Pokročilé databázové technológie

Zadanie 2 – vyhľadávanie a indexovanie

Marek Adamovič

Cvičenie: Piatok 11:00 – 12:40

Ing. Ján Balažia, PhD. 2022/2023

Obsah

[1. otázka 3](#_Toc116732085)

[2. otázka 4](#_Toc116732086)

[3. otázka 6](#_Toc116732087)

[4. otázka 7](#_Toc116732088)

[5. otázka 8](#_Toc116732089)

[6. otázka 10](#_Toc116732090)

[7. otázka 11](#_Toc116732091)

[8. otázka 12](#_Toc116732092)

[9. otázka 14](#_Toc116732093)

[10. otázka 16](#_Toc116732094)

[11. otázka 18](#_Toc116732095)

[12. otázka 19](#_Toc116732096)

[13. otázka 21](#_Toc116732097)

[14. otázka 22](#_Toc116732098)

[15. otázka 24](#_Toc116732099)

[16. otázka 25](#_Toc116732100)

# otázka

**Otázka:**

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?

**Odpoveď:**

Obrázok, na ktorom je text

Automaticky generovaný popis

Plánovač vybral paralelný sekvenčný sken, keďže vie, že potrebujeme skenovať celú tabuľku. Pri skenovaní celej tabuľky ľahko rozdelí prácu na viacerých workerov, čo značne urýchli dopyt. Plánovač nemôže použiť index scan, keďže defaultný index je len pre stĺpec s PRIMARY KEY (teda v tomto prípade id) a iný index vytvorený nemáme.

# otázka

**Otázka:**

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)?

**Odpoveď:**

Na danom selecte pracovali 2 workeri. Slúžia na využitie paralelizmu pri dopytoch. Každý worker dostane pridelenú robotu, ktorej výsledky sa na záver dopytu spoja do jedného výsledku.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok, na ktorom je text

Automaticky generovaný popis

|  |  |  |
| --- | --- | --- |
| Max. workers | Used workers | Time (in ms) |
| 2 | **2** | **542.399** |
| 4 | **4** | **483.592** |
| 8 | **5** | **492.133** |

Prvým stropom je premenná **max\_parallel\_workers\_per\_gather**, ktorá udáva, koľko workerov môže byť použitých na jedno využitie paralelizmu. Defaultne je nastavená na hodnotu 2. Ďalším obmedzením je premenná **max\_parallel\_workers**, čo je maximálny počet pre všetky použitia paralelizmu (teda je to globálnejšie nastavenie). Defaultná hodnota tejto premennej je 8. Posledným obmedzením je premenná **max\_worker\_processes**, ktorá určuje, koľko procesov môže bežať na pozadí. Táto premenná sa dá zmeniť len počas štartu servera. Defaultná hodnota je 8.

Taktiež je nutné poznamenať, že plánovač vyberá počet workerov podľa toho, čo sa najviac oplatí. Vyšší počet workerov nemusí vždy znamenať lepší čas. Môže to byť spôsobené napríklad tým, že čím viacej workerov máme, tým viacej výsledkov sa musí na konci dopytu spájať do finálneho výstupu.

# otázka

**Otázka:**

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?

**Odpoveď:**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Plánovač v tomto prípade nepotrebuje viac workerov, keďže neprehľadávame tabuľku sekvenčne, ale podľa indexu (Index Scan). Zásadnú zmenu času spôsobilo, že vďaka indexu vieme, kde máme hľadať záznamy s názvom, ktorý máme v podmienke. Tým pádom nemusíme prehľadávať zbytok tabuľky. Je to podobný princíp ako keby v telefónnom zozname hľadáme konkrétne meno -> taktiež nemusíme prečítať celý telefónny zoznam. Zlepšili sme čas na úkor pamäte.

# otázka

**Otázka:**

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?

**Odpoveď:** **Obrázok, na ktorom je text

Automaticky generovaný popis**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Je tam veľký rozdiel v čase kvôli tomu, že plánovač použil v druhom prípade workerov, teda paralelizmus. O počte workerov rozhoduje plánovač na základe podmienky vo WHERE klauzule. Paralelný sekvenčný sken je v tomto prípade rýchlejší ako klasický sekvenčný sken.

