

Query 8

- Find the names of sailors who have reserved a red boat .

Note !

- Flags Hashjoin and HashAgg here were disabled for future after many trials and errors , I've discovered the best way to show the difference in terms of the cost and to beat the Postgres Query Optimizer Algorithm to be able to show indices effect and cost differences .
- The Execution time was changing by 10 Ms in each Execution which is considered high and I can't take it as a measurable Metric because it was Linux (Ubuntu) Operating System performance and I took permission from Prof. Wael as do not take it as my main objective I take the Overall Cost and Compare it .
- I removed the primary constraint to be able to remove the already built in B-tree index to simulate no-inices behavior (I asked Prof.).

Original Query

```
select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
```

Result Set

- 673 Rows

The screenshot shows the pgAdmin 4 interface. On the left is a sidebar with various icons for database management tasks. The main area has a title bar "pgAdmin 4" and a status bar at the bottom indicating "May 17 12:03 AM". The central part contains a "Properties" tab, a "SQL" tab, and a "Dependents" tab. Below these is a "Dashboard" section. The "Properties" tab is active, showing a tree view of the database structure under "schema3/postgres". The "SQL" tab contains two queries:

```
112 explain analyze select s.sname
113   from sailors s
114   where
115     s.sid in( select r.sid
116       from reserves r
117      where r.bid = 103 );
118
119
120 -- Query 7 optimized (STATISTICS)
121
122 explain analyze select s.sname
123   from sailors s inner join (select distinct r.sid from reserves r where r.bid =103 ) as r1 on s.sid =r1.sid
124
125
126
127 -- ====== Query 8 ======
128
129
130 -- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673
131
132 select count(s.sid)
133   from sailors s
134   where s.sid in ( select r.sid
135     from reserves r
136    where r. bid in (select b.bid
137      from boat b
138     where b.color = 'red'));
139
```

Below the SQL tab is a "Data output" tab showing the result of the third query:

count	bigint
1	673

At the bottom of the pgAdmin window, there is a message bar: "Successfully run. Total query runtime: 51 msec. 582 rows affected. Ln 129, Col 1".

Report

1. given query without an index :

pgAdmin 4

May 17 12:08 AM

schema3/postgres@postgres*

```

15
16 select count(sid)
from sailors;
17
18 select count(*)
from boat;
19
20
21
22 select count(*)
from reserves;
23
24
25
26 delete from Reserves;
27
28 delete from sailors;
29
30 delete from Boat;
31
32
33 --- Sailors
34 -- see what indexes are created for that table
35 select *
36 from pg_indexes
37 where tablename = 'sailors' or tablename='reserves' or tablename='boat';
38
39 -- see constraint names
40 SELECT con.*
41   FROM pg_catalog.pg_constraint con
42     INNER JOIN pg_catalog.pg_class rel

```

Data output

schemaname	tablename	indexname	tablespace	indexdef
name	name	name	name	text

Total rows: 0 of 0 Query complete 00:00:00.17

Successfully run. Total query runtime: 51 msec. 582 rows affected.

Ln 35, Col 1

pgAdmin 4

Jun 9 6:05 PM

schema3/postgres@postgres*

Explain Analyze [Shift (F7)]

```

208
209
210
211 DROP INDEX IF EXISTS b_sailorsSID cascade;
212 DROP INDEX IF EXISTS b_reservesSID cascade;
213 DROP INDEX IF EXISTS b_reservesBID cascade;
214 DROP INDEX IF EXISTS R_reservesBID cascade;
215
216
217 set enable_hashagg = off;
218 set enable_hashjoin = off;
219 set enable_seqscan = on;
220 -- Query 8 (STATISTICS)
221
222
223
224 explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
225
226
227
228
229
230
231
232 -- Query 8 optimized (STATISTICS)
233
234 -- Query 8 view table of reserves sids that have red boats
235 create MATERIALIZED VIEW query_8
236 as
237 select r1.sid
238   from (select r.sid
239         from reserves r
240         where exists (select * from boat b where r.bid=b.bid and color='red' )) as r1
241
242 explain analyze select s.sname
243 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
244
245
246
247
248 -- ****

```

Graphical Analysis Statistics

```

graph TD
    sailors[sailors] --> Sort1[Sort]
    Sort1 --> MergeJoin1[Merge Join]
    reserves[reserves] --> Sort2[Sort]
    Sort2 --> MergeJoin2[Merge Semi Join]
    boat[boat] --> Sort3[Sort]
    Sort3 --> MergeJoin2
    MergeJoin1 --> MergeJoin2

```

Total rows: 1 of 1 Query complete 00:00:00.054

Successfully run. Total query runtime: 54 msec. 1 rows affected.

Successfully run. Total query runtime: 137 msec. 4 rows affected.

Ln 224, Col 17

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree view shows a connection to 'schema3/postgres@postgres'. In the center-left, the 'Query' tab is selected, displaying a series of SQL commands. On the right, the 'Data output' tab shows the results of the last two queries, both of which were successfully run. The 'Query Plan' tab at the top right displays the execution plan for the current query, which includes a merge semi join and multiple sort operations.

```

208
209
210
211 DROP INDEX IF EXISTS b_sailorsSID cascade;
212 DROP INDEX IF EXISTS b_reservesSID cascade;
213 DROP INDEX IF EXISTS b_reservesBID cascade;
214 DROP INDEX IF EXISTS R_reservesBID cascade;
215
216
217 set enable_hashagg = off;
218 set enable_hashjoin = off;
219 set enable_seqscan = on;
220 -- Query 8 (STATISTICS)
221
222
223
224 explain analyze select s.sname
225 from sailors s
226 where s.sid in ( select r.sid
227 from reserves r
228 where r.bid in (select b.bid
229 from boat b
230 where b.color = 'red'));
231
232 -- Query 8 optimized (STATISTICS)
233
234 -- Query 8 view table of reserves sids that have red boats
235 create MATERIALIZED VIEW query_8
236 as
237 select r1.sid
238 from (select r.sid
239 from reserves r
240 where exists (select * from boat b where r.bid=b.bid and color='red')) as r1
241
242 explain analyze select s.sname
243 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
244
245
246
247
248 -- ====== Query 9 ======
Total rows: 23 of 23   Query complete 00:00:00.082

```

✓ Successfully run. Total query runtime: 82 msec. 23 rows affected
✓ Successfully run. Total query runtime: 137 msec. 4 rows affected
Ln 217, Col 1

Explanation :

- Metrics :

Execution Time : 17.183 ms Total Expected Cost : 5664.74

- Here I removed all indices and the performance is not that bad due to Query optimizer and the size of the data is in thousands .

