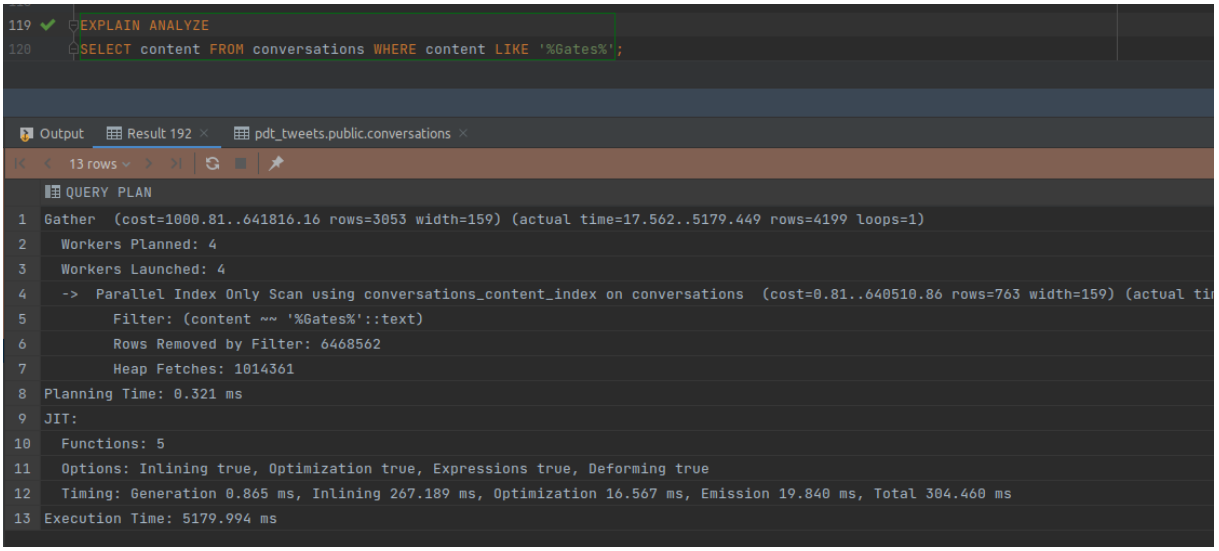
Priemerná veľkosť predmetu zapísaného v indexe sa líši podľa toho, či sa jedná o numerickú hodnotu, alebo o texty. V prípade inner nódov pri stringoch dostanem rôzne hodnoty, podľa veľkosti textu a naplnenia nódu. Pri číslach má každý kľúč preddefinovanú veľkosť - počet bitov potrebných na uloženie čísla v DB. Pri leaf nódoch platí to isté - textové polia sa budú pohybovať, zatiaľčo čísla potrebujú jednotnú veľkosť.

9. Conversations content meno “Gates”

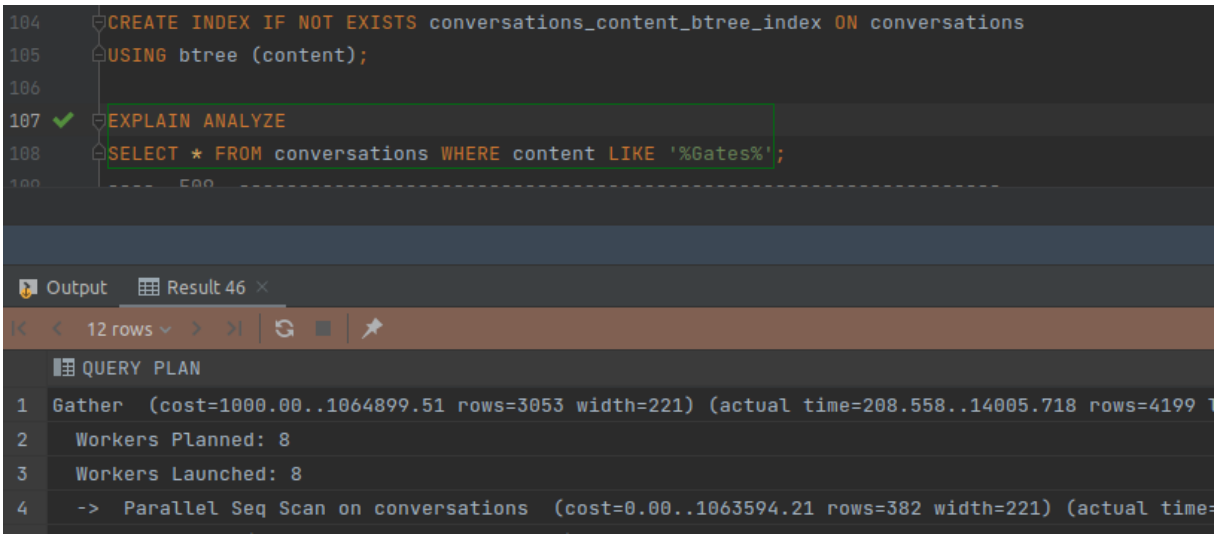
Bez indexu:



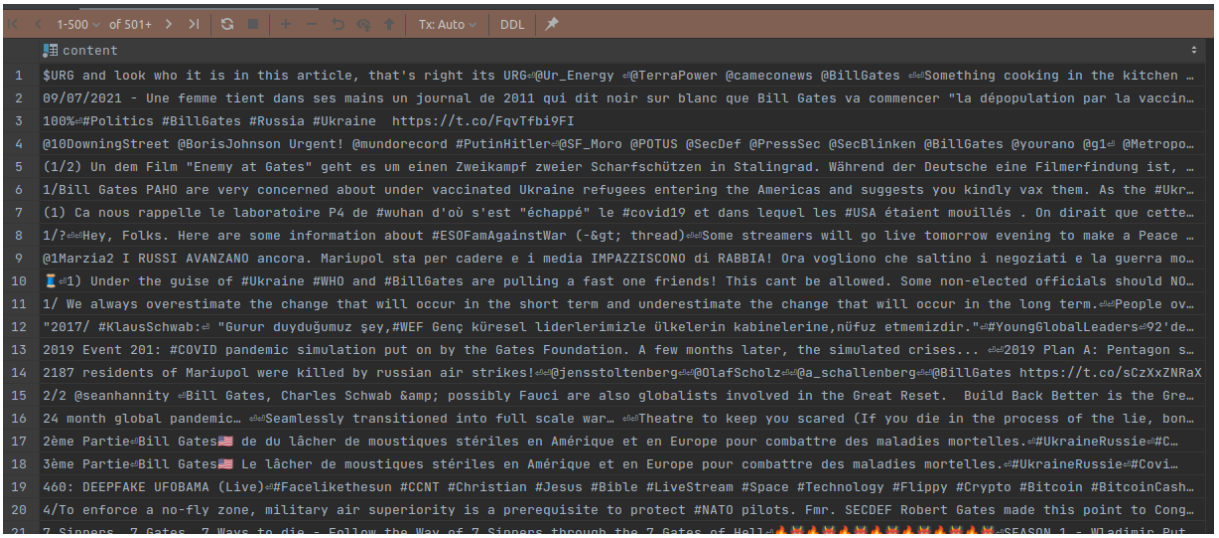
S indexom:



Pri práci so zaindexovanými dátami prebehlo použitie “paralel index only” skenu. Ten je v aktuálnej verzii postgresu podporovaný iba pre btree indexy. Funguje tak, že každý worker vezme jeden blok indexu, prejde ním a vráti všetky platné hodnoty v danom bloku. Tento prístup sa ale použije iba v prípade, že hľadám content - ak by som vyhľadával všetky stĺpce tabuľky (nie len zaindexovaný content), tak použijem paralelný sekvenčný sken, hoci počet riadkov ostane rovnaký. Index sa nedá použiť, ak slovo nie je na začiatku stringu.



