

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

twitter/postgres@PostgreSQL 13 ▾	
Query Editor	Query History
1 EXPLAIN ANALYZE SELECT * FROM accounts WHERE screen_name = 'realDonaldTrump';	
Data Output	Explain Messages Notifications
QUERY PLAN text	
1	Gather (cost=1000.00..223523.68 rows=1 width=118) (actual time=0.581..5545.225 rows=1 loops=1)
2	[...] Workers Planned: 2
3	[...] Workers Launched: 2
4	[...] -> Parallel Seq Scan on accounts (cost=0.00..222523.58 rows=1 width=118) (actual time=3428.130..5272.803 rows=0 loops=3)
5	[...] Filter: ((screen_name)::text = 'realDonaldTrump':text)
6	[...] Rows Removed by Filter: 3229079
7	Planning Time: 0.122 ms
8	Execution Time: 5545.350 ms

2. Na selecte z predošlej úlohy pracovali dvaja workery. Zdá sa, že zvýšenie počtu workerov neovplyvňuje čas – v mojom prípade trvalo vykonávanie query pred zvýšením počtu zväčša 4-5 sekúnd. Niekedy sa však query vykonalo podstatne rýchlejšie (1.7 – 2.1 sekundy). Podobné časy a podobné skoky vo vykonávaní nastávali aj pri zvýšení počtu workerov. Počet workerov je obmedzený podľa nastavenia tabuľky, respektíve servera.

Query Editor	Query History
1 2 SET max_parallel_workers_per_gather = 8; 3 EXPLAIN ANALYZE SELECT * FROM accounts WHERE screen_name = 'realDonaldTrump'; 4	
Data Output	Explain Messages Notifications
QUERY PLAN text	
1	Gather (cost=1000.00..197322.14 rows=1 width=118) (actual time=0.720..4002.562 rows=1 loops=1)
2	[...] Workers Planned: 5
3	[...] Workers Launched: 4
4	[...] -> Parallel Seq Scan on accounts (cost=0.00..196322.04 rows=1 width=118) (actual time=2782.134..3575.540 rows=0 loops=5)
5	[...] Filter: ((screen_name)::text = 'realDonaldTrump':text)
6	[...] Rows Removed by Filter: 1937447
7	Planning Time: 0.145 ms
8	Execution Time: 4002.588 ms

3. Plánovač využije v tomto prípade jedného workera, pretože nad daným stĺpcom je vytvorený index a teda sa zmení aj metóda na „index scan“ – tento prístup nie je paralelný a teda nevyužíva viac workerov. Zásadnú zmenu času ovplyvnilo vytvorenie indexu.

Query Editor		Query History
1		
2	SET max_parallel_workers_per_gather = 8;	
3	EXPLAIN ANALYZE SELECT * FROM accounts WHERE screen_name = 'realDonaldTrump';	
4		
5	CREATE INDEX index_screen_name_btree ON accounts USING btree(screen_name);	
6		
Data Output		Explain Messages Notifications
	QUERY PLAN	
	text	
1	Index Scan using index_screen_name_btree on accounts (cost=0.43..8.45 rows=1 width=118) (actual time=0.071..0.072 rows=1 loops=1)	
2	[...] Index Cond: ((screen_name)::text = 'realDonaldTrump':text)	
3	Planning Time: 0.164 ms	
4	Execution Time: 0.096 ms	

4. Správanie je iné ako v prvej úlohe. V tomto prípade bolo použité sekvenčné skenovanie. V prípade prvej úlohy išlo o **paralelné** sekvenčné skenovanie. Rozdiel oproti tretej úlohe je vo využití indexu – v tomto prípade nemáme vytvorený index a teda bolo použité sekvenčné skenovanie. Paralelné sa využíva v prípade malého počtu záznamov (resp. keď podmienka WHERE odfiltruje veľa záznamov). V takomto prípade sa využije viac workerov. V prípade veľkého počtu záznamov je zložité prenášať tieto záznamy medzi workermi. Nakoľko v tomto prípade pracujeme s veľkým počtom záznamov a nemáme vytvorený index, plánovač použil sekvenčné skenovanie.

Query Editor		Query History
7	EXPLAIN ANALYZE SELECT * FROM accounts WHERE followers_count BETWEEN 100 AND 200;	
8		
9		
Data Output		Explain Messages Notifications
	QUERY PLAN	
	text	
1	Seq Scan on accounts (cost=0.00..317444.57 rows=1296933 width=118) (actual time=0.025..3141.659 rows=1269496 loops=1)	
2	[...] Filter: ((followers_count >= 100) AND (followers_count <= 200))	
3	[...] Rows Removed by Filter: 8417742	
4	Planning Time: 0.152 ms	
5	Execution Time: 3183.624 ms	

5. „Bitmap Index Scan“ sa využíva, ak pracujeme s veľkým počtom záznamov. V takomto prípade sa neukladajú priamo dáta (konkrétne záznamy tabuľky), ale poloha potenciálnych záznamov (respektíve celé stránky). Následne „Bitmap Heap Scan“ dekoduje tieto údaje, aby našiel konkrétne záznamy. „Recheck Cond“ znamená, že tieto záznamy je potrebné znovu prekontrolovať.

