## Fakulta informatiky a informačných technológií Slovenská technická univerzita

Pokročilé databázové technológie

Vyhľadávanie a indexovanie

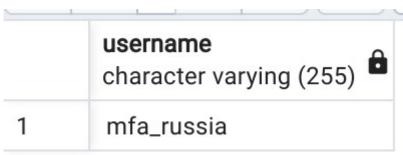
https://github.com/Matovic/postgres-indexes

Zadanie 2

2022 Erik Matovič

# 1. Vyhľadajte v authors username s presnou hodnotou 'mfa\_russia' a analyzujte daný select. Akú metódu vám vybral plánovač a prečo - odôvodnite prečo sa rozhodol tak ako sa rozhodol?

Z obrázku č. 02 vidíme výsledok z explain analyse prvej úlohy. Plánovač vybral paralelný sekvenčný sken s dvomi naplánovanými a spustenými workermi, a teda bola použitá paralelizácia dopytu.



Obr. č. 01: Výsledok úlohy 1

	QUERY PLAN text	â
1	Gather (cost=1000.00147229.44 rows=1 width=12) (actual time=401.853418.202 rows=1 loops=1)	
2	Workers Planned: 2	
3	Workers Launched: 2	
4	-> Parallel Seq Scan on authors (cost=0.00146229.34 rows=1 width=12) (actual time=267.965387.284 rows=0 loops=3)	
5	Filter: ((username)::text = 'mfa_russia'::text)	
6	Rows Removed by Filter: 1965058	
7	Planning Time: 0.071 ms	
8	JIT:	
9	Functions: 12	
10	Options: Inlining false, Optimization false, Expressions true, Deforming true	
11	Timing: Generation 1.205 ms, Inlining 0.000 ms, Optimization 0.786 ms, Emission 11.842 ms, Total 13.833 ms	
12	Execution Time: 418.618 ms	

Obr. č. 02: Výsledok z explain analyse úlohy 1

# 2. Koľko workerov pracovalo na danom selecte a na čo slúžia? Zdvihnite počet workerov a povedzte ako to ovplyvňuje čas. Je tam nejaký strop? Ak áno, prečo? Od čoho to závisí (napíšte a popíšte všetky parametre)?

Z obrázku č. 02 vidíme, že maximálny počet workerov bol 2 a slúžia na paralelizáciu dopytu. Paralelizácia závisí od parametra max\_parallel\_workers\_per\_gather, ktorý si vieme zobraziť a pred úpravou bol maximálny počet nastavený na 2, viď obr. č. 03, preto aj plánovač pri prvom spustení dopytu naplánoval a spustil 2 workerov. Tento parameter je limitovaný ďalšími parametrami ako max\_worker\_processes a max\_parallel\_workers, pričom horný limit celočíselnej hodnoty max\_parallel\_workers\_per\_gather je maximálne rovný hodnote max\_parallel\_workers, ktorý je zas nesmie prekročiť hodnotu max\_worker\_processes, pretože nebude mať žiadny efekt.¹ Oba tieto parametre, max\_worker\_processes a max\_parallel\_workers, sú nastavené na predvolené hodnoty, viď obr. č. 04.

	max_parallel_workers_per_gather text	9
1	2	

Obr. č. 03: Parameter max\_parallel\_workers\_per\_gather pred zmenou

	max_worker_processes text		max_parallel_workers text
1	8	1	8

Obr. č. 04: Hodnoty parametrov max\_worker\_processes a max\_parallel\_workers Zo začiatku sme zdvihli hodnotu max\_parallel\_workers\_per\_gather na 4, čo skrátilo čas približne o polovicu, viď obrázok č. 05. Pri zdvihnutí na hodnotu max\_parallel\_workers máme kratší čas, ale nie už tak zásadne, pertože plánovač naplánoval a spustil 5 workerov a nie 8(viď obr. č. 06), čo je bežné v paralelizácií, že nie vždy maximálny počet workerov s využije.

<sup>1</sup> PostgreSQL 14 Documentation. Chapter 15. Parallel Query: 15.1. How Parallel Query Works. https://www.postgresql.org/docs/14/how-parallel-query-works.html

	QUERY PLAN text
1	Gather (cost=1000.00134996.51 rows=1 width=12) (actual time=256.631267.924 rows=1 loops=1)
2	Workers Planned: 4
3	Workers Launched: 4
4	-> Parallel Seq Scan on authors (cost=0.00133996.41 rows=1 width=12) (actual time=193.400236.565 rows=0 loops=5)
5	Filter: ((username)::text = 'mfa_russia'::text)
6	Rows Removed by Filter: 1179035
7	Planning Time: 0.075 ms
8	JIT:
9	Functions: 20
10	Options: Inlining false, Optimization false, Expressions true, Deforming true
11	Timing: Generation 2.058 ms, Inlining 0.000 ms, Optimization 1.273 ms, Emission 19.986 ms, Total 23.317 ms
12	Execution Time: 268.299 ms