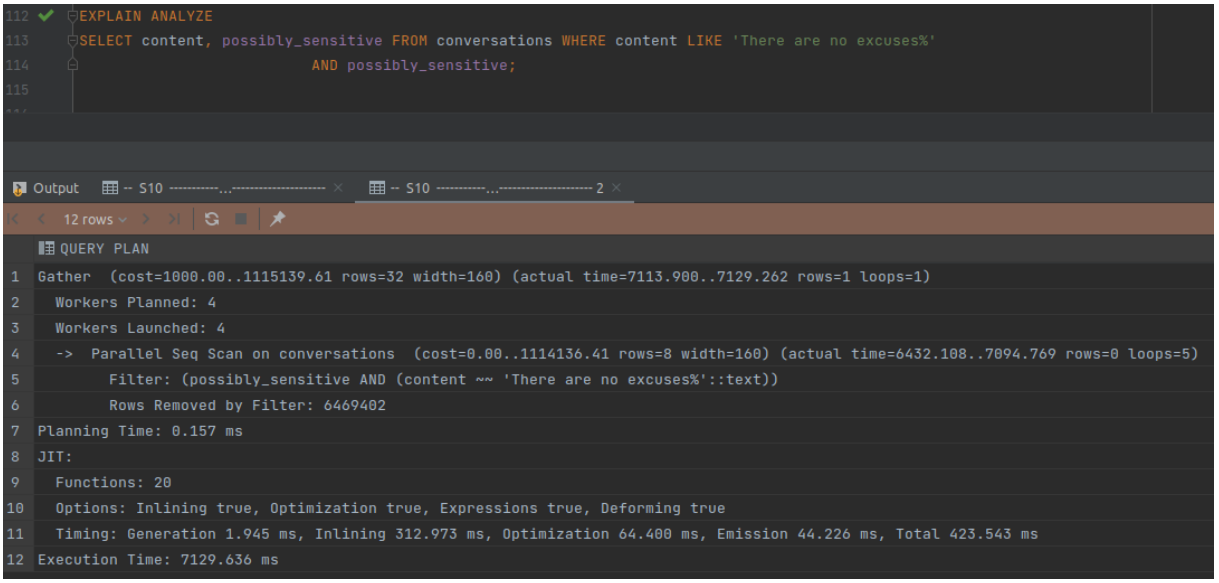
Výsledky:



	content
1	$URG and look who it is in this article, that's right its URG⏎@Ur_Energy ⏎@TerraPower @cameconews @BillGates ⏎⏎Something cooking in the kitchen …
2	09/07/2021 - Une femme tient dans ses mains un journal de 2011 qui dit noir sur blanc que Bill Gates va commencer "la dépopulation par la vaccin…
3	100%⏎⏎#Politics #BillGates #Russia #Ukraine https://t.co/FqvTfbi9FI
4	@10DowningStreet @BorisJohnson Urgent! @mundorecord #PutinHitler⏎@SF_Moro @POTUS @SecDef @PressSec @SecBlinken @BillGates @yourano @g1⏎ @Metropo…
5	(1/2) Un dem Film "Enemy at Gates" geht es um einen Zweikampf zweier Scharfschützen in Stalingrad. Während der Deutsche eine Filmerfindung ist, …
6	1/Bill Gates PAHO are very concerned about under vaccinated Ukraine refugees entering the Americas and suggests you kindly vax them. As the #Ukr…
7	(1) Ca nous rappelle le laboratoire P4 de #wuhan d'où s'est "échappé" le #covid19 et dans lequel les #USA étaient mouillés . On dirait que cette…
8	1/?⏎⏎Hey, Folks. Here are some information about #ESOFamAgainstWar (-> thread)⏎⏎Some streamers will go live tomorrow evening to make a Peace …
9	@1Marzia2 I RUSSI AVANZANO ancora. Mariupol sta per cadere e i media IMPAZZISCONO di RABBIA! Ora vogliono che saltino i negoziati e la guerra mo…
10	🧵⏎1) Under the guise of #Ukraine #WHO and #BillGates are pulling a fast one friends! This cant be allowed. Some non-elected officials should NO…
11	1/ We always overestimate the change that will occur in the short term and underestimate the change that will occur in the long term.⏎⏎People ov…
12	"2017/ #KlausSchwab:⏎ "Gurur duyduğumuz şey,#WEF Genç küresel liderlerimizle ülkelerin kabinelerine,nüfuz etmemizdir."⏎⏎#YoungGlobalLeaders⏎92'de…
13	2019 Event 201: #COVID pandemic simulation put on by the Gates Foundation. A few months later, the simulated crises... ⏎⏎2019 Plan A: Pentagon s…
14	2187 residents of Mariupol were killed by russian air strikes!⏎⏎@jensstoltenberg⏎⏎@OlafScholz⏎⏎@a_schallenberg⏎⏎@BillGates https://t.co/sCzXxZNRaX
15	2/2 @seanhannity ⏎Bill Gates, Charles Schwab & possibly Fauci are also globalists involved in the Great Reset. Build Back Better is the Gre…
16	24 month global pandemic… ⏎⏎Seamlessly transitioned into full scale war… ⏎⏎Theatre to keep you scared (If you die in the process of the lie, bon…
17	2ème Partie⏎Bill Gates🇺🇸 de du lâcher de moustiques stériles en Amérique et en Europe pour combattre des maladies mortelles.⏎⏎#UkraineRussie⏎#C…
18	3ème Partie⏎Bill Gates🇺🇸 Le lâcher de moustiques stériles en Amérique et en Europe pour combattre des maladies mortelles.⏎⏎#UkraineRussie⏎#Covi…
19	460: DEEPFAKE UFOBAMA (Live)⏎⏎#Facelikethesun #CCNT #Christian #Jesus #Bible #LiveStream #Space #Technology #Flippy #Crypto #Bitcoin #BitcoinCash…
20	4/To enforce a no-fly zone, military air superiority is a prerequisite to protect #NATO pilots. Fmr. SECDEF Robert Gates made this point to Cong…
21	7 Sinners. 7 Gates. 7 Ways to die - Follow the Way of 7 Sinners through the 7 Gates of Hell🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥⏎⏎SEASON 1 - Wladimir Put…

Ako som už vyššie uvádzal, výsledky sa pre nezaindexované tabuľky nemenia, maximálne tak poradie. Všetky contenty je možné nájsť v súbore *z02_09_schon.csv*.

10. “There are no excuses…” tweet



```
112  EXPLAIN ANALYZE
113  SELECT content, possibly_sensitive FROM conversations WHERE content LIKE 'There are no excuses%'
114                           AND possibly_sensitive;
115
```

Output | -- S10 ----------...---------- | -- S10 ----------...---------- 2

12 rows

	QUERY PLAN
1	Gather (cost=1000.00..1115139.61 rows=32 width=160) (actual time=7113.900..7129.262 rows=1 loops=1)
2	Workers Planned: 4
3	Workers Launched: 4
4	-> Parallel Seq Scan on conversations (cost=0.00..1114136.41 rows=8 width=160) (actual time=6432.108..7094.769 rows=0 loops=5)
5	Filter: (possibly_sensitive AND (content ~~ 'There are no excuses%'::text))
6	Rows Removed by Filter: 6469402
7	Planning Time: 0.157 ms
8	JIT:
9	Functions: 20
10	Options: Inlining true, Optimization true, Expressions true, Deforming true
11	Timing: Generation 1.945 ms, Inlining 312.973 ms, Optimization 64.400 ms, Emission 44.226 ms, Total 423.543 ms
12	Execution Time: 7129.636 ms

Index sa nepoužil, pretože dopyt obsahoval podmienku cez nezaindexovaný stĺpec.

Ak by som používal samotné vyhľadávanie cez There are no excuses… na začiatku vety, tak takýto dopyt dokáže Btree index spracovať:

```
EXPLAIN ANALYZE
SELECT content FROM conversations WHERE content LIKE 'There are no excuses%';
```

Output | -- S10 ---------...---------- | Result 58

13 rows

```
QUERY PLAN
Gather  (cost=1000.81..641816.16 rows=3053 width=159) (actual time=2327.895..2414.086 rov
  Workers Planned: 4
  Workers Launched: 4
  ->  Parallel Index Only Scan using conversations_content_btree_index on conversations
```

Výsledok:

```
---- S10 ----------
EXPLAIN ANALYZE
SELECT content, possibly_sensitive FROM conversations WHERE content LIKE 'There are no excuses%'
                    AND possibly_sensitive;
---- F10 ----------
```

Output | pdt_tweets.public.conversations

1 row | Tx: Auto | DDL

content	possibly_sensitive
There are no excuses, zero, non, zilch, nada.↵The US and NATO have been arming scum, 14000 dead by Zelensky and Azov's hand. No amount of ap…	true

Tento výsledok bude zhodný pre všetky následovné riešenia bez ohľadu na indexáciu.

