# 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

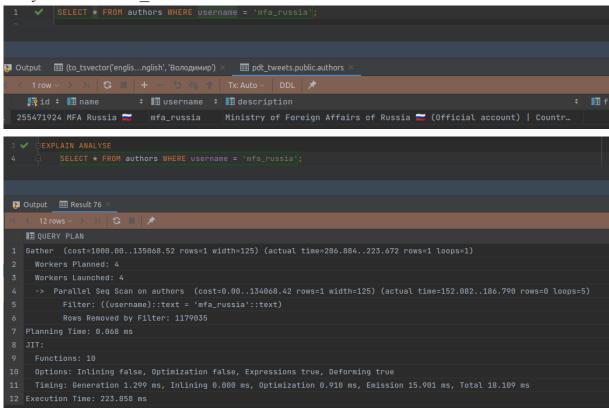
Predmet: Pokročilé databázové technológie

Zimný semester 2022/2023

# 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.

```
12
13 (1) ALTER TABLE authors SET (parallel_workers = 1025);

[22023] ERROR: value 1025 out of bounds for option "parallel_workers"

Detail: Valid values are between "0" and "1024".
```

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:

reštartovať DB. Toto všetko mi teda určí **maximálny** počet workerov, čo ale neznamená, že ich query planner vždy všetkých využije. Proces určovania počtov 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

```
■ QUERY PLAN

Gather (cost=1000.00..116718.06 rows=1 width=125) (actual time=375.984..4243.039 rows=1 loops=1)

Workers Planned: 1024

Workers Launched: 1023

-> Parallel Seg Scan on authors (cost=0.00..115717.96 rows=1 width=125) (actual time=10.219..10.499 rows=0 loops=1024)

Filter: ((username)::text = 'mfa_russia'::text)

Rows Removed by Filter: 5757

Planning Time: 0.061 ms

JIT:

Functions: 2048

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

Timing: Generation 1076.040 ms, Inlining 0.000 ms, Optimization 33.943 ms, Emission 996.222 ms, Total 2106.204 ms

Execution Time: 4243.213 ms
```

```
## QUERY PLAN

1 Gather (cost=1808.08..125857.31 rows=1 width=125) (actual time=168.257..184.850 rows=1 loops=1)

2 Workers Planned: 8

3 Workers Launched: 8

4 -> Parallel Seq Scan on authors (cost=0.80..124857.21 rows=1 width=125) (actual time=121.377..135.808 rows=8 loops=9)

5 Filter: ((username)::text = 'mfa_russia'::text)

6 Rows Removed by Filter: 655019

7 Planning Time: 8.898 ms

8 JIT:

9 Functions: 18

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

11 Timing: Generation 2.878 ms, Inlining 8.888 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

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:

### 4. Followers count

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);

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

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

Ploutput  Result 140 ×

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

```
DNOP INDEX authors_followers_count_index DN authors
USEND bytes

(followers_count);

(
```

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.

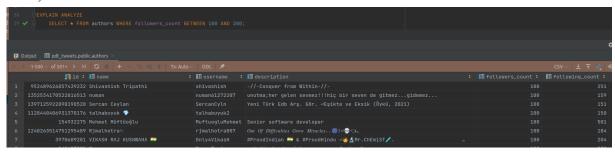
Na dosiahnutie Bitmap Heap scan som potreboval znížiť rozsah na <100, 110>, pretože pri väčšom rozsahu nefungoval.

## Výsledky queries pre bod 4 a 5:

## <100, 200> bez indexu



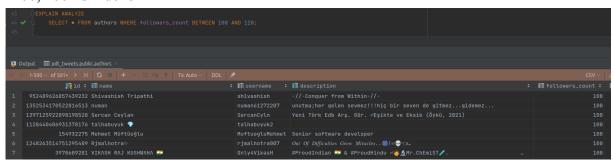
#### <100, 200> s indexom



#### <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:

```
| SEXPLAIN ANALYZE | TINSERT INTO authors (id, name, username, description, followers_count, following_count, tweet_count, listed_count) | Tinsert into authors (id, name, username, description, following_count, tweet_count, listed_count) | Tinsert into authors | Tinsert into
```

#### Vkladanie po dropnutí indexov:

```
| State | Stat
```

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.

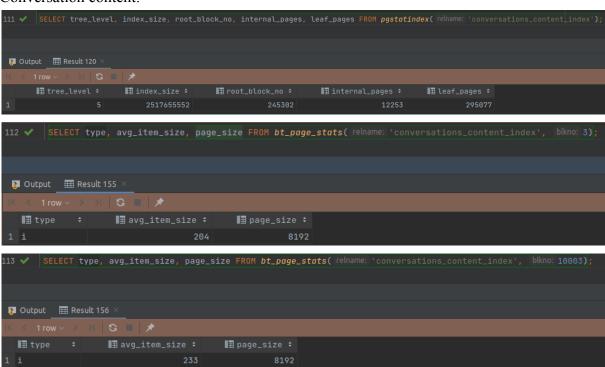
# 7. Dĺžka vytvárania indexu

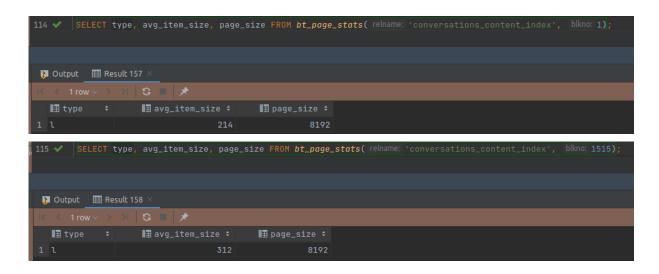
Ako vidno, index nad kontentom konverzácií sa vytvára 4-násobne pomalšie. Dlžka vytvárania indexu je ovplyvnená viacerými faktormi - softvérom, konfiguráciou DB, hardvérom, počtom riadkov tabuľky, vyťaženosťou zariadenia,... V tomto prípade je ale väčšina týchto "externých" parametrov rovnaká.

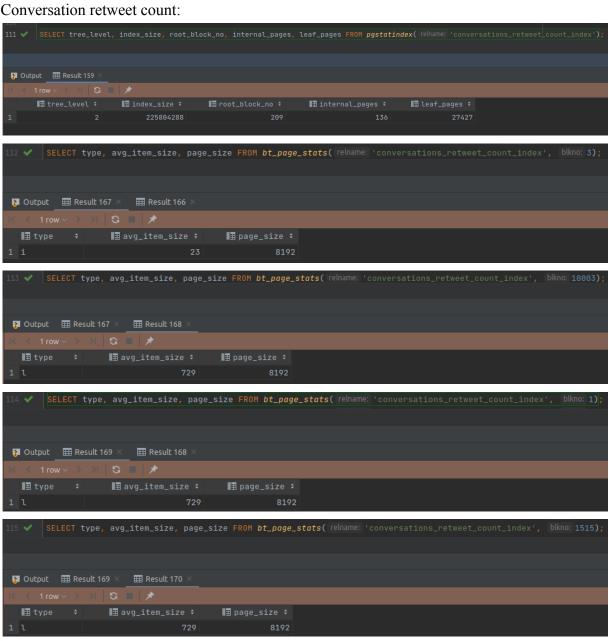
Rozdiel medzi contentom a retweet\_countom je ten, že retweet\_count obsahuje jedno číslo pre každý riadok tabuľky. Pri contente ale máme dlhé vety, ktoré obsahujú desiatky slov, ktoré je celé potrebné uložiť do listov stromu a je potrebné k ním spraviť cestu. Je potrebné nájsť väčšie bloky voľnej pamäte, 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

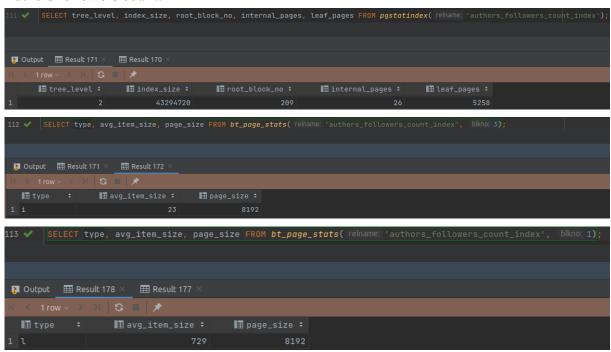
#### Conversation content:



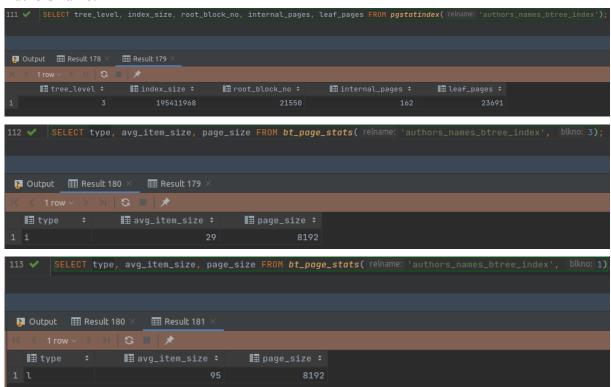




#### Authors followers count:



#### 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

$URG and look who it is in this article, that's right its URG-@Ur_Energy @TerraPower @cameconews @BillGates & Something cooking in the kitchen _
80/87/2021 - 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...
100% #Politics #BillGates #Russia #Ukraine https://t.co/Fqv7fbi9FI

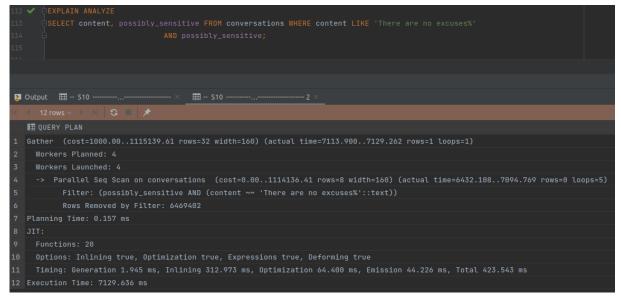
@1800wwingStreet @BorisJohnson Urgent! @wundorecord #PutinHitler-@SF_Moro @POTUS @SecDef @PressSec @SecBlinken @BillGates @yourano @gl-@ @Metropo...