Obr. č. 05: Výsledok z explain analyse úlohy pri navýšeni workers na 4

QUERY F text	PLAN
Gather (	(cost=1000.00131326.62 rows=1 width=12) (actual time=41.867226.465 rows=1 loops=1)
Workers	s Planned: 5
Workers	s Launched: 5
-> Parall	lel Seq Scan on authors (cost=0.00130326.52 rows=1 width=12) (actual time=161.630190.316 rows=0 loops=6)
Filter: ((	(username)::text = 'mfa_russia'::text)
Rows Re	emoved by Filter: 982529
Planning	g Time: 0.071 ms
JIT:	
Function	ns: 24
Options	: Inlining false, Optimization false, Expressions true, Deforming true
Timing:	Generation 2.114 ms, Inlining 0.000 ms, Optimization 1.564 ms, Emission 26.170 ms, Total 29.849 ms
Execution	on Time: 226.842 ms

Obr. č. 06: Výsledok z explain analyse úlohy pri navýšeni workers na 8

# 3. Vytvorte btree index nad username a pozrite ako sa zmenil čas a porovnajte výstup oproti požiadavke bez indexu. Potrebuje plánovač v tejto požiadavke viac workerov? Čo ovplyvnilo zásadnú zmenu času?

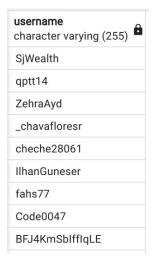
Pri porovnaní obrázku 06 a 07 jasne vidíme, že aj keď najlepšia možná paralelizácia, ktorá nám vyšla(5 workerov na dopyt), je niekoľkonásobne pomalšia ako použitie btree indexu. Je to z toho dôvodu, že btree index funguje vhodne pri porovnávaní dát a nakoľko pri vytváraní indexu je už predvolene nastavená utrieďovacá metóda ASC, tak pri hľadaní rovnosti v stromovej štuktúre je kratší čas ako použitie sekvenčného skenu a to ovplyvnilo zmenu času. Viac workerov nie je potrebných, paralelizácia sa v tomto prípade nepoužije, pretože by bolo potrebné synchronizovať ktorý worker našiel v strome správnu hodnotu a nie je to potrebné nakoľko btree index hľadá podľa porovnania(rovnosť v našom prípade) v utriedenej stromovej štuktúre.

QUERY PLAN text
Index Only Scan using idx_authors_username on authors (cost=0.432.65 rows=1 width=12) (actual time=0.0470.050 rows=1 loops=
Index Cond: (username = 'mfa_russia'::text)
Heap Fetches: 0
Planning Time: 0.135 ms
Execution Time: 0.068 ms

Obr. č. 07: Výsledok z explain analyse úlohy 3

4. Vyberte používateľov, ktorý majú followers\_count väčší, rovný ako 100 a zároveň menší, rovný 200. Potom zmeňte rozsah na väčší, rovný ako 100 a zároveň menší, rovný 120. Je tam rozdiel, ak áno prečo?

Z obrázkov 08 a 09 vidíme rozdiel pri paralelizácií. Pri prvej podmienke nie je použitá paralelizácia, a preto je výsledok pomalší. Je to z dôvodu parametra parallel\_tuple\_cost, ktorý plánovaču odhaduje cenu presunu riadku medzi workermi a je nastavený predvolene na 0.1, pri jeho vynulovaní dosiahneme paralelizáciu, viď obr. č. 10 a dokonca aj o niečo lepší čas.





Obr. č. 07: Vzorok výsledkov úlohy 4. Vľavo pri podmienke followers\_count >= 100 AND followers\_count <= 200 a vpravo pri podmienke followers\_count >= 100 AND followers\_count <= 120. Kompletné výsledky oboch podmienok su priložené v príslušných CSV súboroch.

QUERY PLAN text
Seq Scan on authors (cost=0.00204074.64 rows=761932 width=12) (actual time=3.2181308.570 rows=760088 loops=1)
Filter: ((followers_count >= 100) AND (followers_count <= 200))
Rows Removed by Filter: 5135088
Planning Time: 0.073 ms
JIT:
Functions: 4
Options: Inlining false, Optimization false, Expressions true, Deforming true
Timing: Generation 0.214 ms, Inlining 0.000 ms, Optimization 0.156 ms, Emission 3.045 ms, Total 3.415 ms
Execution Time: 2159.148 ms

Obr. č. 08: Výsledok z explain analyse pri podmienke followers\_count >= 100 AND followers\_count <= 200

#### **QUERY PLAN**

text

Gather (cost=1000.00..154301.43 rows=199689 width=12) (actual time=3.148..256.879 rows=199937 loops=1)

Workers Planned: 5

Workers Launched: 5

-> Parallel Seq Scan on authors (cost=0.00..133332.53 rows=39938 width=12) (actual time=3.863..114.412 rows=33323 loops=6)

Filter: ((followers\_count >= 100) AND (followers\_count <= 120))

Rows Removed by Filter: 949206

Planning Time: 0.067 ms

JIT:

Functions: 24

Options: Inlining false, Optimization false, Expressions true, Deforming true

Timing: Generation 1.473 ms, Inlining 0.000 ms, Optimization 0.979 ms, Emission 22.001 ms, Total 24.452 ms

Execution Time: 495.265 ms

Obr. č. 09: Výsledok z explain analyse pri podmienke followers\_count >= 100 AND followers\_count <= 120

## **QUERY PLAN**

text

Gather (cost=1000.00..134332.53 rows=761932 width=12) (actual time=3.832..938.487 rows=760088 loops=1)