2. given query with B+ trees indices only :

pgAdmin 4

Jun 9 6:12 PM • pgAdmin 4

Servers Local PostgreSQL

Query History

```

1 p_92 select count(s.sname)
2   from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
3 
4 p_93
5 p_94
6 p_95
7 p_96
8 p_97
9 p_98
10 p_99
11 p_00 CREATE INDEX b_sailorsSID ON sailors USING btree(sid );
12 p_01 CREATE INDEX b_reservesSID ON reserves USING btree(sid );
13 p_02 CREATE INDEX b_reservesBID ON reserves USING btree(bid);
14 p_03 CREATE INDEX b_boat ON boat USING btree(bid,color );
15 p_04 CREATE INDEX b_query_8 ON query_8 USING btree(sid );
16 p_05
17 p_06
18 p_07
19 p_08 select *
20   from pg_indexes
21  where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_8';
22 
23 p_09
24 p_10
25 p_11
26 p_12
27 p_13
28 p_14
29 p_15
30 p_16
31 p_17
32 p_18
33 p_19
34 p_20
35 p_21
36 p_22
37 p_23
38 p_24
39 p_25
40 p_26
41 p_27
42 p_28
43 p_29
44 p_30
45 p_31
46 p_32
47 p_33
48 p_34
49 p_35
50 p_36
51 p_37
52 p_38
53 p_39
54 p_40
55 p_41
56 p_42
57 p_43
58 p_44
59 p_45
60 p_46
61 p_47
62 p_48
63 p_49
64 p_50
65 p_51
66 p_52
67 p_53
68 p_54
69 p_55
70 p_56
71 p_57
72 p_58
73 p_59
74 p_60
75 p_61
76 p_62
77 p_63
78 p_64
79 p_65
80 p_66
81 p_67
82 p_68
83 p_69
84 p_70
85 p_71
86 p_72
87 p_73
88 p_74
89 p_75
90 p_76
91 p_77
92 p_78
93 p_79
94 p_79
95 p_79
96 p_79
97 p_79
98 p_79
99 p_79
100 p_79
101 p_79
102 p_79
103 p_79
104 p_79
105 p_79
106 p_79
107 p_79
108 p_79
109 p_79
110 p_79
111 p_79
112 p_79
113 p_79
114 p_79
115 p_79
116 p_79
117 p_79
118 p_79
119 p_79
120 p_79
121 p_79
122 p_79
123 p_79
124 p_79
125 p_79
126 p_79
127 p_79
128 p_79
129 p_79
130 p_79
131 p_79
132 p_79
133 p_79
134 p_79
135 p_79
136 p_79
137 p_79
138 p_79
139 p_79
140 p_79
141 p_79
142 p_79
143 p_79
144 p_79
145 p_79
146 p_79
147 p_79
148 p_79
149 p_79
150 p_79
151 p_79
152 p_79
153 p_79
154 p_79
155 p_79
156 p_79
157 p_79
158 p_79
159 p_79
160 p_79
161 p_79
162 p_79
163 p_79
164 p_79
165 p_79
166 p_79
167 p_79
168 p_79
169 p_79
170 p_79
171 p_79
172 p_79
173 p_79
174 p_79
175 p_79
176 p_79
177 p_79
178 p_79
179 p_79
180 p_79
181 p_79
182 p_79
183 p_79
184 p_79
185 p_79
186 p_79
187 p_79
188 p_79
189 p_79
190 p_79
191 p_79
192 p_79
193 p_79
194 p_79
195 p_79
196 p_79
197 p_79
198 p_79
199 p_79
200 p_79
201 p_79
202 p_79
203 p_79
204 p_79
205 p_79
206 p_79
207 p_79
208 p_79
209 p_79
210 p_79
211 p_79
212 p_79
213 p_79
214 p_79
215 p_79
216 p_79
217 p_79
218 p_79
219 p_79
220 p_79
221 p_79
222 p_79
223 p_79
224 p_79
225 p_79
226 p_79
227 p_79
228 p_79
229 p_79
230 p_79
231 p_79
232 p_79
233 p_79
234 p_79
235 p_79
236 p_79
237 p_79
238 p_79
239 p_79
240 p_79
241 p_79
242 p_79
243 p_79
244 p_79
245 p_79

```

Data output Messages Explain Notifications

schemaname	tablename	Indexname	Tablespace	Indexdef
public	sailors	b_sailorsSID	[null]	CREATE INDEX b_sailorsSID ON public.sailors USING btree (sid)
public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesSID ON public.reserves USING btree (sid)
public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesBID ON public.reserves USING btree (bid)
public	boat	b_boat	[null]	CREATE INDEX b_boat ON public.boat USING btree (bid, color)
public	query_8	b_query_8	[null]	CREATE INDEX b_query_8 ON public.query_8 USING btree (sid)

Successfully run. Total query runtime: 137 msec. 4 rows affected.