Zlepšiť dopyt môžeme tak, že

a) pridáme index aj nad possibly sensitive

```
EXPLAIN ANALYZE
SELECT content, possibly_sensitive FROM conversations WHERE content LIKE '%There are no excuses%'
                    AND possibly_sensitive;
```

Output | pdt_tweets.public.conversations | Result 213

9 rows

```
QUERY PLAN
Gather  (cost=1000.44..29739.28 rows=32 width=160) (actual time=11636.611..11694.081 rows=1 loops=1)
  Workers Planned: 4
  Workers Launched: 4
  ->  Parallel Index Scan using conversations_sensitive_index on conversations  (cost=0.44..28736.08 rows=8 width=160) (actual time=11440.334..11501.667 rows=0 loops=5)
        Index Cond: (possibly_sensitive = true)
        Filter: (content ~~ '%There are no excuses%'::text)
        Rows Removed by Filter: 71664
Planning Time: 0.416 ms
Execution Time: 11694.106 ms
```

Ak mám dva indexy nad dvoma stĺpcami, čas sa zlepší približne o 15%.

b) Vytvorenie jedného, zloženého indexu nad množinou (content, possibly_sensitive)

```
EXPLAIN ANALYZE
SELECT content, possibly_sensitive FROM conversations WHERE content LIKE '%There are no excuses%'
                        AND possibly_sensitive;
```

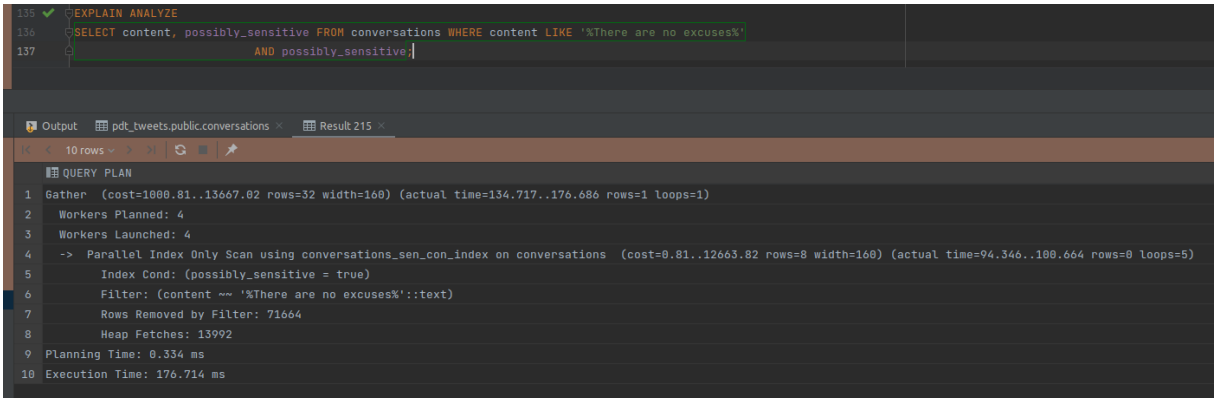
Output | pdt_tweets.public.conversations | Result 218

9 rows

```
QUERY PLAN
Gather  (cost=1000.44..29739.28 rows=32 width=160) (actual time=6995.803..7043.608 rows=1 loops=1)
  Workers Planned: 4
  Workers Launched: 4
  ->  Parallel Index Scan using conversations_sensitive_index on conversations  (cost=0.44..28736.08 rows=8 width=160) (actual time=6946.019..6947.617 rows=0 loops=5)
        Index Cond: (possibly_sensitive = true)
        Filter: (content ~~ '%There are no excuses%'::text)
        Rows Removed by Filter: 71664
Planning Time: 0.313 ms
Execution Time: 7043.625 ms
```

Týmto sa dostanem na hodnotu približne 7 sekúnd, čo je opäť zlepšenie oproti rozdeleným indexom.

c) Najrýchlejšie riešenie - vytvorenie zloženému indexu nad dvoma stĺpcami v poradí (possibly_sensitive, content)



```
EXPLAIN ANALYZE
SELECT content, possibly_sensitive FROM conversations WHERE content LIKE '%There are no excuses%'
                        AND possibly_sensitive;
```

Output | pdt_tweets.public.conversations | Result 215

10 rows

```
QUERY PLAN
Gather  (cost=1000.81..13667.02 rows=32 width=160) (actual time=134.717..176.686 rows=1 loops=1)
  Workers Planned: 4
  Workers Launched: 4
  ->  Parallel Index Only Scan using conversations_sen_con_index on conversations  (cost=0.81..12663.82 rows=8 width=160) (actual time=94.346..100.664 rows=0 loops=5)
        Index Cond: (possibly_sensitive = true)
        Filter: (content ~~ '%There are no excuses%'::text)
        Rows Removed by Filter: 71664
        Heap Fetches: 13992
Planning Time: 0.334 ms
Execution Time: 176.714 ms
```

Takéto riešenie, podobne ako b) používa Paralelný index scan. Toto riešenie je však neporovnateľne najrýchlejšie, vďaka štýlu query a typu, akým vyberáme hodnoty. Tento index najskôr zahodí všetky výsledky, ktoré nie sú possibly_sensitive - tie tvoria 98.9% výsledkov. Vďaka tomu je vyhľadávanie nad contentom extrémne rýchle. Samozrejme, tento konkrétny výsledok dostanem len pre takto zadanú úlohu - ak by som napríklad mal vyhľadať tie, ktoré NIE SÚ possibly_sensitive, tak stále ostane preskúmať skoro všetky riadky a teda rýchlosť nebude takto závratne rýchlejšia.

11. Index na vyhľadávanie podľa konca reťazca

```
146  CREATE INDEX reverse_content_index ON conversations
147  USING btree
148      (reverse(content));
149
150  EXPLAIN ANALYZE
151  SELECT * FROM conversations WHERE reverse(content) LIKE reverse('%https://t.co/pkFwLXZlEm');
```

Output | pdt_tweets.public.conversations | Result 265

12 rows

```
QUERY PLAN
1  Gather  (cost=1000.00..1151526.79 rows=161735 width=221) (actual time=12764.635..12800.690 rows=1 loops=1)
2    Workers Planned: 4
3    Workers Launched: 4
4    ->  Parallel Seq Scan on conversations  (cost=0.00..1134353.29 rows=40434 width=221) (actual time=11844.330..12737.073 rows=0 loops=5)
5          Filter: (reverse(content) ~~ 'mElZXLwFkp/oc.t//:sptth%'::text)
6          Rows Removed by Filter: 6469402
7  Planning Time: 0.063 ms
8  JIT:
9    Functions: 10
10   Options: Inlining true, Optimization true, Expressions true, Deforming true
11   Timing: Generation 1.792 ms, Inlining 342.935 ms, Optimization 60.366 ms, Emission 27.797 ms, Total 432.889 ms
12 Execution Time: 12800.914 ms
```

V tejto úlohe je podstatné správne vytvoriť index. Btree indexy nedokážu vyhľadávať values podľa konečných hodnôt, je potrebné správať sa k tomuto zadaniu tak, ako by znenie bolo "...tweet, ktorý začína reťazcom…". Preto teda otočím celý content. Pri vytváraní indexu použijem reverse nad kontentom, teda sa mi zaindexuje naopak. text_pattern_ops je potom dobrým parametrom, pretože podporuje kontrolovanie po charakteroch poľa textu (preto *text*_pattern_ops).

Následne pri SELECTe vyberám otočený content a porovnávam ho s otočeným textom, keďže chcem nájsť link v tomto tvare na začiatku textu contentu.

Výsledok je len jeden:

```
146  SELECT * FROM conversations WHERE reverse(content) LIKE reverse('%https://t.co/pkFwLXZlEm');
147
148
```

i reverse_content_index

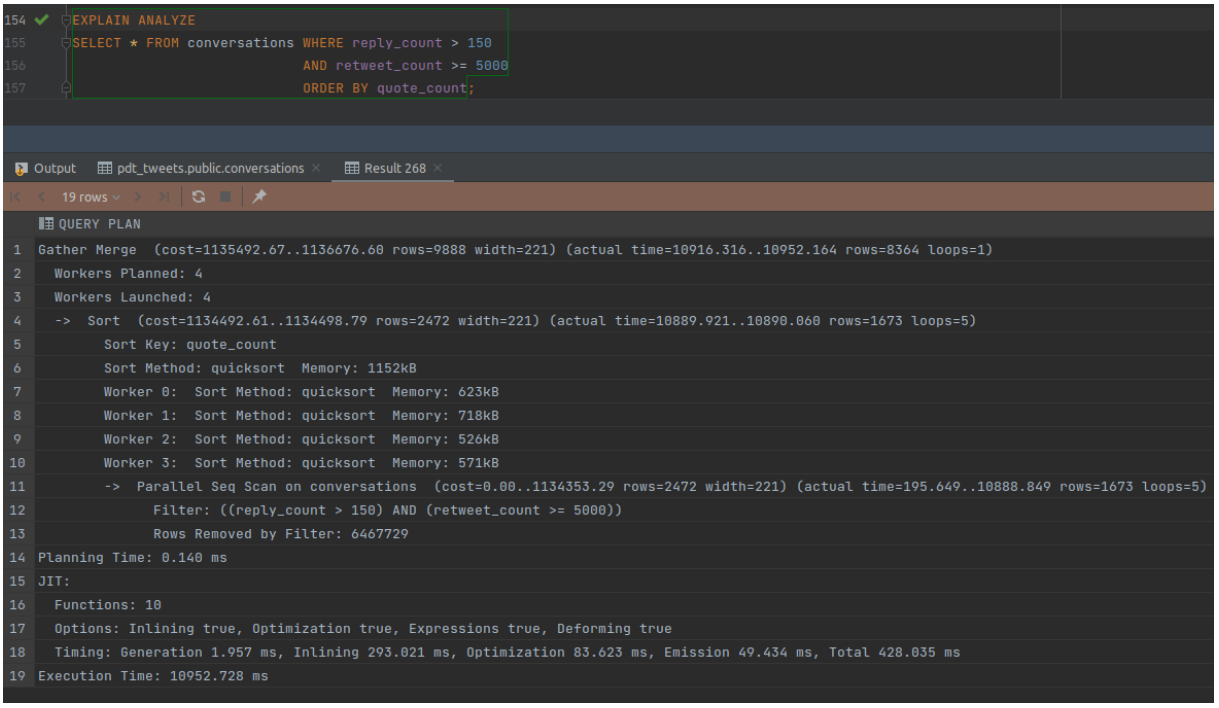
Output | -- S10 ----------...---------- | -- S10 ----------...---------- 2 | pdt_tweets.public.conversations