Workers Planned: 5

Workers Launched: 5

-> Parallel Seq Scan on authors (cost=0.00..133332.53 rows=152386 width=12) (actual time=3.486..201.112 rows=126681 loops=6)

Filter: ((followers\_count >= 100) AND (followers\_count <= 200))

Rows Removed by Filter: 855848

Planning Time: 0.094 ms

JIT:

Functions: 24

Options: Inlining false, Optimization false, Expressions true, Deforming true

Timing: Generation 1.620 ms, Inlining 0.000 ms, Optimization 1.025 ms, Emission 19.760 ms, Total 22.405 ms

Execution Time: 1837.923 ms

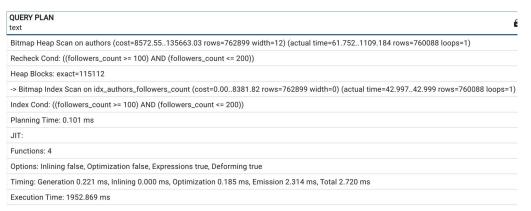
Obr. č. 10: Výsledok z explain analyse pri podmienke followers\_count >= 100 AND followers\_count <= 200 po úprave parametra parallel\_tuple\_cost na 0

# 5. Vytvorte index nad 4 úlohou a v oboch podmienkach popíšte prácu s indexom. Čo je to Bitmap Index Scan a prečo je tam Bitmap Heap Scan? Prečo je tam recheck condition? Použil sa vždy index?

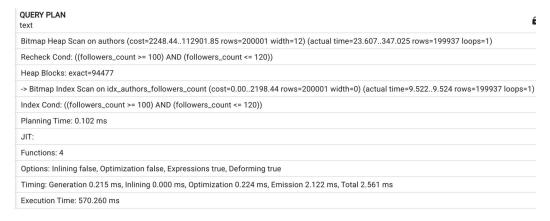
Práca s indexom fungovala rýchlejšie pri prvej podmienke bez paralelizmu, avšak ako je vidieť rozdiel medzi obrázkami 10 a 11, tak rýchlejšie bolo, ak sme aktivovali paralelizmus pri prvej podmienke a podobne aj pri druhej podmienke je rýchlejší paralelizmus, avšak jedná sa približne len o pol sekundy. Index bol použitý pri oboch dopytoch.

Bitmap Index Scan má prvky Index Scan, pretože skenuje index pre určenie aké dáta musia byť získané, avšak podobá sa aj na sekvenčný sken, pretože číta dáta vo väčšom objeme a zároveň si vytvorí bitmapu potencionálnych pozícií riadkov. Bitmap Index Scan je väčšinou potomkom uzla Bitmap Heap Scan, preto sa objavuje v našich výsledkoch.<sup>2</sup>

Bitmap Heap Scan vyhľadá dáta tak, že zoberie polohu riadku, ktorá bola vygenerovaná od Bitmap Index Scan, pričom sa môže stať, že ukazovateľ nie je nastavený priamo na riadok, ale na stránku kvôli pamäti a vtedy je potrebná recheck condition kvôli identifikácií konkrétnych riadkov na stránke.<sup>3</sup>



Obr. č. 11: Výsledok z explain analyse pri prvej podmienke



Obr. č. 12: Výsledok z explain analyse pri druhej podmienke

<sup>2</sup> EXPLAIN - Bitmap Index Scan. Pganalyze: https://pganalyze.com/docs/explain/scan-nodes/bitmap-index-scan

<sup>3</sup> EXPLAIN - Bitmap Heap Scan. Pganalyze: https://pganalyze.com/docs/explain/scan-nodes/bitmap-heap-scan

# 6. Vytvorte d'alšie 3 btree indexy na name, followers\_count, a description a insertnite si svojho používatel'a (to je jedno aké dáta) do authors. Koľko to trvalo? Dropnite indexy a spravte to ešte raz. Prečo je tu rozdiel?

Ako ukazuje obrázok 13 a 14, tak rýchlejšie je insertovanie bez indexov(čas vykonania 0.156 ms oproti 5.466 ms pri indexoch), pretože čím viac indexov má tabuľka, tým pomalšie trvá vykonanie insertu.

QUERY PLAN text

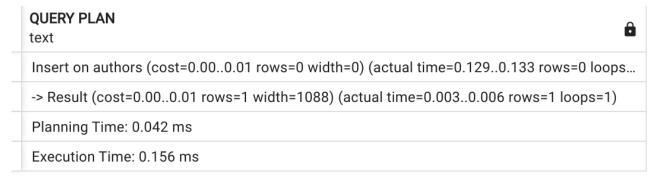
Insert on authors (cost=0.00..0.01 rows=0 width=0) (actual time=5.434..5.438 rows=0 loops=1)

-> Result (cost=0.00..0.01 rows=1 width=1088) (actual time=0.002..0.006 rows=1 loops=1)

Planning Time: 0.046 ms

Execution Time: 5.466 ms

Obr. č. 13: Výsledok z explain analyse pri indexoch



Obr. č. 14: Výsledok z explain analyse bez indexov

# 7. Vytvorte btree index nad conversations pre retweet\_count a pre content. Porovnajte ich dĺžku vytvárania. Prečo je tu taký rozdiel? Čím je ovplyvnená dĺžka vytvárania indexu a prečo?