Za normálnych okolností si „work_mem“ ukladá hľadané záznamy. Ak nie je dostatočne veľká, namiesto záznamu si uloží celú stránku (page). Následne „Recheck Cond“ znamená, že tieto stránky je potrebné znovu prekontrolovať, pretože sa v nich pochopiteľne nachádzajú aj také záznamy, ktoré nespĺňajú pôvodnú podmienku.

6

7

8

9

10

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

CREATE INDEX index_followers_count ON accounts (followers_count);

Data Output

Explain

Notifications

Messages

QUERY PLAN

text

1

2

3

4

5

6

7

8

Bitmap Heap Scan on accounts (cost=17834.00..311320.75 rows=1296933 width=118) (actual time=165.902..3399.132 rows=1269496 loops=1)

[...] Recheck Cond: ((followers_count >= 100) AND (followers_count <= 200))

[...] Rows Removed by Index Recheck: 6449000

[...] Heap Blocks: exact=39770 lossy=132186

[...] -> Bitmap Index Scan on index_followers_count (cost=0.00..17509.77 rows=1296933 width=0) (actual time=156.254..156.254 rows=1269496 loops=1)

[...] Index Cond: ((followers_count >= 100) AND (followers_count <= 200))

Planning Time: 2.358 ms

Execution Time: 3448.443 ms

6. Rozdiel je v skenovaní – v tomto prípade bolo použité sekvenčné skenovanie a nie skenovanie pomocou indexu. Je to z dôvodu, že podmienku spĺňa veľa záznamov, teda je lepšie skenovať úložisko sekvenčne a nie podľa indexu, nakoľko skákanie po pamäti pomocou indexu je pomalšia (respektíve drahšia/náročnejšia) operácia. Z toho dôvodu je v konečnom dôsledku sekvenčné skenovanie rýchlejšie ako skenovanie s využitím indexu. Ak by podmienka selectu vyfiltrovala „dostatočný“ počet záznamov, index by sa použil.

6

7

8

9

10

EXPLAIN ANALYZE SELECT * FROM accounts WHERE followers_count BETWEEN 100 AND 1000;

CREATE INDEX index_followers_count ON accounts (followers_count);

Data Output

Explain

Notifications

Messages

QUERY PLAN

text

1

Seq Scan on accounts (cost=0.00..317444.57 rows=4411742 width=118) (actual time=0.106..8348.820 rows=4382646 loops=1)

2

[...] Filter: ((followers_count >= 100) AND (followers_count <= 1000))

3

[...] Rows Removed by Filter: 5304592

4

Planning Time: 0.230 ms

5

Execution Time: 8603.712 ms

7. S použitím indexov trval insert 265ms. Bez indexov to trvalo 250ms. Proces som opakoval ešte raz a v tomto prípade trval insert s indexami 16 ms a bez indexov 6 ms. Výrazný časový rozdiel tu teda nevidím – môže sa jednať čisto o výkon môjho zariadenia. Avšak vo všeobecnosti by to pri indexoch malo trvať dlhšie, nakoľko sa okrem zápisu do tabuľky vykonáva aj zápis (respektíve určitá zmena) v indexoch a teda by to s indexami malo trvať dlhšie.

14

15 EXPLAIN ANALYZE INSERT INTO accounts(screen_name, name, description, followers_count, friends_count, statuses_count)

16 VALUES ('random_screen_name2', 'random name 2', 'description is not usefull at all second round', 100, 50, 10);

17

Data Output

Explain

Notifications

Messages

QUERY PLAN

text

1

Insert on accounts (cost=0.00..0.01 rows=1 width=888) (actual time=16.567..16.568 rows=0 loops=1)

2

[...] -> Result (cost=0.00..0.01 rows=1 width=888) (actual time=0.050..0.050 rows=1 loops=1)

3

Planning Time: 0.063 ms

4

Execution Time: 16.816 ms

```

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

```

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

8. Vytvorenie indexu nad tweets.content je omnoho dlhšie, pretože zaindexovať integer je jednoduchšie ako zaindexovať varchar, respektíve text. Aj čo sa týka hlavičiek, hlavičky indexov integerov sú kratšie ako hlavičky indexov textu, čo sa všetko odráža na celkovom čase potrebnom pre vytvorenie indexu.

```

22
23 CREATE INDEX index_retweet_count_btree ON tweets USING btree(retweet_count);
24 CREATE INDEX index_content_btree ON tweets USING btree(content);
25
26

```

	Data Output	Explain	Notifications	Messages
CREATE INDEX				
	Query returned successfully in 1 min 15 secs.			

```

23 CREATE INDEX index_retweet_count_btree ON tweets USING btree(retweet_count);
24 CREATE INDEX index_content_btree ON tweets USING btree(content);
25
26

```

	Data Output	Explain	Notifications	Messages
CREATE INDEX				
	Query returned successfully in 10 min 48 secs.			