(1/2) Un dem Film "Enemy at Gates" geht es um einen Zweikampf zweier Scharfschützen in Stalingrad. Während der Deutsche eine Filmerfindung ist, _
1/Bill Gates PAHO are very concerned about under vaccinated Ukraine refugees entering the Americas and suggests you kindly vax them. As the #Ukr...
(1) Ca nous rappelle le laboratoire P4 de #wuhan d'où s'est "échappé" le #covidi9 et dans lequel les #USA étaient mouillés . On dirait que cette.
1/?a-Mey, Folks. Here are some information about #ESDFamAgainstWar (-$qt; thread)& Some streamers will go live tomorrow evening to make a Peace ...
@ @Harzia2 I RUSSI AVANZANO ancora. Mariupol sta per cadere e i media IMPAZZISCONO di RABBIA! Ora vogliono che saltino i negoziati e la guerra mo...

$\frac{1}{2}\text{1}\text{ Under the guise of #Ukraine #WHO and #BillGates are pulling a fast one friends! This cant be allowed. Some non-elected officials should NO...

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.
2 "2017/ #KlausSchwab: & "Gurur duyduğumuz şey, #WEF Genç küresel liderlerinizle @Uevlerin kabinelerine, mötuz etmenizdir." & YoungGlobalLeaders & 202 (de...
2 "2017/ #KlausSchwab: & "Gurur duyduğumuz şey, #WEF Genç küresel liderlerinizle @Uevlerin kabinelerine, mötuz etmenizdir." & YoungGlobalLeaders & 202 (de...)
2 "2017/ #KlausSchwab: & "Gurur duyduğumuz şey, #WEF Genç küresel liderlerinizle @Uevlerin kabinelerine, mötuz etmenizdir." & YoungGlobalLeaders & 202
```