Z obržku 14 a 15 vidíme, že dlhšie trvá vytvorenie indexu pre content, pretože retweet\_count je hodnota typu int4 a contest je text, ktorý môže mať rôznu dĺžku.

CREATE INDEX

Query returned successfully in 17 secs 827 msec.

Obr. č. 15: Výsledok pre retweet\_count

CREATE INDEX

Query returned successfully in 2 min 18 secs.

Obr. č. 16: Výsledok pre content

# 8. Porovnajte indexy pre retweet\_count, content, followers\_count, name,... v čom sa líšia pre nasledovné parametre: počet root nódov, level stromu, a priemerná veľkosť itemu. Vysvetlite.

Z obrázkoby 17 – 26 vidíme, že hodnoty followers\_count a retweet\_count sú identické, preto ďalej uvádzame len jednu z nich, je to z dôvodu, že obe hodnoty sú dátového typu int4. Pozorujeme, že pri zoradení priemernej veľkosti itemu vzostupne máme nasledovné indexy pre: username, name, content, followers\_count. Ďalej pozorujeme, že vzostupne pocet root nódov máme pre indexy nasledovne: followers\_count, username, name, content. Level stromu je zoradený vzostupne pre indexy: followers\_count, username, name, content.

Z vyššie uvedeného jasne vidíme, že uvedené parametre stromov majú nižšie hodnoty, ak sa jedná o číselné typy ako int4, potom pre varchar(255) a nakoniec dátový typ text, ktorý je v naŠom prípade content a vidíme, že má najviac levelov stromu, najväčší počet root nódov a jednu z najväčší priemerných veľkostí itemu.

integer integer	bigint		bigint	bigint	bigint	last_cleanup_num_delpages bigint	double precision	allequalimage boolean
340322	4	209	2	209	2	0	-1	true

Obr. č. 17: Informácie metapage indexu pre followers\_count

<b>blkno</b> bigint	â	type "char" (1)	live_items integer	dead_items integer	avg_item_size integer	page_size integer	free_size integer	btpo_prev bigint	btpo_next bigint	btpo_level bigint	btpo_flags integer
	1	1	10	0	729	8192	812	0	2	0	1

Obr. č. 18: Informácie pagu indexu pre followers\_count

magic integer	version integer	â	root bigint	â	level bigint	fastroot bigint	fastlevel bigint	last_cleanup_num_delpages bigint	last_cleanup_num_tuples double precision	allequalimage boolean
340322		4	2	220	2	220	2	0	-1	true

Obr. č. 19: Informácie metapage indexu pre username

<b>blkno</b> bigint	â	type "char" (1)	live_items integer	dead_items integer	avg_item_size integer	page_size integer	free_size integer	btpo_prev bigint	btpo_next bigint	btpo_level bigint	btpo_flags integer	
	1	1	279	0	22	8192	800	0	2	0	1	

Obr. č. 20: Informácie pagu indexu pre username

magic integer	version integer	<b>a</b>	root bigint	level bigint	fastroot bigint	fastlevel bigint	last_cleanup_num_delpages bigint	last_cleanup_num_tuples double precision	allequalimage boolean
340322		4	245302	5	245302	5	0	-1	true

Obr. č. 21: Informácie metapage indexu pre content

<b>blkno</b> bigint	â	type "char" (1)	live_items integer	dead_items integer	avg_item_size integer	page_size integer	free_size integer	btpo_prev bigint	btpo_next bigint	btpo_level bigint	btpo_flags integer
	1	1	34	0	214	8192	716	0	2	0	1

Obr. č. 22: Informácie pagu indexu pre content

magic integer	â	version integer	â	root bigint	level bigint	fastroot bigint	fastlevel bigint	last_cleanup_num_delpages bigint	last_cleanup_num_tuples double precision	allequalimage boolean
34032	2		4	22250	3	22250	3	0	-1	true

Obr. č. 23: Informácie metapage indexu pre name

<b>blkno</b> bigint	â	type "char" (1)	live_items integer	dead_items integer	avg_item_size integer	page_size integer	free_size integer	btpo_prev bigint	btpo_next bigint	btpo_level bigint	btpo_flags integer
	1	T	74	0	95	8192	788	0	2	0	1
			_	31 Y 34	T C / ·	. 1					

## Obr. č. 24: Informácie pagu indexu pre name

magic integer	version integer	â	root bigint	â	level bigint	fastroot bigint	fastlevel bigint	last_cleanup_num_delpages bigint	last_cleanup_num_tuples double precision	allequalimage boolean
340322		4	- 2	209	2	209	2	0	-1	true

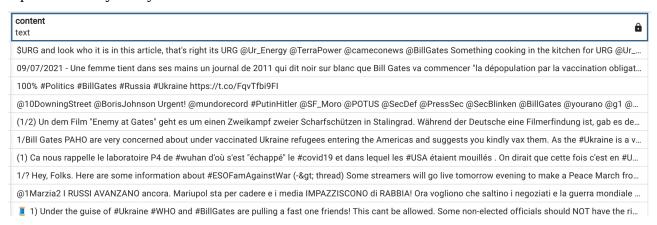
## Obr. č. 25: Informácie metapage indexu pre retweet\_count