9.

10. S indexom alebo bez neho, v oboch prípadoch bol čas vykonania približne rovnaký. Dokonca bolo použité paralelné sekvenčné skenovanie. Jediný rozdiel bol v čase plánovania. V prípade pred vytvorením indexu bol čas plánovania niekoľkonásobne väčší.

Pred vytvorením indexu:

28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983

12. Použitie indexu ovplyvnilo pridanie parametru „text_pattern_ops“. Vďaka tomuto parametru sa dáta voči vyhľadávanému výrazu porovnávajú po znakoch, čo je výhodou pri vyhľadávaní s použitím „LIKE“.

Query Editor	Query History
<pre>1 CREATE INDEX index_content_btree_with_parameter ON tweets USING btree(content text_pattern_ops); 2 3 EXPLAIN ANALYZE SELECT * FROM tweets t WHERE t.content LIKE 'The Cable and Deep State%'; 4</pre>	
Data Output	Explain
<div>QUERY PLAN</div> <div>text</div> <div>1 Index Scan using index_content_btree_with_parameter on tweets t (cost=0.81..8.83 rows=3033 width=385) (actual time=0.019..0.020 row...</div> <div>2 [...] Index Cond: ((content ~>=~ 'The Cable and Deep State%':text) AND (content ~<~ 'The Cable and Deep State%':text))</div> <div>3 [...] Filter: (content ~~ 'The Cable and Deep State%':text)</div> <div>4 Planning Time: 0.580 ms</div> <div>5 Execution Time: 0.045 ms</div>	

Pri vyhľadávaní „%Gates%“ sa index už nepoužil. Je to pravdepodobne pre to, že v predošlom prípade sme jasne definovali čím sa hľadaný výraz začína. V tomto prípade na hľadaný výraz z počiatku sedia všetky záznamy, tým pádom sa neoplatí používať index, ale plánovač porovnáva záznamy po poradí.

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

13. Nerozumiem aké funkcionality mám opísať. Pri vytváraní indexu je možné definovať aj obsah, respektíve hodnotu, ktorú chceme zaindexovať, pomocou podmienky WHERE:

<pre>34 CREATE INDEX index_content_btree_specific_qanon ON tweets USING btree(content) WHERE content ILIKE '%idiot #QAnon%'; 35</pre>	
Data Output	Explain
<div>CREATE INDEX</div> <div>Query returned successfully in 8 min 52 secs.</div>	

14. Z obrázkov nižšie môžeme vidieť, že po vytvorení indexov sa použili indexy nad atribútmi followers_count a friends_count. Vytvárať index nad statuses_count nemá zmysel, pretože tento atribút nijako nesúvisí s podmienkou WHERE, zatiaľ čo atribúty followers_count a friends_count súvisia a teda sa pre ne oplatí vytvárať index.

Pred vytvorením indexov:

```

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

```

Data Output Explain Notifications Messages

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

Po vytvoření indexov:

```

4
5 CREATE INDEX index_followers_count ON accounts(followers_count);
6 CREATE INDEX index_friends_count ON accounts(friends_count);
7 CREATE INDEX index_statuses_count ON accounts(statuses_count);
8

```

```

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

```

Data Output Explain Notifications Messages

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

15. Použitie zloženého indexu výrazne zrýchliło vyhľadávanie. Je to z dôvodu, že v podmienke sledujeme dva atribúty, pričom ak sú vytvorené dva indexy zvlášť na každý atribút, vyhľadávanie je pomalšie, ako keď vytvoríme zložený index nad oboma atribútmi. V prípade zloženého indexu sa indexujú rôzne „dvojice“ hodnôt v týchto atribútoch. Ak teda vyhľadávame konkrétne čísla, vďaka zloženému indexu vieme rýchlo zistiť, ktoré záznamy spĺňajú podmienku. V prípade jednoduchých indexov musím pri vyhľadávaní využiť oba indexy a na záver spraviť prienik medzi vyhľadanými záznamami, aby sme zistili, ktoré zo záznamov spĺňajú obe podmienky.

S použitím zloženého indexu nad všetkými atribútmi:

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

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

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

S použitím zloženého indexu nad atribútmi followers_count a friends_count:

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

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

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

16. V prípade zmeny podmienky sa opäť použilo sekvenčné vyhľadávanie (paralelné), pretože podmienku spĺňa omnoho viac záznamov a teda sa neoplatí skákať v pamäti podľa indexu. Takisto aj celový čas vyhľadávania je výrazne dlhší.

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

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