1 row | Tx: Auto | DDL

	id	author_id	possibly_sensitive	content	language	
1	1502105081151311873	1501402131030364163	false	Vladimir Putin Russia …	en	Twi

12. Jednoduché indexy nad countami

Bez indexu:



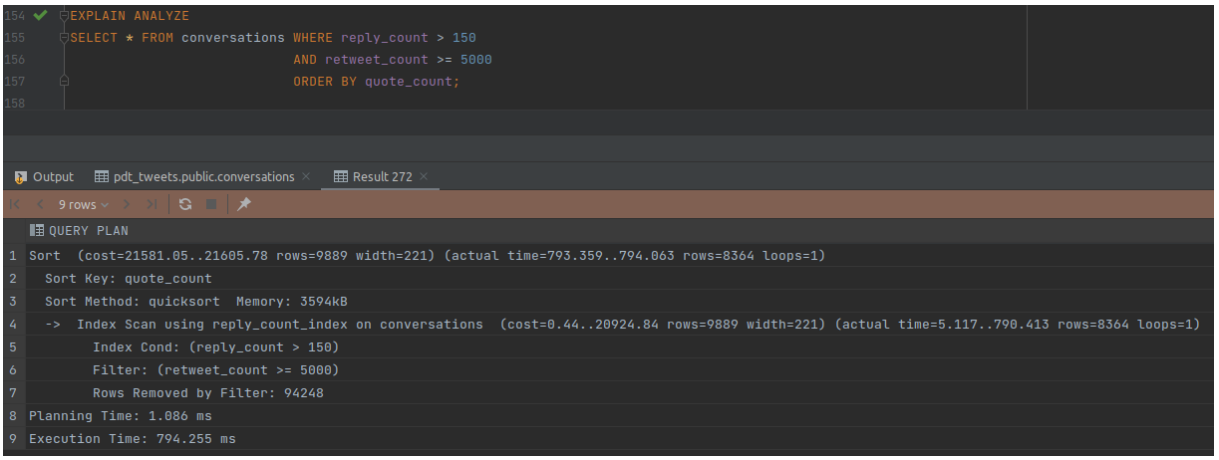
```
EXPLAIN ANALYZE
SELECT * FROM conversations WHERE reply_count > 150
                             AND retweet_count >= 5000
                             ORDER BY quote_count;
```

Output | pdt_tweets.public.conversations | Result 268

19 rows

```
QUERY PLAN
Gather Merge  (cost=1135492.67..1136676.60 rows=9888 width=221) (actual time=10916.316..10952.164 rows=8364 loops=1)
  Workers Planned: 4
  Workers Launched: 4
  ->  Sort  (cost=1134492.61..1134498.79 rows=2472 width=221) (actual time=10889.921..10890.060 rows=1673 loops=5)
        Sort Key: quote_count
        Sort Method: quicksort  Memory: 1152kB
        Worker 0:  Sort Method: quicksort  Memory: 623kB
        Worker 1:  Sort Method: quicksort  Memory: 718kB
        Worker 2:  Sort Method: quicksort  Memory: 526kB
        Worker 3:  Sort Method: quicksort  Memory: 571kB
        ->  Parallel Seq Scan on conversations  (cost=0.00..1134353.29 rows=2472 width=221) (actual time=195.649..10888.849 rows=1673 loops=5)
              Filter: ((reply_count > 150) AND (retweet_count >= 5000))
              Rows Removed by Filter: 6467729
Planning Time: 0.140 ms
JIT:
  Functions: 10
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 1.957 ms, Inlining 293.021 ms, Optimization 83.623 ms, Emission 49.434 ms, Total 428.035 ms
Execution Time: 10952.728 ms
```

S indexami:



```
EXPLAIN ANALYZE
SELECT * FROM conversations WHERE reply_count > 150
                             AND retweet_count >= 5000
                             ORDER BY quote_count;
```

Output | pdt_tweets.public.conversations | Result 272

9 rows

```
QUERY PLAN
Sort  (cost=21581.05..21605.78 rows=9889 width=221) (actual time=793.359..794.063 rows=8364 loops=1)
  Sort Key: quote_count
  Sort Method: quicksort  Memory: 3594kB
  ->  Index Scan using reply_count_index on conversations  (cost=0.44..20924.84 rows=9889 width=221) (actual time=5.117..790.413 rows=8364 loops=1)
        Index Cond: (reply_count > 150)
        Filter: (retweet_count >= 5000)
        Rows Removed by Filter: 94248
Planning Time: 1.086 ms
Execution Time: 794.255 ms
```

Pri spustení s 3 jednoduchými indexami sa využije iba reply_count_index, pretože je efektívnejší, ako retweet_count, keďže reply_count je viac limitujúci - teda ostane mi menej dát. Quicksort použitý cez quote_count je tiež rýchlejší, keďže využije vlastnosti Btree - vľavo menšia, vpravo väčšia hodnota.

Najskôr sa vyfiltruje retweet_count (bez indexu) cez filter - tento filter kontroluje každý riadok, ktorý je vybraný ako vhodný na základe indexu nad reply_countom. Nakoniec sú výsledky usporiadané a to pomocou quicksortu nad quote_countom. Vďaka využitiu indexov je táto query stále násobne rýchlejšia, ako bezindexová.

13. Zložený index nad countami

```
154  CREATE INDEX quote_retweet_reply_index ON conversations
155  USING btree
156  (reply_count, retweet_count, quote_count);
157
158  EXPLAIN ANALYZE
159  SELECT * FROM conversations WHERE retweet_count >= 5000
160                             AND reply_count >= 150
161                             ORDER BY quote_count;
162
```

Output | pdt_tweets.public.conversations | Result 285

7 rows

```
QUERY PLAN
1 Sort  (cost=7809.03..7834.17 rows=10056 width=221) (actual time=13.756..14.722 rows=8365 loops=1)
2   Sort Key: quote_count
3   Sort Method: quicksort  Memory: 3594kB
4   ->  Index Scan using quote_retweet_reply_index on conversations  (cost=0.56..7140.52 rows=10056 width=221) (actual time=0.029..11.411 rows=8365 loops=1)
5         Index Cond: ((reply_count >= 150) AND (retweet_count >= 5000))
6 Planning Time: 0.270 ms
7 Execution Time: 14.948 ms
```

Ako sa dalo predpokladať, zložený index beží rýchlejšie, ako tri separátne indexy. Select dokáže prehľadávať nad tabuľkou všetkých troch indexov, vďaka čomu je rýchlejší. Rozdiel v pláne je, že nie je potrebné opakovane aplikovať filter, ale aplikuje sa iba raz - v podobe filtrovania skrz index. To zabezpečí veľkú rýchlosť, keďže nie je potrebné vyhľadávať viac krát, ale stačí iba raz - a aj to v Btree.

Tento query plan vyberá vhodné záznamy rovno pomocou index BTree nad reply_countom a retweet_countom a teda nemá potrebu filtrovať ešte riadky dodatočne - netreba prechádzať každý riadok viacnásobne. Sortovanie potom rovnako prebehne cez quick_sort quote_countu.