<b>blkno</b> bigint	â	type "char" (1)	live_items integer	dead_items integer	avg_item_size integer	page_size integer	free_size integer	btpo_prev bigint	btpo_next bigint	btpo_level bigint	btpo_flags integer	
	1	1	10	0	729	8192	812	0	2	0	1	

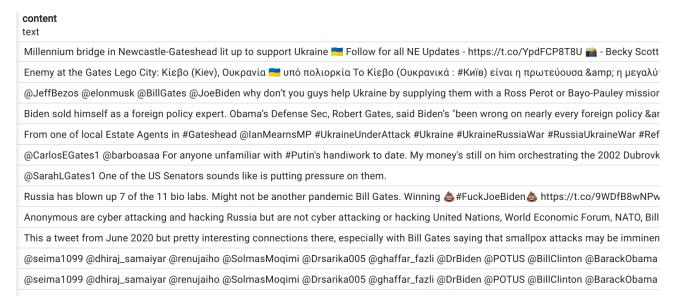
Obr. č. 26: Informácie pagu indexu pre retweet\_count

## 9. Vyhľadajte v conversations content meno "Gates" na ľubovoľnom mieste a porovnajte výsledok po tom, ako content naindexujete pomocou btree. V čom je rozdiel a prečo?

Z obrázkov 27 a 28 jasne vidíme, že btree index má utriedený výstup, pretože pri vytváraní indexu je už predvolene nastavená utrieďovacá metóda. Zároveň z obrázkov 29 a 30 pozorujeme rozdiel v rýchlosti, pri oboch prípadoch bol použitý paralelizmus, avšak použitie indexu dosahuje lepšie časové výsledky.



Obr. č. 27: Vzorok výsledku úlohy 9 pre index. Kompletný výsledok je priložený v príslušnom CSV súbore.



Obr. č. 28: Vzorok výsledku úlohy 9 bez indexu. Kompletný výsledok je priložený v príslušnom CSV súbore.

# QuERY PLAN text Gather (cost=1000.81..758593.42 rows=2854 width=157) (actual time=9.803..3097.524 rows=4199 loops=1) Workers Planned: 2 Workers Launched: 2 > Parallel Index Only Scan using idx\_conversations\_content on conversations (cost=0.81..757308.02 rows=1189 width=157) (actual time=29.251..3045.986 rows=1400 loops=3) Filter: (content ~~ %Gates%::text) Rows Removed by Filter: 10780937 Heap Fetches: 2421903 Planning Time: 0.608 ms JIT: Functions: 3 Options: Inlining true, Optimization true, Expressions true, Deforming true Timing: Generation 0.340 ms, Inlining 62.592 ms, Optimization 6.786 ms, Emission 6.988 ms, Total 76.705 ms Execution Time: 3102.632 ms

Obr. č. 29: Výsledok z explain analyse pre index.

IERY PLAN ct	
other (cost=1000.001182814.42 rows=2854 width=157) (actual time=37.6604669.384 rows=4199 loops=1)	
orkers Planned: 2	
orkers Launched: 2	
Parallel Seq Scan on conversations (cost=0.001181529.02 rows=1189 width=157) (actual time=54.2534645.172 rows=1400 loop	)s=3)
ter: (content ~~ '%Gates%'::text)	
ows Removed by Filter: 10780937	
anning Time: 0.749 ms	
Γ:	
nctions: 12	
otions: Inlining true, Optimization true, Expressions true, Deforming true	
ming: Generation 0.644 ms, Inlining 99.799 ms, Optimization 22.655 ms, Emission 15.568 ms, Total 138.666 ms	
ecution Time: 4674.553 ms	

Obr. č. 30: Výsledok z explain analyse bez indexu.

## 10. Vyhľadajte tweet, ktorý začína "There are no excuses" a zároveň je obsah potenciálne senzitívny (possibly sensitive). Použil sa index? Prečo? Ako guery zefektívniť?

Ako vidno z obr. č. 31, tak je iba 1 tweet, ktorý začína danou frázou. Ako vidno z obr. č. 32, tak bol použitý paralelný sekvenčný sken a nie index, pretože hľadáme frázu, ktorá môže ďalej pokračovať, nie je to čisto porovnávacia operácia(napr. rovná sa), kedy je btree efektívny. Zefektívnenie nastalo, keď sme zvýšili paralelných workerov, viď obr. 33. Alternatívne by bolo možné použiť FTS operácie a indexy typu GIN alebo GiST.

#### content

text

There are no excuses, zero, non, zilch, nada. The US and NATO have been arming scum, 14000 dead by Zelensky and Azov's hand.

Obr. č. 31: Výsledok úlohy 10.

#### **QUERY PLAN**

text

Gather (cost=1000.00..1182532.12 rows=31 width=157) (actual time=1352.281..1357.649 rows=1 loops=1)

Workers Planned: 2

Workers Launched: 2

-> Parallel Seq Scan on conversations (cost=0.00..1181529.02 rows=13 width=157) (actual time=1320.096..1338.005 rows=0 loops=3)

Filter: (possibly\_sensitive AND (content ~~ 'There are no excuses%'::text))

Rows Removed by Filter: 10782337

Planning Time: 0.536 ms

JIT:

Functions: 12

Options: Inlining true, Optimization true, Expressions true, Deforming true

Timing: Generation 0.923 ms, Inlining 97.186 ms, Optimization 31.132 ms, Emission 22.493 ms, Total 151.734 ms

Execution Time: 1358.086 ms

Obr. č. 32: Výsledok explain analysis úlohy 10 pri max parallel workers per gather = 2.