# otázka

**Otázka:**

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?

**Odpoveď:**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Vytvorili sme index nad stĺpcom followers\_count (keďže naň sú smerované nasledujúce selecty)

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok, na ktorom je text

Automaticky generovaný popis

V prvom prípade sa použil Bitmap Heap Scan, ktorý využíva Bitmap Index Scan. Bitmap Index Scan najskôr vytvorí bitmapu podľa indexov, aby sa zistilo, ktoré bloky dát treba prečítať. V závislosti od (ne)dostatku pamäte môžu byť v tejto bitmape presné ukazovatele priamo na dáta alebo nepresné ukazovatele na stránku, kde je dát viacej. Pri nepresných ukazovateľoch potom musíme riešiť recheck condition, ktorý nám na konkrétnych dátach znovu skontroluje danú podmienku. Túto bitmapu používa Bitmap Heap Scan, aby vedel, ktoré dáta má prečítať. Index sa použil v oboch prípadoch, avšak paralelizmus sa použil len v druhom, vďaka čomu bol druhý dopyt značne rýchlejší.

# otázka

**Otázka:**

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

**Odpoveď:**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok, na ktorom je text

Automaticky generovaný popis

Insert bez indexov trval oveľa kratšie. To vďaka tomu, že keď vkladáme záznam a existujú indexy, musíme ho vložiť a usporiadať aj do týchto indexov, čo je značne pomalšie oproti vloženiu (len) do hlavnej tabuľky.

# otázka

**Otázka:**

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?

**Odpoveď:**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Dĺžka vytvárania indexu je ovplyvnená veľkosťou dát. Keďže stĺpec retweet\_count pozostáva z jedného BIGINT, čo je 8B, tak vytváranie indexu nad ním bude oveľa rýchlejšie ako nad stĺpcom content, kde sú dlhé stringy dát, ktoré zberajú oveľa viacej miesta. Indexovanie je z veľkej časti zoraďovanie a menšie dáta sa zoraďujú ľahšie.

# otázka

**Otázka:**

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.

**Odpoveď:**

**Obrázok, na ktorom je stôl

Automaticky generovaný popis**

Ako vidíme na obrázku, výsledky sa líšia v internal\_nodes, tree\_level a taktiež v priemernej veľkosti itemu. Veľkosť stromu (internal\_nodes a tree\_level) je ovplyvnená najmä počtom dát a veľkosťou itemov. Veľkosť itemu je zas ovplyvnená formou dát, teda je samozrejmé, že dlhé reťazce budú mať väčšiu veľkosť ako čísla.

# otázka

**Otázka:**

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?

**Odpoveď:**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Jediný markantnejší rozdiel vidíme v Planning Time, čo môže byť spôsobené tým, že pri existencii indexu uvažuje plánovač nad jeho použitím. Pri oboch dopytoch bol použitý paralelný sekvenčný sken (s rovnakým počtom workerov). Index nevieme použiť kvôli tomu, že používame wildcard % na hľadanie v LIKE.

# otázka

**Otázka:**

Vyhľadajte tweet, ktorý začína “There are no excuses” a zároveň je obsah potenciálne senzitívny (possibly\_sensitive). Použil sa index? Prečo? Ako query zefektívniť?

**Odpoveď:**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Index sa nepoužil, keďže to nie je možné, ak nepoznáme **presný text** (to znamená ak máme v LIKE „wildcards“ \_ alebo %), ktorý chceme vyhľadať. Aby sme query zefektívnili, môžeme vytvoriť index na prvých 20 znakov stĺpca content. Tým pádom vieme spraviť vyhľadávanie podľa indexu, kde tweet začína s „There are no excuses“, čo je presne 20 znakov.

**Obrázok, na ktorom je text

Automaticky generovaný popis**

# otázka

**Otázka:**

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.