Výsledky boli pre úlohu 12 a 13 rovnaké, akurát sme sa k ním dostali iným spôsobom a sú teda v inom poradí.

```
151  SELECT * FROM conversations WHERE retweet_count >= 5000
152                             AND reply_count >= 150
153                             ORDER BY quote_count;
```

Output | -- S10 ----------...---------- | -- S10 ----------...---------- 2 | pdt_tweets.public.conversations

1-500 of 501+ | Tx: Auto | DDL

	id	author_id	content	possib
1	1497786759710752768	1428688987259080704	$50 | 700K • 5 Hours ⏎⏎RT & Follow Me (🔔)	false
2	1498156335581573120	1428688987259080704	$50 | 700K • 4 Hours ⏎⏎RT & Follow Me (🔔)	false
3	444848423804866560	258809606	What an asshole. #Putin knows #angelamerkel is afraid of dogs yet brings…	false
4	1502965141998272513	1347140802577788928	$100 • 1,4 JT | 12 HOURS⏎⏎RT & Follow @DukeShiller (🔔)	false
5	1498836083416584195	1389658584196149251	$50 | 700.000 IDR • End 8 hour ⏎⏎RT Like & Follow me🔔	false
6	1504290852151390212	1347140802577788928	$100 • 1,4 JT | 12 HOURS⏎⏎RT & Follow me (🔔)	false
7	1497825913178066948	1347140802577788928	$50 • 700K | 4 HOURS⏎⏎RT + Follow @Missypromotes	false
8	1497807026092531721	1363765131172220933	$75 || 1,05 JT in 5 HOURS ⏎⏎RT & Follow @brxwlol	false
9	1504084445384507394	1200355745600069635	$150 in 24 hours 🌸💐⏎⏎-RT + Follow ⏎⏎LRI@CryptoRussDPDI + @cryptogod1101	false
10	1503936472994381825	1200355745600069635	$150 in 24 hours 🥑🌮🍿⏎⏎-RT + Follow ⏎⏎@Imaginary_Apes⏎⏎@metarpgnft⏎⏎@Tom…	false
11	1498174280890449921	1461324154381103112	$200 | 2.800.000 IDR in 24 Hours 🦄⏎⏎- RT & Follow @colinjordan + Li…	false
12	1498607064436068353	1449414811436023812	giveaway 100$ | 1,400,000 IDR⏎⏎-- rt follow (🔔) @flozinbtc ⏎⏎ends in 8 …	false
13	1502275400336314368	1449414811436023812	giveaway 100$ | 1,400,000 IDR⏎⏎--rt follow (🔔) @nemmogiveaway ⏎⏎ends in…	false
14	1497827472431149056	1347140802577788928	$80 • 1.12 JT | 12 HOURS⏎⏎RT & Follow @aiyahgameverse	false
15	1504756243969490950	1461324154381103112	$150 | 2.100.000 IDR in 12 Hours 🦄⏎⏎- RT & Follow @HectorDAO_HEC	false
16	1497815959637151744	1449414811436023812	giveaway 100$ | 1,400,000 IDR⏎⏎-- rt & follow @FynnToTheMoon⏎⏎ends i…	false
17	1497839586822656001	1363765131172220933	$75 || 1,05 JT in 5 HOURS ⏎⏎RT & Follow @GlazeCrypto (🔔)	false
18	420419735072014336	116362700	Winston is the truth	false

Všetky hodnoty sa dajú nájsť v súbore *z02_12_schon.csv*.

14. Index pre Putin a New World Order cez GiST a GIN

GIN:

```
187  CREATE INDEX conversations_gin_content_index ON conversations
188      USING gin(to_tsvector('english', content));
189  EXPLAIN ANALYZE
190  SELECT *
191   FROM conversations
192   WHERE to_tsvector('english', content) @@ to_tsquery('english', 'Putin & New<->World<->Order')
193  AND possibly_sensitive;
```

Output | a | COUNT(*):bigint | plainto_tsquery(url):tsquery | Result 437

10 rows

```
QUERY PLAN
1  Bitmap Heap Scan on conversations  (cost=27.50..28.86 rows=1 width=221) (actual time=43.485..71.689 rows=7 loops=1)
2    Recheck Cond: (to_tsvector('english'::regconfig, content) @@ '''putin'' & ''new'' <-> ''world'' <-> ''order'''::tsquery)
3    Rows Removed by Index Recheck: 88
4    Filter: possibly_sensitive
5    Rows Removed by Filter: 841
6    Heap Blocks: exact=935
7    ->  Bitmap Index Scan on conversations_gin_content_index  (cost=0.00..27.50 rows=1 width=0) (actual time=41.392..41.392 rows=936 loops=1)
8          Index Cond: (to_tsvector('english'::regconfig, content) @@ '''putin'' & ''new'' <-> ''world'' <-> ''order'''::tsquery)
9  Planning Time: 0.129 ms
10 Execution Time: 71.712 ms
```

GIN výsledky:

```
190  SELECT *
191  FROM conversations
192  WHERE to_tsvector('english', content) @@ to_tsquery('english', 'Putin & New<->World<->Order')
193  AND possibly_sensitive;
```

Output | a | COUNT(*):bigint | plainto_tsquery(url):tsquery | pdt_tweets.public.conversations

7 rows | Tx Auto | DDL | CSV

	id	author_id	content	possibly_sensitive	language	source	retweet_count
1	1497395218659745802	47853235	The experience @davidahearst has had in Russia qualifies him for explaini…	true	en	Twitter for Android	0
2	1497653608028028931	1178005700188004355	#Putin and the New World Order from Damascus to Kiev=#WhiteHouse spokesma…	true	en	Twitter for Android	3
3	1498019842183544838	834033178067234816	RT @veteranstoday: Putin: New World Order Worships Satan=https://t.co/fDi…	true	en	Twitter for Android	8
4	1498219488969437184	736549807901282304	"We fight for the New World Order"=We got it…=#UkraineRussiaWar #πολεμος …	true	en	Twitter for iPhone	1
5	1498483618804649998	435312888	#Putin Has Banned #Rothschild And His New World Order Banking Cartel Fami…	true	en	Twitter for Android	0
6	1501745570045743104	3247724967	#Putin is gonna Destroy The New World Order! 🇷🇺 https://t.co/Vvha9YIwX4	true	en	Twitter for iPhone	0
7	1498341252927946758	1506482726	Would Putin take responsibility for this? 6 year old Ukrainian girl kille…	true	en	Twitter Web App	138

GIST:

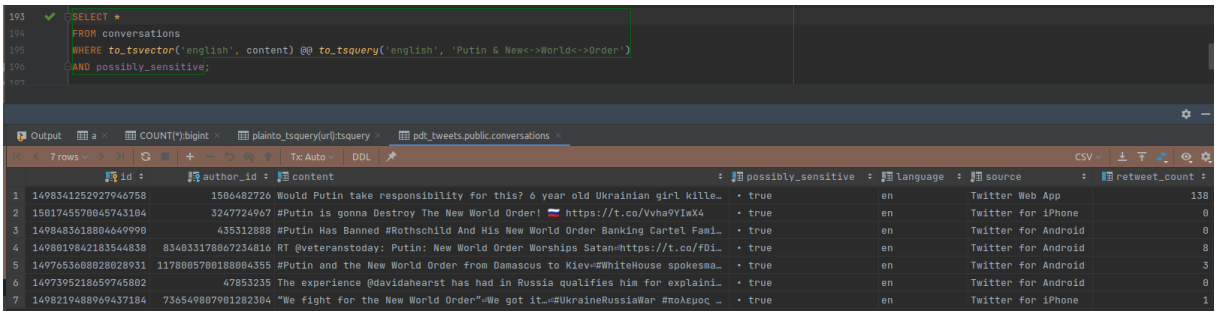
```
192  EXPLAIN ANALYZE
193  SELECT *
194   FROM conversations
195   WHERE to_tsvector('english', content) @@ to_tsquery('english', 'Putin & New<->World<->Order')
196  AND possibly_sensitive;
```

Output | a | COUNT(*):bigint | plainto_tsquery(url):tsquery | Result 440

7 rows

```
QUERY PLAN
1  Index Scan using conversations_gist_content_index on conversations  (cost=0.42..2.64 rows=1 width=221) (actual time=99.852..1300.460 rows=7 loops=1)
2    Index Cond: (to_tsvector('english'::regconfig, content) @@ '''putin'' & ''new'' <-> ''world'' <-> ''order'''::tsquery)
3    Rows Removed by Index Recheck: 88
4    Filter: possibly_sensitive
5    Rows Removed by Filter: 841
6  Planning Time: 9.063 ms
7  Execution Time: 1300.485 ms
```