#### **QUERY PLAN**

text

Gather (cost=1000.00..1071820.62 rows=31 width=157) (actual time=482.994..488.026 rows=1 loops=1)

Workers Planned: 7

Workers Launched: 7

-> Parallel Seq Scan on conversations (cost=0.00..1070817.52 rows=4 width=157) (actual time=463.075..466.070 rows=0 loops=8)

Filter: (possibly\_sensitive AND (content ~~ 'There are no excuses%'::text))

Rows Removed by Filter: 4043376

Planning Time: 0.517 ms

JIT:

Functions: 32

Options: Inlining true, Optimization true, Expressions true, Deforming true

Timing: Generation 3.037 ms, Inlining 366.214 ms, Optimization 92.822 ms, Emission 64.749 ms, Total 526.821 ms

Execution Time: 488.315 ms

Obr. č. 33: Výsledok explain analysis úlohy 10 pri max\_parallel\_workers\_per\_gather = 8.

11. Vytvorte nový btree index, tak aby ste pomocou neho vedeli vyhľadať tweet, ktorý končí reťazcom "https://t.co/pkFwLXZlEm" kde nezáleží na tom ako to napíšete. Popíšte čo jednotlivé funkcie robia.

Obr. č. 35 ukazuje výsledok úlohy 11 o veľkosti jedného riadku, pričom obrázok 34 ukazuje, že bol použitý paralelný index sken. Bolo potrebné použiť funkciu LOWER, ktorá dá veľké písmena tweetu na malé a funkciu REVERSE, ktorá obráti hodnotu stringu, pretože v tomto prípade bude tweet začínať na daný reťazec.



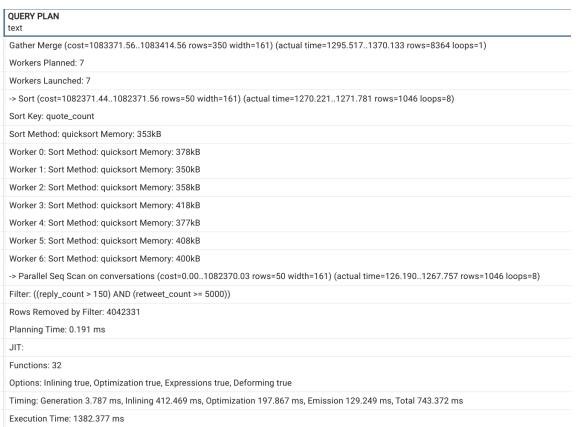
Obr. č. 35: Výsledok úlohy 11.

12. Nájdite conversations, ktoré majú reply\_count väčší ako 150, retweet\_count väčší rovný ako 5000 a výsledok zoraďte podľa quote\_count. Následne spravte jednoduché indexy a popíšte ktoré má a ktoré nemá zmysel robiť a prečo. Popíšte a vysvetlite query plan, ktorý sa aplikuje v prípade použitia jednoduchých indexov.

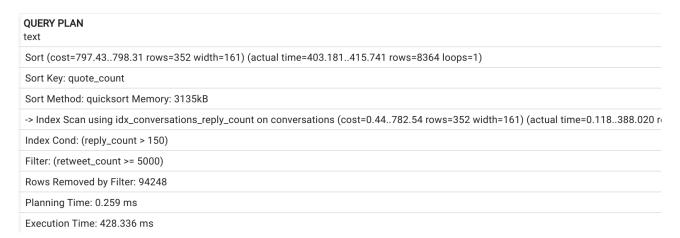
Výsledok úlohy je v priloženom CSV súbory, vzorka výsledku je vidieť na obrázku 36. Najprv sa vykonal paralelný sekvenčný sken(obr. č. 37), avšak vytvorili sme 3 indexy. Neoplati sa index nad quote\_count, ktorý sa ani nepoužije. Lepší čas dosahuje index nad reply\_count(obr. č. 38, čas je viac ako trojnásobne lepší) a najlepší čas dosahuje index nad content a s príslušnou podmienkou, čas je rádovo lepší len v desiatkách ms, viď obrázok 39.

content text
\$50   700K • 5 Hours RT & Me (4)
\$50   700K • 4 Hours RT & Amp; Follow Me (4)
\$100 • 1,4 JT   12 HOURS RT & amp; Follow me (4)
\$50   700.000 IDR • End 8 hour RT Like & Down the state of the state o
What an asshole. #Putin knows #angelamerkel is afraid of dogs yet brings his anyway. Next time she should bring gays. http://t.co/THdbxVfXT6
\$100 • 1,4 JT   12 HOURS RT & Dlow @DukeShiller ( )
\$150 in 24 hours 🕑 🌮 🍿 -RT + Follow @Imaginary_Apes @metarpgnft @TomorrowNFTland
\$50 • 700K   4 HOURS RT + Follow @Missypromotes
\$150 in 24 hours See 3 -DT + Follow @CrystoDuseD + @crystogod1101

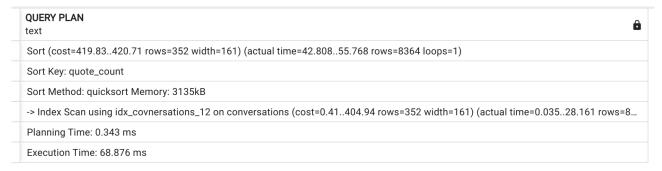
Obr. č. 36: Vzorka výsledku úlohy 12. Kompletný výsledok je priložený v príslušnom CSV súbore.