pgAdmin 4

Jun 9 6:14 PM • pgAdmin 4

Servers Local PostgreSQL

Query History

```

1 p_285
2 p_286
3 p_287
4 p_288
5 p_289
6 p_290
7 p_291
8 p_292
9 p_293
10 p_294
11 p_295
12 p_296
13 p_297
14 p_298
15 p_299
16 p_300
17 p_301
18 p_302
19 p_303
20 p_304
21 p_305
22 p_306
23 p_307
24 p_308
25 p_309
26 p_310
27 p_311
28 p_312
29 p_313
30 p_314
31 p_315
32 p_316
33 p_317
34 p_318
35 p_319
36 p_320
37 p_321
38 p_322
39 p_323
40 p_324
41 p_325
42 p_326
43 p_327
44 p_328
45 p_329
46 p_330
47 p_331
48 p_332
49 p_333
50 p_334
51 p_335
52 p_336
53 p_337
54 p_338
55 p_339
56 p_340
57 p_341
58 p_342
59 p_343
60 p_344
61 p_345
62 p_346
63 p_347
64 p_348
65 p_349
66 p_350
67 p_351
68 p_352
69 p_353
70 p_354
71 p_355
72 p_356
73 p_357
74 p_358
75 p_359
76 p_360
77 p_361
78 p_362
79 p_363
80 p_364
81 p_365
82 p_366
83 p_367
84 p_368
85 p_369
86 p_370
87 p_371
88 p_372
89 p_373
90 p_374
91 p_375
92 p_376
93 p_377
94 p_378
95 p_379
96 p_380
97 p_381
98 p_382
99 p_383
100 p_384
101 p_385
102 p_386
103 p_387
104 p_388
105 p_389
106 p_390
107 p_391
108 p_392
109 p_393
110 p_394
111 p_395
112 p_396
113 p_397
114 p_398
115 p_399
116 p_400
117 p_401
118 p_402
119 p_403
120 p_404
121 p_405
122 p_406
123 p_407
124 p_408
125 p_409
126 p_410
127 p_411
128 p_412
129 p_413
130 p_414
131 p_415
132 p_416
133 p_417
134 p_418
135 p_419
136 p_420
137 p_421
138 p_422
139 p_423
140 p_424
141 p_425
142 p_426
143 p_427
144 p_428
145 p_429
146 p_430
147 p_431
148 p_432
149 p_433
150 p_434
151 p_435
152 p_436
153 p_437
154 p_438
155 p_439
156 p_440
157 p_441
158 p_442
159 p_443
160 p_444
161 p_445
162 p_446
163 p_447
164 p_448
165 p_449
166 p_450
167 p_451
168 p_452
169 p_453
170 p_454
171 p_455
172 p_456
173 p_457
174 p_458
175 p_459
176 p_460
177 p_461
178 p_462
179 p_463
180 p_464
181 p_465
182 p_466
183 p_467
184 p_468
185 p_469
186 p_470
187 p_471
188 p_472
189 p_473
190 p_474
191 p_475
192 p_476
193 p_477
194 p_478
195 p_479
196 p_480
197 p_481
198 p_482
199 p_483
200 p_484
201 p_485
202 p_486
203 p_487
204 p_488
205 p_489
206 p_490
207 p_491
208 p_492
209 p_493
210 p_494
211 p_495
212 p_496
213 p_497
214 p_498
215 p_499
216 p_500
217 p_501
218 p_502
219 p_503
220 p_504
221 p_505
222 p_506
223 p_507
224 p_508
225 p_509
226 p_510
227 p_511
228 p_512
229 p_513
230 p_514
231 p_515
232 p_516
233 p_517
234 p_518
235 p_519
236 p_520
237 p_521
238 p_522
239 p_523
240 p_524
241 p_525
242 p_526
243 p_527
244 p_528
245 p_529
246 p_530
247 p_531
248 p_532
249 p_533
250 p_534
251 p_535
252 p_536
253 p_537
254 p_538
255 p_539
256 p_540
257 p_541
258 p_542
259 p_543
260 p_544
261 p_545
262 p_546
263 p_547
264 p_548
265 p_549
266 p_550
267 p_551
268 p_552
269 p_553
270 p_554
271 p_555
272 p_556
273 p_557
274 p_558
275 p_559
276 p_560
277 p_561
278 p_562
279 p_563
280 p_564
281 p_565
282 p_566
283 p_567
284 p_568
285 p_569
286 p_570
287 p_571
288 p_572
289 p_573
290 p_574
291 p_575
292 p_576
293 p_577
294 p_578
295 p_579
296 p_580
297 p_581
298 p_582
299 p_583
300 p_584
301 p_585
302 p_586
303 p_587
304 p_588
305 p_589
306 p_590
307 p_591
308 p_592
309 p_593
310 p_594
311 p_595
312 p_596
313 p_597
314 p_598
315 p_599
316 p_590
317 p_591
318 p_592
319 p_593
320 p_594
321 p_595
322 p_596
323 p_597
324 p_598
325 p_599
326 p_590
327 p_591
328 p_592
329 p_593
330 p_594
331 p_595
332 p_596
333 p_597
334 p_598
335 p_599
336 p_590
337 p_591
338 p_592
339 p_593
340 p_594
341 p_595
342 p_596
343 p_597
344 p_598
345 p_599
346 p_590
347 p_591
348 p_592
349 p_593
350 p_594
351 p_595
352 p_596
353 p_597
354 p_598
355 p_599
356 p_590
357 p_591
358 p_592
359 p_593
360 p_594
361 p_595
362 p_596
363 p_597
364 p_598
365 p_599
366 p_590
367 p_591
368 p_592
369 p_593
370 p_594
371 p_595
372 p_596
373 p_597
374 p_598
375 p_599
376 p_590
377 p_591
378 p_592
379 p_593
380 p_594
381 p_595
382 p_596
383 p_597
384 p_598
385 p_599
386 p_590
387 p_591
388 p_592
389 p_593
390 p_594
391 p_595
392 p_596
393 p_597
394 p_598
395 p_599
396 p_590
397 p_591
398 p_592
399 p_593
400 p_594
401 p_595
402 p_596
403 p_597
404 p_598
405 p_599
406 p_590
407 p_591
408 p_592
409 p_593
410 p_594
411 p_595
412 p_596
413 p_597
414 p_598
415 p_599
416 p_590
417 p_591
418 p_592
419 p_593
420 p_594
421 p_595
422 p_596
423 p_597
424 p_598
425 p_599
426 p_590
427 p_591
428 p_592
429 p_593
430 p_594
431 p_595
432 p_596
433 p_597
434 p_598
435 p_599
436 p_590
437 p_591
438 p_592
439 p_593
440 p_594
441 p_595
442 p_596
443 p_597
444 p_598
445 p_599
446 p_590
447 p_591
448 p_592
449 p_593
450 p_594
451 p_595
452 p_596
453 p_597
454 p_598
455 p_599
456 p_590
457 p_591
458 p_592
459 p_593
460 p_594
461 p_595
462 p_596
463 p_597
464 p_598
465 p_599
466 p_590
467 p_591
468 p_592
469 p_593
470 p_594
471 p_595
472 p_596
473 p_597
474 p_598
475 p_599
476 p_590
477 p_591
478 p_592
479 p_593
480 p_594
481 p_595
482 p_596
483 p_597
484 p_598
485 p_599
486 p_590
487 p_591
488 p_592