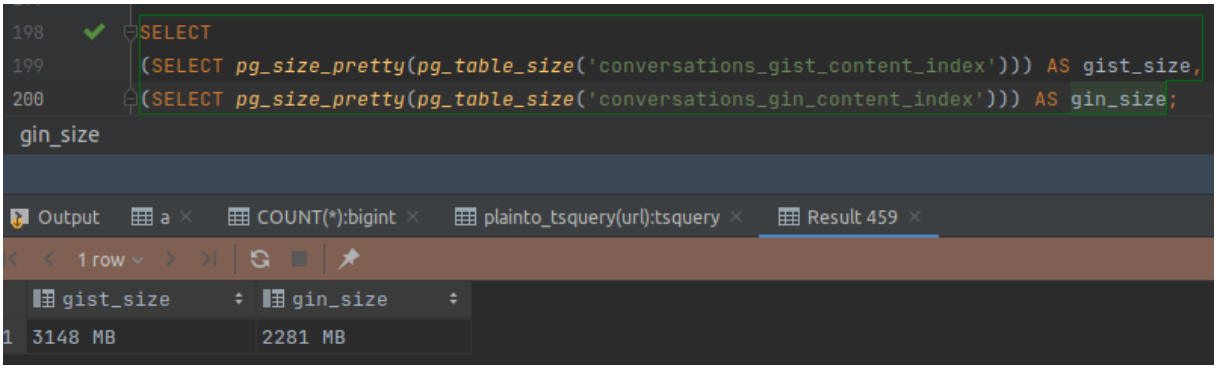
GIST výsledky:



```
SELECT *
FROM conversations
WHERE to_tsvector('english', content) @@ to_tsquery('english', 'Putin & New<->World<->Order')
AND possibly_sensitive;
```

	id	author_id	content	possibly_sensitive	language	source	retweet_count
1	1498341252927946758	1506482726	Would Putin take responsibility for this? 6 year old Ukrainian girl kille…	true	en	Twitter Web App	138
2	1501745570045743104	3247724967	#Putin is gonna Destroy The New World Order! https://t.co/Vvha9YIwX4	true	en	Twitter for iPhone	0
3	1498483618804649990	435312888	#Putin Has Banned #Rothschild And His New World Order Banking Cartel Fami…	true	en	Twitter for Android	0
4	1498019842183544838	834033178867234816	RT @veteranstoday: Putin: New World Order Worships Satan=https://t.co/fDi…	true	en	Twitter for Android	8
5	1497653608028028931	1178005700188004355	#Putin and the New World Order from Damascus to Kiev=#WhiteHouse spokesma…	true	en	Twitter for Android	3
6	1497395218659745802	47853235	The experience @davidahearst has had in Russia qualifies him for explaini…	true	en	Twitter for Android	0
7	1498219488969437184	736549807901282304	“We fight for the New World Order”=We got it…=#UkraineRussiaWar #πολεμος …	true	en	Twitter for iPhone	1

Veľkosť indexov:



```
SELECT
(SELECT pg_size_pretty(pg_table_size('conversations_gist_content_index'))) AS gist_size,
(SELECT pg_size_pretty(pg_table_size('conversations_gin_content_index'))) AS gin_size;
```

gin_size

Output | a | COUNT(*):bigint | plainto_tsquery(url):tsquery | Result 459

1 row

	gist_size	gin_size
1	3148 MB	2281 MB

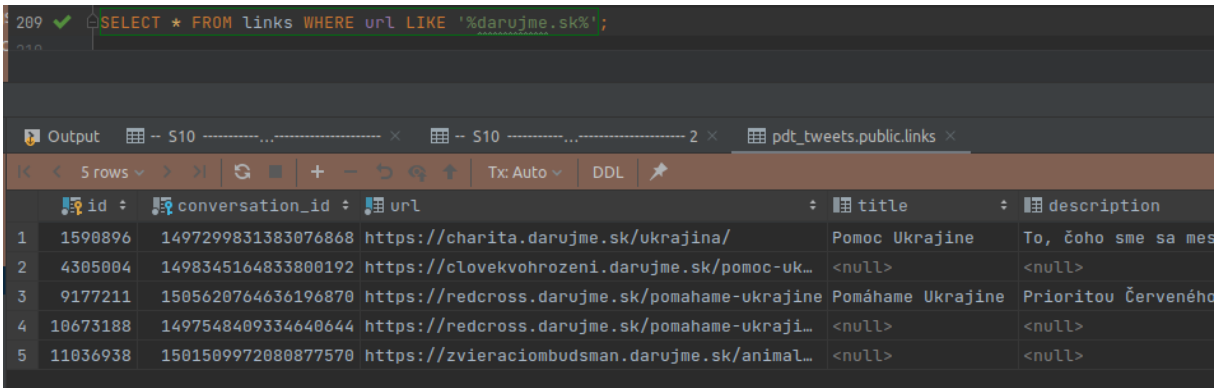
Podarilo sa mi spraviť GIN aj GIST index nad contentom. Následne som pomocou ts_vector a ts_query vytvoril dopyt nad dátami. Výsledky boli rovnaké, dostal som všetky riadky vyhovujúce zadaniu. GIST index je väčší a mal by byť aj efektívnejší. Ak necháme na výber databázy (s tým, že existujú oba indexy) vyberie si GIST. Ten je síce pomalší ako GIN, ale za to lepšie pracuje s pamäťou. V prípade, že by som mal HDD disk tak si myslím, že by GIST bol jednoznačne lepší (vďaka práci s pamäťou a bottle neckom disku). Aktuálne ale ostáva GIN tým, kto zbehne rýchlejšie.

GIN spustí sken, vyberie možné hodnoty a tie následne prefiltruje cez senzitivitu a recheck_condition na overenie si údajov vo vybratých blokoch - pôvodne vybral 936 hodnôt, ktoré recheck condition a senzitivita zredukovali na finálnych 7. Na rozdiel od neho GIST rovno vyberá správne hodnoty. Vyhľadá všetky riadky, ktoré vyhovujú jeho požiadavke na obsah daných slov a potom ich dodatočne prejde filtrom senzitivity.

15. Linky darujme.sk

V tejto úlohe som použil trigramy v kombinácii s GIN indexom. Keďže hľadám darujme.sk v linkoch (a linky sú jeden dlhý string), potrebujem použiť LIKE. Na zaindexovanie dopytov používajúcich LIKE je riešením použiť práve spomínané trigramy. Tie umožňujú vyhľadávanie v texte pomocou indexov. Vďaka tomu môžem vyhľadávať v stringu a zároveň použiť indexy na zrýchlenie takéhoto dopytu - trigramy mi pomôžu rozbiť si text a hľadať v ňom.

Výsledok:



16. Query pomocou FTS

Ako prvé som si vytvoril GIN indexy nad stĺpcami, ktoré budem používať. Vybral som si GIN, nakoľko som spätne skúšal aj GIST a ako rýchlejší mi vyšiel GIN (pri GISTE okolo 3 sekúnd, pri GINe okolo 500ms).

V postgrese NIE JE možné vytvoriť index nad viacerými tabuľkami súčasne. Bolo by možné vytvoriť jeden index nad stĺpcami tabuľky autorov, ale, podobne ako GIST, ani to nezvýšilo efektívnosť.

```
CREATE INDEX authors_description_gin_index ON authors
    USING gin(to_tsvector('english', description));
CREATE INDEX authors_username_gin_index ON authors
    USING gin(to_tsvector('english', username));
CREATE INDEX conversations_content_index ON conversations
    USING gin(to_tsvector('english', content));
```

Poznámka: Po vypracovaní tejto časti zadania som sa na Slacku dočítal, že je možné pristupovať k tejto úlohe tak, že hľadáme stĺpce, v ktorých sa nachádzajú OBE slová (AND medzi nimi). Ja som úlohu pochopil tak, že stačí, ak sa v texte nachádza jedno zo slov (teda OR medzi nimi). V prvej časti teda vysvetľujem svoj postup a myšlenky pri riešení s ORom a v závere popisujem aj ANDové riešenie.