Obr. č. 37: Výsledok explain analyse úlohy 12 bez vytvorených indexov.



Obr. č. 38: Výsledok explain analyse úlohy 12 s vytvoreným indexom nad reply\_count.



Obr. č. 39: Výsledok explain analyse úlohy 12 s vytvoreným indexom nad content a s podmienkou.

## 13. Na predošlú query spravte zložený index a porovnajte výsledok s tým, kedy sú indexy separátne. Výsledok zdôvodnite. Popíšte použitý query plan. Aký je v nich rozdiel?

Zložený index bol vytvorený s podmienkou a nad stĺpcami content, reply\_count, retweet\_count a quote\_count. Obrázok 40 ukazuje, že plánovač vybral zložený index z dôvodu najnižšieho ocenenia utriedenia(sort cost), pričom vykonanie dopytu je o niečo pomalšie oproti najlepšiemu výsledku z predchádzajúcej úlohy, ale stále v desiatkach ms. Na obrázkoch 37-40 zároveň pozorujeme znižovanie sort cost(najväčší je pri použití bez indexov a najnižší pri zloženom indexe).

QUERY PLAN text	
Sort (cost=65.6866.56 rows=352 width=161) (actual time=62.32472.425 rows=8364 loops=1)	
Sort Key: quote_count	
Sort Method: quicksort Memory: 3113kB	
-> Index Only Scan using idx_conversations_13 on conversations (cost=0.4150.79 rows=352 width=161) (actual time=0.03550.247 rows=8	364 loops=1
Heap Fetches: 8139	
Planning Time: 4.964 ms	
Execution Time: 82.453 ms	

Obr. č. 40: Výsledok explain analyse úlohy 13 s vytvoreným zloženým indexom a s podmienkou.

14. Napíšte dotaz tak, aby sa v obsahu konverzácie našlo slovo "Putin" a zároveň spojenie "New World Order", kde slová idú po sebe a zároveň obsah je senzitívny. Vyhľadávanie má byť indexe. Popíšte použitý query plan pre GiST aj pre GIN. Ktorý je efektívnejší?

V našom prípade je výhodnejší časovo GiST index, ktorý má trvanie o polovicu menšie ako GIN, pričom v rámci GIN indexu bol použitý Bitmap Heap Scan a Bitman Index Scan.

# QUERY PLAN text Bitmap Heap Scan on conversations (cost=100.11..134.58 rows=31 width=157) (actual time=0.026..0.054 rows=5 loops=1) Recheck Cond: (possibly\_sensitive AND (content ~~ '%Putin%New World Order%'::text)) Heap Blocks: exact=5 -> Bitmap Index Scan on idx\_conversations\_gin (cost=0.00..100.10 rows=31 width=0) (actual time=0.016..0.018 rows=5 loops=1)

Planning Time: 17.892 ms

Execution Time: 0.092 ms

Obr. č. 41: Výsledok explain analyse úlohy 14 pre GIN index.

con en in i jordani dispraisi disanj i i pre ciri indeni
QUERY PLAN
text
Index Scan using idx_conversations_gist on conversations (cost=0.1235.64 rows=31 width=157) (actual time=0.0090.022 rows=5 loops=1)
Planning Time: 1.118 ms
Execution Time: 0.041 ms

Obr. č. 42: Výsledok explain analyse úlohy 14 pre GiST index.

## 15. Vytvorte vhodný index pre vyhľadávanie v links.url tak, aby ste našli kampane z 'darujme.sk'. Ukážte dotaz a použitý query plan. Vysvetlite prečo sa použil tento index.

Index bol použitý, viď obrázok 45, kvôli nízkemu sort cost. Pre vytvorenie indexu sme využili podmienku uvedenú na obrázku 43.

```
CREATE INDEX idx_url ON links USING GIN(to_tsvector('simple', url)) WHERE url LIKE '%darujme.sk%';

EXPLAIN ANALYSE

SELECT url FROM links WHERE url LIKE '%darujme.sk%';
```

Obr. č. 43: Query pre index a pre SELECT pre úlohu 15.

#### url

character varying (2048)

https://charita.darujme.sk/ukrajina/

https://clovekvohrozeni.darujme.sk/pomoc-ukrajina

https://redcross.darujme.sk/pomahame-ukrajine

https://redcross.darujme.sk/pomahame-ukrajine/

https://zvieraciombudsman.darujme.sk/animal-support-at-ukrainian-slovak-borders/

Obr. č. 44: Výsledok úlohy 15.