489 p_593
490 p_594
491 p_595
492 p_596
493 p_597
494 p_598
495 p_599
496 p_590
497 p_591
498 p_592
499 p_593
500 p_594
501 p_595
502 p_596
503 p_597
504 p_598
505 p_599
506 p_590
507 p_591
508 p_592
509 p_593
510 p_594
511 p_595
512 p_596
513 p_597
514 p_598
515 p_599
516 p_590
517 p_591
518 p_592
519 p_593
520 p_594
521 p_595
522 p_596
523 p_597
524 p_598
525 p_599
526 p_590
527 p_591
528 p_592
529 p_593
530 p_594
531 p_595
532 p_596
533 p_597
534 p_598
535 p_599
536 p_590
537 p_591
538 p_592
539 p_593
540 p_594
541 p_595
542 p_596
543 p_597
544 p_598
545 p_599
546 p_590
547 p_591
548 p_592
549 p_593
550 p_594
551 p_595
552 p_596
553 p_597
554 p_598
555 p_599
556 p_590
557 p_591
558 p_592
559 p_593
560 p_594
561 p_595
562 p_596
563 p_597
564 p_598
565 p_599
566 p_590
567 p_591
568 p_592
569 p_593
570 p_594
571 p_595
572 p_596
573 p_597
574 p_598
575 p_599
576 p_590
577 p_591
578 p_592
579 p_593
580 p_594
581 p_595
582 p_596
583 p_597
584 p_598
585 p_599
586 p_590
587 p_591
588 p_592
589 p_593
590 p_594
591 p_595
592 p_596
593 p_597
594 p_598
595 p_599
596 p_590
597 p_591
598 p_592
599 p_593
600 p_594
601 p_595
602 p_596
603 p_597
604 p_598
605 p_599
606 p_590
607 p_591
608 p_592
609 p_593
610 p_594
611 p_595
612 p_596
613 p_597
614 p_598
615 p_599
616 p_590
617 p_591
618 p_592
619 p_593
620 p_594
621 p_595
622 p_596
623 p_597
624 p_598
625 p_599
626 p_590
627 p_591
628 p_592
629 p_593
630 p_594
631 p_595
632 p_596
633 p_597
634 p_598
635 p_599
636 p_590
637 p_591
638 p_592
639 p_593
640 p_594
641 p_595
642 p_596
643 p_597
644 p_598
645 p_599
646 p_590
647 p_591
648 p_592
649 p_593
650 p_594
651 p_595
652 p_596
653 p_597
654 p_598
655 p_599
656 p_590
657 p_591
658 p_592
659 p_593
660 p_594
661 p_595
662 p_596
663 p_597
664 p_598
665 p_599
666 p_590
667 p_591
668 p_592
669 p_593
670 p_594
671 p_595
672 p_596
673 p_597
674 p_598
675 p_599
676 p_590
677 p_591
678 p_592
679 p_593
680 p_594
681 p_595
682 p_596
683 p_597
684 p_598
685 p_599
686 p_590
687 p_591
688 p_592
689 p_593
690 p_594
691 p_595
692 p_596
693 p_597
694 p_598
695 p_599
696 p_590
697 p_591
698 p_592
699 p_593
700 p_594
701 p_595
702 p_596
703 p_597
704 p_598
705 p_599
706 p_590
707 p_591
708 p_592
709 p_593
710 p_594
711 p_595
712 p_596
713 p_597
714 p_598
715 p_599
716 p_590
717 p_591
718 p_592
719 p_593
720 p_594
721 p_595
722 p_596
723 p_597
724 p_598
725 p_599
726 p_590
727 p_591
728 p_592
729 p_593
730 p_594
731 p_595
732 p_596
733 p_597
734 p_598
735 p_599
736 p_590
737 p_591
738 p_592
739 p_593
740 p_594
741 p_595
742 p_596
743 p_597
744 p_598
745 p_599
746 p_590
747 p_591
748 p_592
749 p_593
750 p_594
751 p_595
752 p_596
753 p_597
754 p_598
755 p_599
756 p_590
757 p_591
758 p_592
759 p_593
760 p_594
761 p_595
762 p_596
763 p_597
764 p_598
765 p_599
766 p_590
767 p_591
768 p_592
769 p_593
770 p_594
771 p_595
772 p_596
773 p_597
774 p_598
775 p_599
776 p_590
777 p_591
778 p_592
779 p_593
780 p_594
781 p_595
782 p_596
783 p_597
784 p_598
785 p_599
786 p_590
787 p_591
788 p_592
789 p_593
790 p_594
791 p_595
792 p_596
793 p_597
794 p_598
795 p_599
796 p_590
797 p_591
798 p_592
799 p_593
800 p_594
801 p_595
802 p_596
803 p_597
804 p_598
805 p_599
806 p_590
807 p_591
808 p_592
809 p_593
810 p_594
811 p_595
812 p_596
813 p_597
814 p_598
815 p_599
816 p_590
817 p_591
818 p_592
819 p_593
820 p_594
821 p_595
822 p_596
823 p_597
824 p_598
825 p_599
826 p_590
827 p_591
828 p_592
829 p_593
830 p_594
831 p_595
832 p_596
833 p_597
834 p_598
835 p_599
836 p_590
837 p_591
838 p_592
839 p_593
840 p_594
841 p_595
842 p_596
843 p_597
844 p_598
845 p_599
846 p_590
847 p_591
848 p_592
849 p_593
850 p_594
851 p_595
852 p_596
853 p_597
854 p_598
855 p_599
856 p_590
857 p_591
858 p_592
859 p_593
860 p_594
861 p_595
862 p_596
863 p_597
864 p_598
865 p_599
866 p_590
867 p_591
868 p_592
869 p_593
870 p_594
871 p_595
872 p_596
873 p_597
874 p_598
875 p_599
876 p_590
877 p_591
878 p_592
879 p_593
880 p_594
881 p_595
882 p_596
883 p_597
884 p_598
885 p_599
886 p_590
887 p_591
888 p_592
889 p_593
890 p_594
891 p_595
892 p_596
893 p_597
894 p_598
895 p_599
896 p_590
897 p_591
898 p_592
899 p_593
900 p_594
901 p_595
902 p_596
903 p_597
904 p_598
905 p_599
906 p_590
907 p_591
908 p_592
909 p_593
910 p_594
911 p_595
912 p_596
913 p_597
914 p_598
915 p_599
916 p_590
917 p_591
918 p_592
919 p_593
920 p_594
921 p_595
922 p_596
923 p_597
924 p_598
925 p_599
926 p_590
927 p_591
928 p_592
929 p_593
930 p_594
931 p_595
932 p_596
933 p_597
934 p_598
935 p_599
936 p_590
937 p_591
938 p_592
939 p_593
940 p_594
941 p_595
942 p_596
943 p_597
944 p_598
945 p_599
946 p_590
947 p_591
948 p_592
949 p_593
950 p_594
951 p_595
952 p_596
953 p_597
954 p_598
955 p_599
956 p_590
957 p_591
958 p_592
959 p_593
960 p_594
961 p_595
962 p_596
963 p_597
964 p_598
965 p_599
966 p_590
967 p_591
968 p_592
969 p_593
970 p_594
971 p_595
972 p_596
973 p_597
974 p_598