**Odpoveď:**

Aby sme vedeli použiť v query index, musíme použiť ten istý princíp ako v 10. úlohe, teda nepoužívať vyhľadávanie pomocou „wildcards“ \_ a %. To zabezpečíme vytvorením indexu pravej strany contentu, konkrétne 23 znakov, čo je rovnako veľa ako má náš reťazec, čo plánujeme vyhľadávať. Funkcia RIGHT() oreže reťazec na daný počet znakov. Túto istú funkciu potom musíme použiť aj pri vyhľadávaní, a to pri podmienke WHERE. Aby naše vyhľadávanie nebolo case sensitive, použijeme funkciu UPPER aj pre dáta z tabuľky content, aj pre dáta, čo hľadáme. Tým pádom nebude záležať na veľkosti písmen, keďže všetky zmeníme na veľké.

Obrázok, na ktorom je text

Automaticky generovaný popis

# otázka

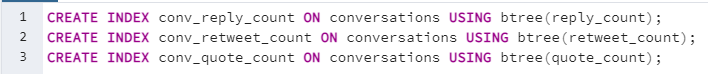
**Otázka:**

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.

**Odpoveď:**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

****

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Z query plánu vidíme, že sa použil len index vytvorený nad stĺpcom reply\_count. Je to z toho dôvodu, že pri využívaní indexu nedáva zmysel pre databázu použiť 2 rôzne indexy (pretože ak zredukujeme množinu výsledkov prvou podmienkou, tak v ďalšej podmienke už nepracujeme s dátami, nad ktorými bol index vytvorený). Ak by sme chceli použiť index na obe podmienky, potrebovali by sme zložený index.

# otázka

**Otázka:**

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

**Odpoveď:**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Z výsledku vidíme, že oproti separátnym indexom je zložený značne rýchlejší, a to z toho dôvodu, že pre obidve podmienky vo WHERE sa využije jeden a ten istý index (ako vidíme vo výstupe na riadku 5). Riešenie používajúce separátny index (v otázke 12) používa index len na prvú podmienku (riadok číslo 5 vo výstupe), zatiaľ čo druhá je riešená bez použitia indexu (riadok číslo 6-7).

# otázka

**Otázka:**

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?

**Odpoveď:**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Z výsledkov vidíme, že pre GiST sa použil klasický index scan. Na druhej strane pri indexe GIN sa použil Bitmap Heap Scan (ktorý sme vysvetlili v otázke číslo 5) a musela sa aj „rechecknúť“ naša podmienka. Podľa časov ľahko vidíme, že v tomto prípade bolo použitie GIN indexu oveľa efektívnejšie.

# otázka

**Otázka:**

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.

**Odpoveď:**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Použili sme index GIN využívajúci trigramy. Tsvector a tsquery neprichádzajú do úvahy, keďže sa hodia na vyhľadávanie slov a nie vyhľadávanie substringov (ako je napr. „darujme.sk“ v strede URL adresy). Na vyhľadávanie substringov sú ideálne trigramy. Tento index sa potom využije pri použití klasického LIKE s wildcards (% alebo \_).

# otázka

**Otázka:**

Vytvorte query pre slová "Володимир" a "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.