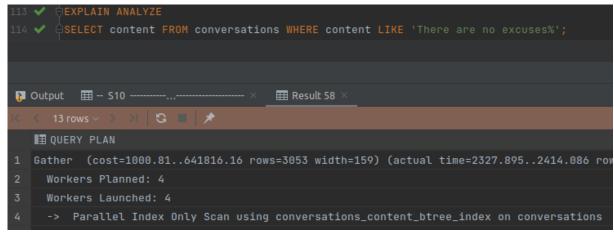
Ako som už vyššie uvádzal, výsledky sa pre nezaindexované tabuľky nemenia, maximálne tak poradie. Všetky contenty je možné nájsť v súbore *z02 09 schon.csv*.

### 10. "There are no excuses..." tweet



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:



#### Výsledok:



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

## Zlepšiť dopyt môžeme tak, že

a) pridáme index aj nad possibly sensitive

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

b) Vytvorenie jedného, zloženého indexu nad množinou (content, possibly\_sensitive)

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)

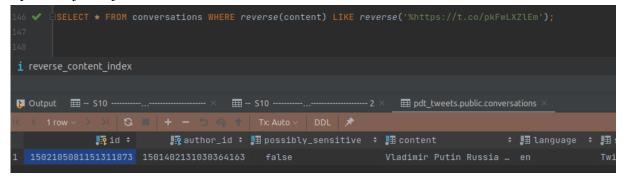
Takéto riešenie, podobne ako b) používa Paralelný index scan. Toto riešenie je však neporovnateľne najrýchlejšie, vďaka štýlu query a typu, akým vyberáme hodnoty. Tento index najskôr zahodí všetky výsledky, ktoré nie sú possibly\_sensitive - tie tvoria 98.9% výsledkov. Vďaka tomu je vyhľadávanie nad 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

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:



# 12. Jednoduché indexy nad countami

#### Bez indexu:

#### S indexami:

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.

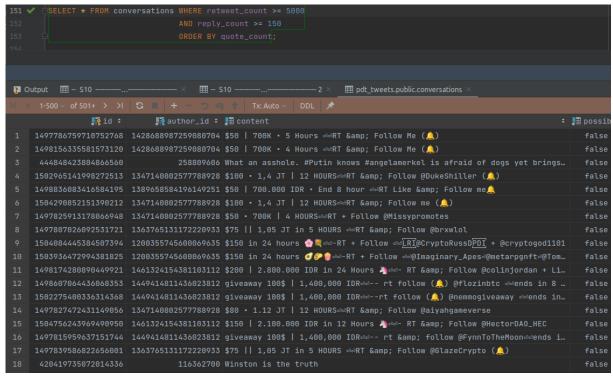
Najskôr sa vyfiltruje retweet\_count (bez indexu) cez filter - tento filter kontroluje každý riadok, ktorý je vybraný ako vhodný na základe indexu nad reply\_countom. Nakoniec sú výsledky usporiadané a to pomocou quicksortu nad quote\_countom. Vďaka využitiu indexov je táto query stále násobne rýchlejšia,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

# 13. Zložený index nad countami

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.



Všetky hodnoty sa dajú nájsť v súbore z02 12 schon.csv.