```

```

query_8.sql
-- Query 8 (STATISTICS)
-- Query 8 view table of reserves sids that have red boats
create MATERIALIZED VIEW query_8
as
select r1.sid
from (select r.sid
      from reserves r
      where exists (select * from boat b where b.bid=r.bid and color='red' ) ) as r1 ;
explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
                  from reserves r
                  where exists (select * from boat b where b.bid=r.bid and color='red' ) ) ;
-- Query 8 optimized (STATISTICS)
-- Query 8 view table of reserves sids that have red boats
create MATERIALIZED VIEW query_8
as
select r1.sid
from (select r.sid
      from reserves r
      where exists (select * from boat b where b.bid=r.bid and color='red' ) ) as r1 ;
explain analyze select s.sname
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
Total rows: 18 of 18   Query complete 00:00:00.134

```

Execution Plan:

```

QUERY PLAN
text
1  Merge Semi Join (cost=1994.16..2779.35 rows=4982 width=21) (actual time=1.514..2.373 rows=673 loops=1)
  Merge Cond: (s.sid = r.sid)
  3 -> Index Scan using b_sailorssid on sailors s (cost=0.29..663.29 rows=19000 width=25) (actual time=0.005..0.447 rows=2999 loops=1)
  4 -> Sort (cost=1993.87..2006.32 rows=4982 width=4) (actual time=1.506..1.562 rows=1136 loops=1)
  5  Sort Key: r.sid
  6  Sort Method: quicksort Memory: 102kB
  7 -> Nested Loop (cost=81.45..1687.91 rows=4982 width=4) (actual time=0.367..1.313 rows=1136 loops=1)
  8 -> Unique (cost=81.16..83.22 rows=427 width=4) (actual time=0.361..0.447 rows=427 loops=1)
  9 -> Sort (cost=81.16..82.22 rows=427 width=4) (actual time=0.361..0.388 rows=427 loops=1)
  10 Sort Key: b.bid
  11 Sort Method: quicksort Memory: 45kB
  12 -> Seq Scan on boat b (cost=0.00..62.50 rows=427 width=4) (actual time=0.008..0.319 rows=427 loops=1)
  13 Filter: (color = 'red'::bpchar)
  14 Rows Removed by Filter: 2573
  15 -> Index Scan using b_reservesbid on reserves r (cost=0.29..3.48 rows=28 width=8) (actual time=0.001..0.002 rows=3 loops=427)
  16 Index Cond: (bid = b.bid)
  17 Planning Time: 0.324 ms
  18 Execution Time: 2.421 ms

```

Successfully run. Total query runtime: 137 msec. 4 rows affected
Ln 219, Col 1

Explanation :

- Metrics :

Execution Time : 2.421 ms Total Expected Cost : 2779.35

- The B-tree helped in the performance it decreased the Execution Time and Expected Cost .
- The Query Planner used the B-tree index because B-Tree is O(Log n) performance for the attributes before the IN Operator cause the leaves are sorted and Query Planner sorted the subquery result to match to Merge Join on condition (b.bid=r.bid).
- The Query Planner used the B-tree index because B-Tree is O(Log n) performance for the attributes before the IN Operator cause the leaves are sorted and Query Planner sorted the subquery result to match to Merge Join on condition (s.sid=r.sid).