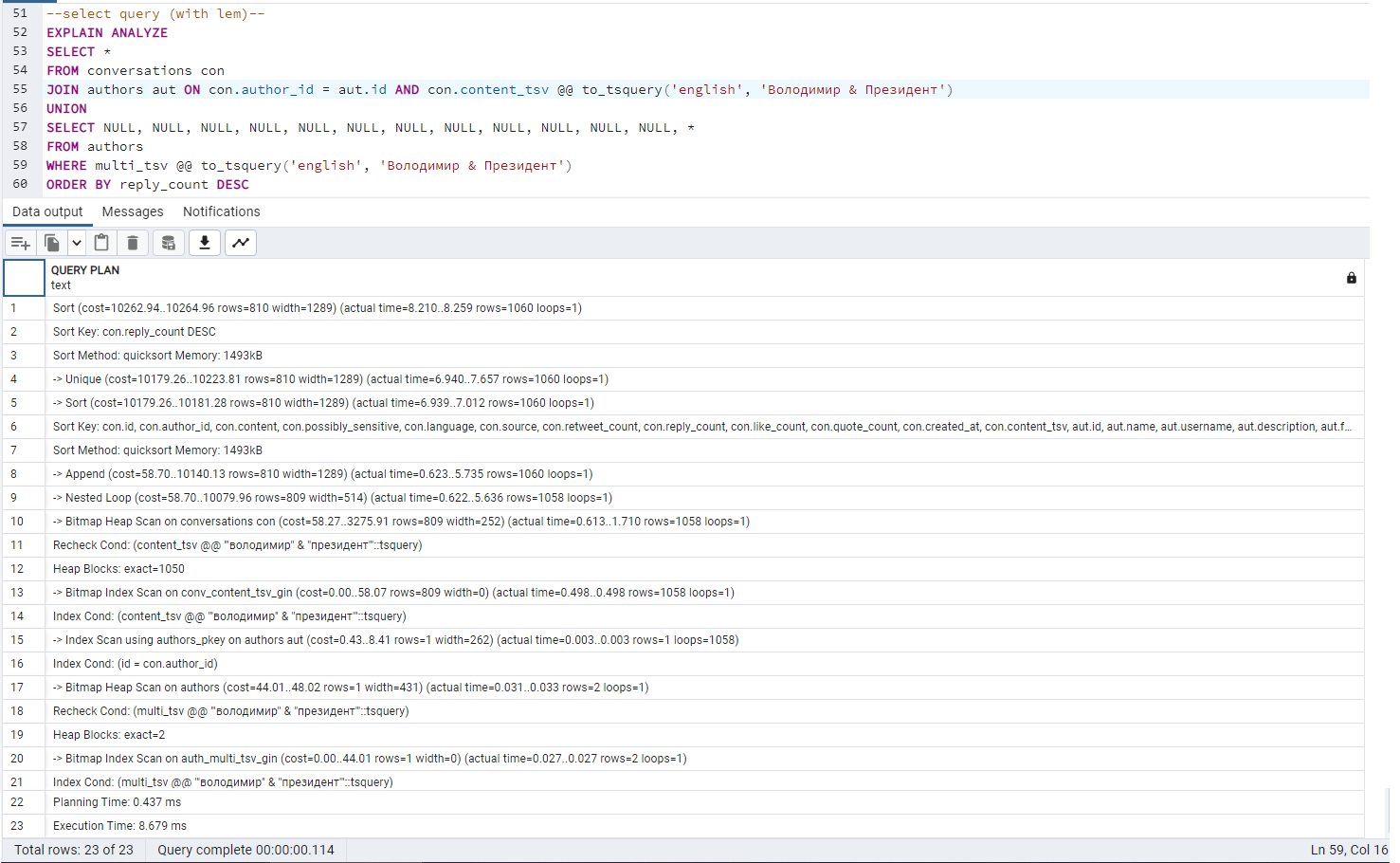
**Odpoveď:**

Ak sme vytvárali tsvector v angličtine (tým pádom slová v azbuke neboli lematizované), mali sme problém, že hľadané slová boli v authors.description a v authors.username vyskloňované (napríklad mali ‘a‘ na konci) a nenastala zhoda so slovami v tsquery. Preto sme sa rozhodli vyskúšať riešiť úlohu aj s lematizáciou, aj bez nej. Jedným z problémov, ktoré sme riešili, bol, ako priradiť konverzáciu k autorovi, ktorý má match v description (keďže jeden autor môže mať viacero konverzácií). Preto autorské zhody riešime samostatne a na konci ich pridáme pomocou UNION s tým, že chýbajúce údaje konverzácie vyplníme s NULLs. Zhody v conversations dopĺňame dátami z autorskej tabuľky pomocou JOIN. Ak by sme chceli použiť query aj na druhú tabuľku v rámci klauzuly JOIN, teda by sme použili OR, nepoužil by sa nám index a taktiež by sme kvôli prvej podmienke (author\_id = id) stratili záznamy o používateľoch, ktorí nemajú žiaden záznam v konverzáciach.

**S (ruskou) lematizáciou**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

****

**Bez lematizácie**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

(Bohužiaľ, index sa už nestihol spraviť pre riešenie bez lematizácie, a to z dôvodu pomalšieho počítaču a časového sklzu 🐸)