Následne som vyskúšal niekoľko prístupov k získaniu sediacich dát. Nasledujúca query vráti 7923 riadkov dát, ktoré verím, že sú správne:

```
SELECT content.content, username.username, description.description, content.retweet_count
 FROM

(SELECT id, author_id, content, retweet_count
 FROM conversations
 WHERE (to_tsvector('english', content)
          @@ to_tsquery('english', 'Володимир | Президент'))) AS content

FULL OUTER JOIN (SELECT id, username
                 FROM authors
                 WHERE (to_tsvector('english', username))
                          @@ to_tsquery('english', 'Володимир | Президент'))
                 AS username
    ON username.id = content.author_id

FULL OUTER JOIN (SELECT id, description
                FROM authors
                WHERE to_tsvector('english', description)
                         @@ to_tsquery('english', 'Володимир | Президент'))
                AS description
    ON description.id = content.author_id

ORDER BY retweet_count DESC NULLS LAST;
```

V tejto query vykonám výber vhodných hodnôt z každého stĺpca samostatne - pomocou tsvectorov a tsqueries operácií nad jednotlivými tabuľkami. Samostatné vyberanie vo vnorenej query prebehne za pomoci indexu a následne môžem takto vybrané hodnoty spojiť pomocou JOINov a ANDov medzi nimi. Na koniec zoradím hodnoty podľa retweet countov. Výsledné dáta prikladám aj v csv súbore - *z02_16_schon.csv*.

Prvých pár výsledkov query:

	content	username	description	retweet_count
1	⚡ Володимир Кличко звернувся до демократичного світу із закликом нега…	<null>	<null>	7351
2	Президент України Володимир Зеленський відвідав у госпіталі поранених…	<null>	<null>	5848
3	💬Президент України Володимир Зеленський:⏎"Всі ми тут - захищаємо наш…	<null>	<null>	5346
4	В бою загинув льотчик-винищувач Олександр Оксанченко. ⏎Він був одним …	<null>	<null>	2799
5	⚡ Володимир Кличко звернувся до громадян Німеччини, Австрії, Швейцарі…	<null>	<null>	2628
6	На митинг в Санкт-Петербурге вышла 84-летняя Ирина Анохина, которая п…	<null>	<null>	2569
7	Володимир Ковальський.⏎У 2016-му році під час виконання бойового завд…	<null>	<null>	2408
8	Президент України @ZelenskyyUa поспілкувався зі звільненим із полону …	<null>	<null>	2200
9	Итого на сегодня в России:⏎- уничтожены почти все независимые СМИ⏎- в…	<null>	<null>	2118
10	“Президент Зеленський оголосив про формування нового підрозділу – «Ін…	<null>	<null>	2054
11	This is a rhetorical question. I know Putin is in his Ural bunker. Th…	<null>	<null>	1676
12	🕪 «Ми вдома і захищаємо Україну. Ніякої зброї ми складати не збираєм…	<null>	<null>	1580
13	Лукашенко, після розмови із Зеленським, жаліється, що "президент Укра…	<null>	<null>	1539
14	Президент Швейцарии заявил, что Швейцария больше не нейтральная стран…	<null>	<null>	1530
15	Головнокомандувачу ЗС України генерал-лейтенанту Валерію Залужному пр…	<null>	<null>	1489
16	Повертаючись до України зі США, у Варшаві мене прийняв Президент Поль…	<null>	<null>	1446
17	Сергій Васіч обороняв позиції на Київщині. ⏎Він знищив ворожу техні…	<null>	<null>	1400
18	Президент Молдовы потребовала от РФ немедленно вывести войска из терр…	<null>	<null>	1341
19	Президент України @ZelenskyyUa :⏎Торік я попереджав лідерів @NATO: як…	<null>	<null>	1306
20	"Я поранений, залишилося недовго!" Заступник мера Нової Каховки встиг…	<null>	<null>	1253
21	Во время Второй мировой войны четыре брата из одной еврейской семьи з…	<null>	<null>	1246
22	!!! Майже 9000 вбитих 🇷🇺 окупантів за один тиждень війни!⏎“Кожен ок…	<null>	<null>	1217
23	Пора уже всем россиянам с мозгами перестать быть «вне политики». ⏎⏎Не…	<null>	<null>	1212
24	Володимир Андрійченко поїхав забирати поранених та вбитих побратимів …	<null>	<null>	1134
25	!!!! Президент України Володимир Зеленський підписав заявку на чле…	<null>	<null>	1109
26	Кстати, а теперь ведь не кажется конспирологией, что президент Качинс…	<null>	<null>	1092

Pomocou EXPLAIN ANALYZE som si overil aj použitie indexov / skenov. Ako je možné vidieť, použili sa indexy, joiny prebehli pomocou hash full joinu. Nebolo potrebné vykonať žiadny sekvenčný sken nad dátami z dôvodov uvedených vyššie. Nad každým subselectom prebehol výber pomocou GIN indexu patriacemu danému stĺpcu medzi dvoma slovami, kde gather spojil možné výsledky. Všetky výsledky boli následne joinuté spolu a vytriedené quicksortom nad retweet_countom.

```
QUERY PLAN
Sort  (cost=1422838.36..1423645.01 rows=322661 width=248) (actual time=517.331..518.812 rows=7923 loops=1)
  Sort Key: conversations.retweet_count DESC
  Sort Method: quicksort  Memory: 3810kB
  ->  Hash Full Join  (cost=125555.95..1371351.72 rows=322661 width=248) (actual time=281.309..514.271 rows=7923 loops=1)
        Hash Cond: (conversations.author_id = authors_1.id)
        ->  Hash Full Join  (cost=64198.34..1291957.13 rows=322661 width=182) (actual time=52.532..273.598 rows=7777 loops=1)
              Hash Cond: (conversations.author_id = authors.id)
              ->  Gather  (cost=3644.73..1230556.53 rows=322661 width=171) (actual time=28.244..230.878 rows=7777 loops=1)
                    Workers Planned: 8
                    Workers Launched: 8
                    ->  Parallel Bitmap Heap Scan on conversations  (cost=2644.73..1197290.43 rows=40333 width=171) (actual time=165.640..169.031 rows=864 loops=9)
                          Recheck Cond: (to_tsvector('english'::regconfig, content) @@ '''володимир'' | ''президент'''::tsquery)
                          Heap Blocks: exact=7639
                          ->  Bitmap Index Scan on conversations_gin_content_index  (cost=0.00..2564.06 rows=322661 width=0) (actual time=26.161..26.161 rows=7777 loops=1)
                                Index Cond: (to_tsvector('english'::regconfig, content) @@ '''володимир'' | ''президент'''::tsquery)
              ->  Hash  (cost=59818.56..59818.56 rows=58804 width=19) (actual time=24.256..24.362 rows=0 loops=1)
                    Buckets: 65536  Batches: 1  Memory Usage: 512kB
                    ->  Gather  (cost=1490.93..59818.56 rows=58804 width=19) (actual time=24.255..24.360 rows=0 loops=1)
                          Workers Planned: 4
                          Workers Launched: 4
                          ->  Parallel Bitmap Heap Scan on authors  (cost=490.93..52938.16 rows=14701 width=19) (actual time=0.058..0.058 rows=0 loops=5)
                                Recheck Cond: (to_tsvector('english'::regconfig, (username)::text) @@ '''володимир'' | ''президент'''::tsquery)
                                ->  Bitmap Index Scan on authors_username_gin_index  (cost=0.00..476.23 rows=58804 width=0) (actual time=0.013..0.014 rows=0 loops=1)
                                      Index Cond: (to_tsvector('english'::regconfig, (username)::text) @@ '''володимир'' | ''президент'''::tsquery)
        ->  Hash  (cost=59818.56..59818.56 rows=58804 width=82) (actual time=228.633..228.746 rows=155 loops=1)
              Buckets: 65536  Batches: 2  Memory Usage: 529kB
              ->  Gather  (cost=1490.93..59818.56 rows=58804 width=82) (actual time=228.457..228.591 rows=155 loops=1)
                    Workers Planned: 4
                    Workers Launched: 4
                    ->  Parallel Bitmap Heap Scan on authors authors_1  (cost=490.93..52938.16 rows=14701 width=82) (actual time=177.939..178.089 rows=31 loops=5)
                          Recheck Cond: (to_tsvector('english'::regconfig, description) @@ '''володимир'' | ''президент'''::tsquery)
                          Heap Blocks: exact=1
                          ->  Bitmap Index Scan on authors_description_gin_index  (cost=0.00..476.23 rows=58804 width=0) (actual time=0.270..0.270 rows=155 loops=1)
                                Index Cond: (to_tsvector('english'::regconfig, description) @@ '''володимир'' | ''президент'''::tsquery)
Planning Time: 0.259 ms
```

```
Planning Time: 0.259 ms
JIT:
  Functions: 88
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 8.510 ms, Inlining 1384.396 ms, Optimization 619.058 ms, Emission 305.446 ms, Total 2317.410 ms
Execution Time: 520.310 ms
```

Na zopakovanie dôvodov uvedených v texte: OR podmienka nie je možná, pretože sa nepoužijú indexy a joiny sú možné, pretože výber dát prebehol pred joinovaním.