3. given query with hash indices only :

pgAdmin 4

Servers: schema3/postgres

Databases: Local | postgres

Data | Query History

```

> p 196
> s 197
> s 198
> s 199
> s 200 CREATE INDEX b_sailorsSID ON sailors USING hash(sid );
> s 201 CREATE INDEX b_reservesID ON reserves USING hash(sid );
> s 202 CREATE INDEX b_reservesBID ON reserves USING hash(bid );
> s 203 CREATE INDEX b_boat1 ON boat USING hash(bid );
> s 204 CREATE INDEX b_boat2 ON boat USING hash(color );
> s 205
> s 206 CREATE INDEX b_query_8 ON query_8 USING hash(sid );
> s 207
> s 208
> s 209 select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_8';
> s 210
> s 211
> s 212
> Log 213
> Tab 214
> postgres 215 DROP INDEX IF EXISTS b_sailorsSID cascade;
216 DROP INDEX IF EXISTS b_reservesID cascade;
217 DROP INDEX IF EXISTS b_reservesBID cascade;
218 DROP INDEX IF EXISTS b_boat1 cascade;
219 DROP INDEX IF EXISTS b_boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = on;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
```

Total rows: 6 of 6 Query complete 00:00:00.169

Data output | Messages | Explain | Notifications

schema name	table name	index name	tablespace	Indexdef
public	sailors	b_sailorsSID	[null]	CREATE INDEX b_sailorsSID ON public.sailors USING hash (sid)
public	reserves	b_reservesID	[null]	CREATE INDEX b_reservesID ON public.reserves USING hash (sid)
public	reserves	b_reservesBID	[null]	CREATE INDEX b_reservesBID ON public.reserves USING hash (bid)
public	boat	b_boat1	[null]	CREATE INDEX b_boat1 ON public.boat USING hash (bid)
public	boat	b_boat2	[null]	CREATE INDEX b_boat2 ON public.boat USING hash (color)
public	query_8	b_query_8	[null]	CREATE INDEX b_query_8 ON public.query_8 USING hash (sid)

Successfully run. Total query runtime: 137 msec. 4 rows affected.

Ln 198, Col 1

pgAdmin 4

Servers: schema3/postgres

Databases: Local | postgres

Data | Query History

```

> p 216 DROP INDEX IF EXISTS b_reservesID cascade;
> s 217 CREATE INDEX b_reservesBID ON reserves USING hash(bid );
> s 218 CREATE INDEX b_boat1 ON boat USING hash(bid );
> s 219 CREATE INDEX b_boat2 ON boat USING hash(color );
> s 220
> s 221 DROP INDEX IF EXISTS b_query_8 cascade;
> s 222
> s 223
> s 224 set enable_hashagg = off;
> s 225 set enable_hashjoin = off;
> s 226 set enable_seqscan = on;
> s 227 -- Query 8 (STATISTICS)
> s 228
> s 229
> s 230
> s 231 explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
```

Total rows: 1 of 1 Query complete 00:00:00.105

Data output | Messages | Explain | Notifications

Graphical | Analysis | Statistics

```

graph LR
    reserves[reserves] -->|Nested Loop Semi Join| b_boat1[b_boat1]
    b_boat1 -->|Sort| Sort[Sort]
    Sort -->|Unique| Unique[Unique]
    Unique -->|Nested Loop Inne Join| b_sailorsid[b_sailorsid]

```

Successfully run. Total query runtime: 137 msec. 4 rows affected.

Ln 231, Col 17

```

216 DROP INDEX IF EXISTS b_reserve_idx cascade;
217 DROP INDEX IF EXISTS b_reservesID cascade;
218 DROP INDEX IF EXISTS b_boat cascade;
219 DROP INDEX IF EXISTS b_boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = on;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
232 from sailors s
233 where s.sid in ( select r.sid
234 from reserves r
235 where r.bid in (select b.bid
236 from boat b
237 where b.color = 'red'));
238
239 -- Query 8 optimized (STATISTICS)
240
241 -- Query 8 view table of reserves sids that have red boats
242 create MATERIALIZED VIEW query_8
243 as
244 select r1.sid
245 from (select r.sid
246 from reserves r
247 where exists (select * from boat b where r.bid=b.bid and color='red' )) as r1 ;
248
249 explain analyze select s.sname
250 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
251
252
253
254
255
256

```

SuccessFully run. Total query runtime: 106 msec. 15 rows affected
Ln 229, Col 1

SuccessFully run. Total query runtime: 137 msec. 4 rows affected
Ln 229, Col 1

Explanation :

- Metrics :

Execution Time : 38.244 ms Total Expected Cost : 2168.36

- The Hash helped in the performance it decreased the Execution Time and Expected Cost .
- The Query Planner used the Hash index because Hash is O(1) performance with Exact Values and considered the best for exact values queries (BEST OF THEM ALL).
- Here it showed the improvement due to for condition of Joining in the Query the Nested Loop Semi Join using index scan using Hash based algorithm on the condition (b.bid=r.bid) which approximatly maded to be O(n) performance .
- Here it showed the improvement due to for condition of Joining in the Query the Nested Loop Semi Join using index scan using Hash based algorithm on the condition (s.sid=r.sid) which approximatly maded to be O(n) performance .
- Here it showed the improvement due to for condition of (color = "red") with performance of O(1) .

- given query with BRIN indices only :

pgAdmin 4

Jun 9 6:22 PM • pgAdmin 4

Servers schema3/postgres Local PostgreSQL Data Query History

```

193 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
194
195
196
197
198
199
200 CREATE INDEX b_sailorsSID ON sailors USING brin(sid );
201 CREATE INDEX b_reservesSID ON reserves USING brin(sid );
202 CREATE INDEX b_reservesBID ON reserves USING brin(bid );
203 CREATE INDEX b_boat1 ON boat USING brin(bid );
204 CREATE INDEX b_boat2 ON boat USING brin(color );
205
206
207
208
209 select *
210 from pg_indexes
211 where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_8';
212
213
214
215 DROP INDEX IF EXISTS b_sailorsSID cascade;
216 DROP INDEX IF EXISTS b_reservesSID cascade;
217 DROP INDEX IF EXISTS b_reservesBID cascade;
218 DROP INDEX IF EXISTS b_boat1 cascade;
219 DROP INDEX IF EXISTS b_boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = on;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
232 from sailors s
233 where s.sid in ( select r.sid
234
Total rows: 6 of 6 Query complete 00:00:00.067