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

#### GIN:

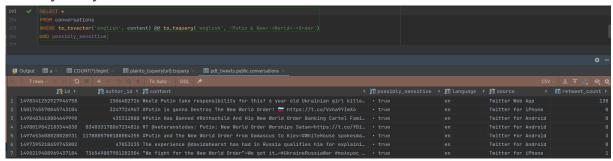
```
| STATE OF CONTROL OF
```

# GIN výsledky:

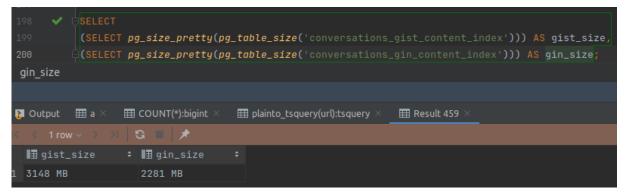


#### GIST:

#### GIST výsledky:



#### Veľkosť indexov:



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

```
CORRATE INDEX links_gin_trigram_url_index ON links

USING gin(url gin_trgm_ops);

VEXPLAIN ANALYZE

SELECT * FROM links WHERE url LIKE '%darujme.sk%'

SELECT * FROM links WHERE url LIKE '%darujme.sk%'

If QUERY PLAN

Bitmap Heap Scan on links (cost=67.82..1267.41 rows=1087 width=360) (actual time=10.384..10.391 rows=5 loops=1)

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

Heap Blocks: exact=5

-> Bitmap Index Scan on links_gin_trigram_url_index (cost=8.00..67.55 rows=1087 width=0) (actual time=10.356..10.356 rows=5 loops=1

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

Planning Time: 0.107 ms

Planning Time: 0.107 ms

Execution Time: 10.410 ms
```

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));
```

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', 'Boлодимир | Президент'))) AS content

FULL OUTER JOIN (SELECT id, username
FROM authors
WHERE (to_tsvector('english', username))

@@ to_tsquery('english', 'Bолодимир | 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', 'Bолодимир | Президент'))
AS description
ON description.id = content.author_id

GORDER 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:

	III content ÷	∎ username ‡	description ≎	I≣ retweet_count ≎
1	🗲 Володимир Кличко звернувся до демократичного світу із закликом нега			7351
2	Президент України Володимир Зеленський відвідав у госпіталі поранених…			5848
3	ాПрезидент України Володимир Зеленський:చాВсі ми тут - захищаємо наш…			5346
4	В бою загинув льотчик-винищувач Олександр Оксанченко. ⊿Він був одним …			2799
5	🗲 Володимир Кличко звернувся до громадян Німеччини, Австрії, Щвейцарі…			2628
6	На митинг в Санкт-Петербурге вышла 84-летняя Ирина Анохина, которая п…			2569
7	Володимир Ковальський.⊲У 2016-му році під час виконання бойового завд…			2408
8	Президент України @ZelenskyyUa поспілкувався зі звільненим із полону …			2200
9	Итого на сегодня в России:⊎- уничтожены почти все независимые СМИ⊎- в…			2118
10	"Президент Зеленський оголосив про формування нового підрозділу – «Ін…			2054
11	This is a rhetorical question. I know Putin is in his Ural bunker. Th…			1676
12	∢0 «Ми вдома і захищаємо Україну. Ніякої зброї ми складати не збираєм…			1580
13	Лукашенко, після розмови із Зеленським, жаліється, що "президент Укра…			1539
14	Президент Швейцарии заявил, что Швейцария больше не нейтральная стран…			1530
15	Головнокомандувачу ЗС України генерал-лейтенанту Валерію Залужному пр…			1489
16	Повертаючись до України зі США, у Варшаві мене прийняв Президент Поль…			1446
17	<b>∮</b> Сергій Васіч обороняв позиції на Київщині. <i>씓</i> Він знищив ворожу техні…			1400
18	Президент Молдовы потребовала от РФ немедленно вывести войска из терр…			1341
19	Президент України @ZelenskyyUa :∉Торік я попереджав лідерів @NATO: як…			1306
20	"Я поранений, залишилося недовго!" Заступник мера Нової Каховки встиг…			1253
21	Во время Второй мировой войны четыре брата из одной еврейской семьи з…			1246
22	‼‼ Майже 9000 вбитих 📨 окупантів за один тиждень війни!↵"Кожен ок…			1217
23	Пора уже всем россиянам с мозгами перестать быть «вне политики». ↵↵Не…			1212
24	Володимир Андрійченко поїхав забирати поранених та вбитих побратимів …			1134
25	!!!! Президент України Володимир Зеленський підписав заявку на чле…			1109
26	Кстати, а теперь ведь не кажется конспирологией, что президент Качинс	<null></null>	<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.

```
35 Planning Time: 0.259 ms
36 JIT:
37 Functions: 88
38 Options: Inlining true, Optimization true, Expressions true, Deforming true
39 Timing: Generation 8.510 ms, Inlining 1384.396 ms, Optimization 619.058 ms, Emission 305.446 ms, Total 2317.410 ms
40 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.

# 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  $\mathfrak{C}$ .