V prípade, že chcem vybrať tie záznamy, ktoré v jednom zo stĺpcov obsahujú obe slová, stačí mojej query jednoduchá úprava, kedy vymením | (or) za & (and):

```
EXPLAIN ANALYZE
SELECT content.content, username.username, description.description, content.retweet_count
 FROM

(SELECT id, author_id, content, retweet_count
 FROM conversations
 WHERE (to_tsvector('english', content)
           @@ to_tsquery('english', 'Володимир & Президент'))) AS content

FULL OUTER JOIN (SELECT id, username
                 FROM authors
                 WHERE (to_tsvector('english', username))
                           @@ to_tsquery('english', 'Володимир & Президент'))
                 AS username
    ON username.id = content.author_id

FULL OUTER JOIN (SELECT id, description
                FROM authors
                WHERE to_tsvector('english', description)
                          @@ to_tsquery('english', 'Володимир & Президент'))
                AS description
    ON description.id = content.author_id

ORDER BY retweet_count DESC NULLS LAST;
```

Takýto dopyt potom vráti 1043 záznamov, tuna sú s najpočetnejším retweet_countom:

	content	username	description	retweet_count
1	Президент України Володимир Зеленський відвідав у госпіталі поранених…	<null>	<null>	5848
2	💬Президент України Володимир Зеленський:⏎"Всі ми тут - захищаємо наш…	<null>	<null>	5346
3	В бою загинув льотчик-винищувач Олександр Оксанченко. ⏎Він був одним …	<null>	<null>	2799
4	Головнокомандувачу ЗС України генерал-лейтенанту Валерію Залужному пр…	<null>	<null>	1489
5	!!! Майже 9000 вбитих 🇷🇺 окупантів за один тиждень війни!⏎"Кожен ок…	<null>	<null>	1217
6	!!!! Президент України Володимир Зеленський підписав заявку на чле…	<null>	<null>	1109
7	‼️ Сьогодні, 1 березня, Президент України Володимир Зеленський висту…	<null>	<null>	649
8	Президент 🇺🇦 Володимир Зеленський підписав указ № 74/2022 про присв…	<null>	<null>	609
9	Президент Володимир Зеленський підписав указ про присвоєння звання Ге…	<null>	<null>	472
10	Президент України Володимир Зеленський звернувся до всіх громадян іно…	<null>	<null>	433
11	Президент у відповідь на пропозицію Байдена щодо евакуації:⏎– Мені по…	<null>	<null>	423
12	Україна звертається до Європейського Союзу щодо невідкладного приєдна…	<null>	<null>	347
13	Україна сьогодні воює за цінності свободи й демократії, і інші країни…	<null>	<null>	341
14	Президент Володимир Зеленський відвідав поранених захисників України,…	<null>	<null>	337
15	Президент України Володимир Зеленський наголошує на тому, що окупуват…	<null>	<null>	303
16	Президент України Володимир Зеленський звернувся до громадян Білорусі…	<null>	<null>	294
17	Президент Володимир Зеленський вручив державні нагороди працівникам С…	<null>	<null>	287
18	Сприяння наданню Україні членства у Європейському Союзі є одним із ва…	<null>	<null>	274
19	💬Президент України Володимир Зеленський:⏎⏎Кожна російська ракета, як…	<null>	<null>	258
20	Президент Володимир Зеленський відвідав наших поранених захисників ht…	<null>	<null>	246
21	Президент Володимир Зеленський підписав заявку на членство України у …	<null>	<null>	219
22	Буча, Ірпінь, Охтирка, Миколаїв стали містами-героями. Відповідний Ук…	<null>	<null>	208
23	RT @DefenceU: Президент України Володимир Зеленський відзначив 7️⃣1…	<null>	<null>	205
24	RT @DefenceU: Президент України Володимир Зеленський відзначив 7️⃣1…	<null>	<null>	205
25	RT @DefenceU: Президент України Володимир Зеленський відзначив 7️⃣1…	<null>	<null>	205
26	RT @DefenceU: Президент України Володимир Зеленський відзначив 7️⃣1…	<null>	<null>	205
27	RT @DefenceU: Президент України Володимир Зеленський відзначив 7️⃣1…	<null>	<null>	205
28	Президент України Володимир Зеленський відзначив 7️⃣1 українського …	<null>	<null>	205
29	RT @DefenceU: Президент України Володимир Зеленський відзначив 7️⃣1…	<null>	<null>	205
30	RT @DefenceU: Президент України Володимир Зеленський відзначив 7️⃣1…	<null>	<null>	205
31	RT @DefenceU: Президент України Володимир Зеленський відзначив 7️⃣1…	<null>	<null>	205
32	RT @DefenceU: Президент України Володимир Зеленський відзначив 7️⃣1…	<null>	<null>	205
33	RT @DefenceU: Президент України Володимир Зеленський відзначив 7️⃣1…	<null>	<null>	205

Na základe takéhoto prístupu získam aj iný QUERY plán a to nasledovný.

Output | Query to get results

26 rows

```
QUERY PLAN
Sort  (cost=1593.55..1595.57 rows=809 width=248) (actual time=3.658..3.769 rows=1043 loops=1)
  Sort Key: conversations.retweet_count DESC NULLS LAST
  Sort Method: quicksort  Memory: 553kB
  ->  Hash Full Join  (cost=450.25..1554.48 rows=809 width=248) (actual time=1.112..3.227 rows=1043 loops=1)
        Hash Cond: (conversations.author_id = authors_1.id)
        ->  Hash Full Join  (cost=235.41..1337.51 rows=809 width=182) (actual time=1.079..2.942 rows=1043 loops=1)
              Hash Cond: (conversations.author_id = authors.id)
              ->  Bitmap Heap Scan on conversations  (cost=20.57..1120.54 rows=809 width=171) (actual time=1.059..2.452 rows=1043 loops=1)
                    Recheck Cond: (to_tsvector('english'::regconfig, content) @@ '''володимир'' & ''президент'''::tsquery)
                    Heap Blocks: exact=1017
                    ->  Bitmap Index Scan on conversations_gin_content_index  (cost=0.00..20.37 rows=809 width=0) (actual time=0.821..0.822 rows=1043 loops=1)
                          Index Cond: (to_tsvector('english'::regconfig, content) @@ '''володимир'' & ''президент'''::tsquery)
              ->  Hash  (cost=213.01..213.01 rows=147 width=19) (actual time=0.011..0.012 rows=0 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 8kB
                    ->  Bitmap Heap Scan on authors  (cost=13.24..213.01 rows=147 width=19) (actual time=0.011..0.011 rows=0 loops=1)
                          Recheck Cond: (to_tsvector('english'::regconfig, (username)::text) @@ '''володимир'' & ''президент'''::tsquery)
                          ->  Bitmap Index Scan on authors_username_gin_index  (cost=0.00..13.21 rows=147 width=0) (actual time=0.010..0.010 rows=0 loops=1)
                                Index Cond: (to_tsvector('english'::regconfig, (username)::text) @@ '''володимир'' & ''президент'''::tsquery)
        ->  Hash  (cost=213.01..213.01 rows=147 width=82) (actual time=0.028..0.029 rows=0 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 8kB
              ->  Bitmap Heap Scan on authors authors_1  (cost=13.24..213.01 rows=147 width=82) (actual time=0.028..0.028 rows=0 loops=1)
                    Recheck Cond: (to_tsvector('english'::regconfig, description) @@ '''володимир'' & ''президент'''::tsquery)
                    ->  Bitmap Index Scan on authors_description_gin_index  (cost=0.00..13.21 rows=147 width=0) (actual time=0.025..0.026 rows=0 loops=1)
                          Index Cond: (to_tsvector('english'::regconfig, description) @@ '''володимир'' & ''президент'''::tsquery)
Planning Time: 0.380 ms
Execution Time: 3.869 ms
```

Takto napísaná query zbehne veľmi, veľmi rýchlo a využíva indexy - žiadne sekvenčné skeny. Tento query planner sa líši tým, že nepoužíva gatherery po výbere, nakoľko cez GIN index a AND podmienku priamo vyberie vhodné riadky, ktoré následne recheckne a takto pripravené dáta joinne na ostatné dáta zo stĺpcov. Nakoniec vytriedi dáta cez quicksort nad retweet_countom.

Záver

Celé pozorovanie bolo uskutočnené v PostgreSQL verzie 14. SQL súbor s dopytmi, rovnako ako aj spomínané dátové súbory sa nachádzajú priložené pri tomto dokumente, prípadne sú zverejnené na githube spomenutom na prvej strane tohto dokumentu. Zadanie som vypracoval samostatne, myšlienky sú moje vlastné. Verím, že sa vám dobre čítalo a všetko bolo zrozumiteľné.