```

Data output Messages Explain Notifications

schemaname	tablename	indexname	tablespace	Indexdef
1 public	sailors	b_sailorsSID	[null]	CREATE INDEX b_sailorsSID ON public.sailors USING brin(sid)
2 public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesSID ON public.reserves USING brin(sid)
3 public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesBID ON public.reserves USING brin(bid)
4 public	boat	b_boat1	[null]	CREATE INDEX b_boat1 ON public.boat USING brin(bid)
5 public	boat	b_boat2	[null]	CREATE INDEX b_boat2 ON public.boat USING brin(color)
6 public	query_8	b_query_8	[null]	CREATE INDEX b_query_8 ON public.query_8 USING brin(sid)

Successfully run. Total query runtime: 137 msec. 4 rows affected. Ln 208, Col 1

pgAdmin 4

Jun 9 6:24 PM • pgAdmin 4

Servers schema3/postgres Local PostgreSQL Data Query History

```

209 select *
210 from pg_indexes
211 where tablename = 'sailors' or tablename='reserves'
212
213
214
215 DROP INDEX IF EXISTS b_sailorsSID cascade;
216 DROP INDEX IF EXISTS b_reservesSID cascade;
217 DROP INDEX IF EXISTS b_reservesBID cascade;
218 DROP INDEX IF EXISTS b_boat1 cascade;
219 DROP INDEX IF EXISTS b_boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = off;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
232 from sailors s
233 where s.sid in ( select r.sid
234 from reserves r
235 where r.bid in (select b.bid
236 from boat b
237 where b.color = 'red'));
238
239 -- Query 8 optimized (STATISTICS)
240
241 -- Query 8 view table of reserves sids that have
242 create MATERIALIZED VIEW query_8
243 as
244 select r1.sid
245 from (select r.sid
246 from reserves r
247 where exists (select * from boat b where
248
Total rows: 1 of 1 Query complete 00:00:01.895

```

Explain Analyze

Successfully run. Total query runtime: 137 msec. 4 rows affected. Ln 231, Col 17

```

Jun 9 6:23 PM • pgAdmin 4
Activities pgAdmin 4 ▾ Jun 9 6:23 PM • pgAdmin 4
File Object Tools Help
Properties SQL Dependents Statistics Dashboard schema3/postgres@postgres*
Servers Local Postgr
Data Query History
> P 209
> S 210
> S 211
where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='que
> S 212
> S 213
> S 214
> S 215
DROP INDEX IF EXISTS b_sailorsSID cascade;
> S 216
DROP INDEX IF EXISTS b_reservesID cascade;
> S 217
DROP INDEX IF EXISTS b_reservesSID cascade;
> S 218
DROP INDEX IF EXISTS b_boat1 cascade;
> S 219
DROP INDEX IF EXISTS b_boat2 cascade;
> S 220
> S 221
DROP INDEX IF EXISTS b_query_8 cascade;
> S 222
> S 223
> S 224
set enable_hashagg = off;
set enable_hashjoin = off;
set enable_seqscan = off;
-- Query 8 (STATISTICS)
> S 225
> S 226
> S 227
-- Query 8 (STATISTICS)
> S 228
> S 229
> S 230
explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
-- Query 8 optimized (STATISTICS)
> S 231
-- Query 8 view table of reserves sids that have red boats
create MATERIALIZED VIEW query_8
as
select r1.sid
from (select r.sid
from reserves r
where exists (select * from boat b where r.bid=b.bid and color='red')) as r1;
explain analyze select s.sname
Total rows: 34 of 34 Query complete 0:00:01.887

```

Jun 9 6:23 PM • pgAdmin 4

```

Data output Messages Explain Notifications
QUERY PLAN
text
1 Nested Loop (cost=2602463.80..1126150803.66 rows=4982 width=21) (actual time=1064.570..1871.485 rows=673 loops=1)
   -> Unique (cost=237019.76..2377044.67 rows=4982 width=4) (actual time=1064.528..1065.172 rows=673 loops=1)
      -> Sort (cost=237019.76..2377032.22 rows=4982 width=4) (actual time=1064.527..1064.774 rows=1136 loops=1)
         Sort Key: r.sid
         Sort Method: quicksort Memory: 102k8
      -> Sort Key: r.sid
      -> Sort Method: quicksort Memory: 45kB
   -> Unique (cost=93.29..95.43 rows=427 width=4) (actual time=133.922..134.387 rows=427 loops=1)
      -> Sort (cost=93.29..94.36 rows=427 width=4) (actual time=133.921..134.070 rows=427 loops=1)
         Sort Key: b.bid
         Sort Method: quicksort Memory: 45kB
      -> Bitmap Heap Scan on boat b (cost=12.14..74.64 rows=427 width=4) (actual time=133.220..133.849 rows=427 loops=1)
         Index Cond: (color = 'red'::bpchar)
      -> Bitmap Heap Scan on reserves r (cost=5124.07..5565.57 rows=28 width=8) (actual time=0.089..2.173 rows=3 loops=427)
         Recheck Cond: (bid = b.bid)
         Rows Removed by Index Recheck: 23677
         -> Heap Blocks: lossy=25
         -> Bitmap Index Scan on b_boat2 (cost=0..0.12 rows=3000 width=0) (actual time=0.041..0.041 rows=250 loops=1)
         Index Cond: (color = 'red'::bpchar)
         -> Bitmap Heap Scan on boat b (cost=12.14..74.64 rows=427 width=4) (actual time=133.220..133.849 rows=427 loops=1)
            Index Cond: (color = 'red'::bpchar)
            Rows Removed by Index Recheck: 15359
            -> Heap Blocks: lossy=54656
            -> Bitmap Index Scan on b_reservesid (cost=0..0.5124.06 rows=35000 width=0) (actual time=0.012..0.012 rows=1280 loops=427)
            Index Cond: (bid = b.bid)
            -> Bitmap Heap Scan on sailors s (cost=225444.03..225566.78 rows=1 width=25) (actual time=0.041..1.195 rows=1 loops=673)
            Recheck Cond: (sid = r.sid)
            Rows Removed by Index Recheck: 23677
            -> Heap Blocks: lossy=86144
            -> Bitmap Index Scan on b_sailorssid (cost=0..0.225444.03 rows=9500 width=0) (actual time=0.011..0.011 rows=1280 loops=673)
            Index Cond: (sid = r.sid)
            Planning Time: 0.321 ms
Planning Time: 0.321 ms