## **QUERY PLAN**

text

Bitmap Heap Scan on links (cost=16.75..1141.43 rows=1019 width=66) (actual time=0.021..0.040 rows=5 loops=1)

Recheck Cond: ((url)::text ~~ '%darujme.sk%'::text)

Heap Blocks: exact=5

-> Bitmap Index Scan on idx\_url (cost=0.00..16.50 rows=1019 width=0) (actual time=0.013..0.014 rows=5 loops=1)

Planning Time: 0.570 ms

Execution Time: 0.068 ms

Obr. č. 45: Výsledok explain analyse úlohy 15.

16. Vytvorte query pre slová "Володимир" а "Президент" pomocou FTS (tsvector a tsquery) v angličtine v stĺpcoch conversations.content, authors.decription a authors.username, kde slová sa môžu nachádzať v prvom, druhom ALEBO treťom stĺpci. Teda vyhovujúci záznam je ak aspoň jeden stĺpec má "match". Výsledky zoradíte podľa retweet\_count zostupne. Pre túto query vytvorte vhodné indexy tak, aby sa nepoužil ani raz sekvenčný scan (správna query dobehne rádovo v milisekundách, max sekundách na super starých PC). Zdôvodnite čo je problém s OR podmienkou a prečo AND je v poriadku pri joine.

Výsledok úlohy je na obrázku 46, explain analyse obr. 47. Indexy boli vytvorené, sú odovzdané v príslušnom SQL súbore, avšak výsledok sa nepodarilo zrýchliť.

username character varying (255)	description text	content text
DefenceU	Official page of the Ministry of Defense of Ukraine	Президент України Володимир Зеленський відвідав у госпіталі поранени
DefenceU	Official page of the Ministry of Defense of Ukraine	
UkrArmyBlog	СЛАВА УКРАЇНІ ТА ЇЇ ВОЇНАМ! Неофіційна сторінка	В бою загинув льотчик-винищувач Олександр Оксанченко. Він був одним
DefenceU	Official page of the Ministry of Defense of Ukraine =	Головнокомандувачу ЗС України генерал-лейтенанту Валерію Залужному
UkrArmyBlog	СЛАВА УКРАЇНІ ТА ЇЇ ВОЇНАМ! Неофіційна сторінка	!   № Майже 9000 вбитих 🗃 окупантів за один тиждень війни! "Кожен
verkhovna_rada	Офіційна сторінка Верховної Ради України  Official	! !   Президент України Володимир Зеленський підписав заявку на
verkhovna_rada	Офіційна сторінка Верховної Ради України  Official	! ! Сьогодні, 1 березня, Президент України Володимир Зеленський вис
UkrArmyBlog	СЛАВА УКРАЇНІ ТА ЇЇ ВОЇНАМ! Неофіційна сторінка	Президент 🔜 Володимир Зеленський підписав указ № 74/2022 про прис
ua_industrial	My name is Pavlo. Here the whole world reads about t	Президент Володимир Зеленський підписав указ про присвоєння звання

Obr. č. 46: Vzorka výsledku úlohy 16. Kompletný výsledok je priložený v príslušnom CSV súbore.

QUERY PLAN text	
Gather Merge (cost=14	176425.201496295.50 rows=161735 width=248) (actual time=186907.254187655.047 rows=1041 loops=1)
Workers Planned: 7	
Workers Launched: 7	
-> Sort (cost=1475425.	081475482.84 rows=23105 width=248) (actual time=186862.390186866.241 rows=130 loops=8)
Sort Key: conversations	s.retweet_count DESC
Sort Method: quicksort	Memory: 139kB
Worker 0: Sort Method:	quicksort Memory: 91kB
Worker 1: Sort Method:	quicksort Memory: 86kB
Worker 2: Sort Method:	quicksort Memory: 84kB
Worker 3: Sort Method:	quicksort Memory: 107kB
Worker 4: Sort Method:	quicksort Memory: 113kB
Worker 5: Sort Method:	quicksort Memory: 87kB
Worker 6: Sort Method:	quicksort Memory: 83kB
-> Parallel Hash Join (c	ost=159447.291473750.44 rows=23105 width=248) (actual time=37565.133186865.466 rows=130 loops=8)
Hash Cond: (conversat	ions.author_id = authors.id)
Join Filter: (to_tsvector	('english'::regconfig, (((((authors.username)::text    ' '::text)    authors.description)    ' '::text)    conversations.content)) @@ to_tsquery('Володимир & Президент'::text)
Rows Removed by Join	ı Filter: 4043246
-> Parallel Seq Scan on	conversations (cost=0.001059265.02 rows=4621002 width=169) (actual time=18.42016265.645 rows=4043376 loops=8)
-> Parallel Hash (cost=	127437.35127437.35 rows=1179035 width=95) (actual time=2505.0232505.027 rows=736897 loops=8)
Buckets: 524288 Batch	ies: 16 Memory Usage: 51648kB
-> Parallel Seq Scan on	authors (cost=0.00127437.35 rows=1179035 width=95) (actual time=172.8451293.135 rows=736897 loops=8)
Planning Time: 0.427 m	ns
JIT:	
Functions: 96	
Options: Inlining true, O	Optimization true, Expressions true, Deforming true
Timing: Generation 6.9	57 ms, Inlining 467.779 ms, Optimization 592.738 ms, Emission 322.600 ms, Total 1390.074 ms

Obr. č. 47: Výsledok z explain analyse úlohy 16.