```

Jun 9 6:23 PM • pgAdmin 4

```

Data output Messages Explain Notifications
QUERY PLAN
text
6 -> Nested Loop (cost=5217.37..2376713.81 rows=4982 width=4) (actual time=134.092..1064.060 rows=1136 loops=1)
7 -> Unique (cost=93.29..95.43 rows=427 width=4) (actual time=133.922..134.387 rows=427 loops=1)
8 -> Sort (cost=93.29..94.36 rows=427 width=4) (actual time=133.921..134.070 rows=427 loops=1)
9 Sort Key: b.bid
10 Sort Method: quicksort Memory: 45kB
11 -> Bitmap Heap Scan on boat b (cost=12.14..74.64 rows=427 width=4) (actual time=133.220..133.849 rows=427 loops=1)
12 Recheck Cond: (color = 'red'::bpchar)
13 Rows Removed by Index Recheck: 23677
14 Heap Blocks: lossy=25
15 -> Bitmap Index Scan on b_boat2 (cost=0..0.12 rows=3000 width=0) (actual time=0.041..0.041 rows=250 loops=1)
16 Index Cond: (color = 'red'::bpchar)
17 -> Bitmap Heap Scan on reserves r (cost=5124.07..5565.57 rows=28 width=8) (actual time=0.089..2.173 rows=3 loops=427)
18 Recheck Cond: (bid = b.bid)
19 Rows Removed by Index Recheck: 23677
20 Heap Blocks: lossy=54656
21 -> Bitmap Index Scan on b_reservesid (cost=0..0.5124.06 rows=35000 width=0) (actual time=0.012..0.012 rows=1280 loops=427)
22 Index Cond: (bid = b.bid)
23 -> Bitmap Heap Scan on sailors s (cost=225444.03..225566.78 rows=1 width=25) (actual time=0.041..1.195 rows=1 loops=673)
24 Recheck Cond: (sid = r.sid)
25 Rows Removed by Index Recheck: 15359
26 Heap Blocks: lossy=86144
27 -> Bitmap Index Scan on b_sailorssid (cost=0..0.225444.03 rows=9500 width=0) (actual time=0.011..0.011 rows=1280 loops=673)
28 Index Cond: (sid = r.sid)
29 Planning Time: 0.321 ms
30 JIT:
31 Functions: 16
32 Options:Inlining true, Optimization true, Expressions true, Deforming true
33 Timing: Generation 1.444 ms, Inlining 6.898 ms, Optimization 72.225 ms, Emission 53.792 ms, Total 134.359 ms
34 Execution Time: 1873.233 ms

```

Jun 9 6:23 PM • pgAdmin 4

```

Data output Messages Explain Notifications
QUERY PLAN
text
Total rows: 34 of 34 Query complete 0:00:01.887

```

Jun 9 6:23 PM • pgAdmin 4

```

Data output Messages Explain Notifications
QUERY PLAN
text
Total rows: 34 of 34 Query complete 0:00:01.887

```

Explanation :

- Metrics :

Execution Time : 18733.23 ms Total Expected Cost : 1126150803.66

- Here the BRIN was not used in the original Query Plan settings (Hashjoin and HashAgg are off) so I've made seqscan=off too.
- The Execution Time and Expected Cost became the (WORST OF THEM ALL) .

- This happened because the Query Optimizer didn't use it from the first place due to BRIN Usage here was not suitable so we have used it to simulate seqscan behaviour only we traversed it all and followed all its pointers so it is worst index to use in this case.
- Because there were no Aggregation used and was not low selectivity Query so it was not helpful to the performance.
- The plan performs a (Bit map index scan) on all indices and then to (Bit map heap scan) to select relevant rows .

5. given query with mixed indices (any mix of your choice) :

pgAdmin 4

Jun 9 6:43 PM ● pgAdmin 4

Servers schema3/postgres Local [postgres] Data Query History Execute/Refresh (F5)

Data output Messages Explain Notifications

schemaname	tablename	indexname	tablespace	indexdef
public	sailors	b_sailorsid	[null]	CREATE INDEX b_sailorsid ON public.sailors USING hash(sid)
public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesid ON public.reserves USING btree(sid)
public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesid ON public.reserves USING btree(bid)
public	boat	b_boat1	[null]	CREATE INDEX b_boat1 ON public.boat USING btree(bid)
public	boat	b_boat2	[null]	CREATE INDEX b_boat2 ON public.boat USING btree(color)
public	query_8	b_query_8	[null]	CREATE INDEX b_query_8 ON public.query_8 USING btree(sid)

```

198
CREATE INDEX b_sailorsSID ON sailors USING hash(sid );
199
CREATE INDEX b_reservesSID ON reserves USING btree(sid );
200
CREATE INDEX b_reservesBID ON reserves USING btree(bid );
201
CREATE INDEX b_boat1 ON boat USING btree(bid );
202
CREATE INDEX b_boat2 ON boat USING btree(color );
203
CREATE INDEX b_query_8 ON query_8 USING btree(sid );

204
select *
205
from pg_indexes
206
where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_8';

207
208
209
210
211
212
213
214
215
DROP INDEX IF EXISTS b_sailorsSID cascade;
216
DROP INDEX IF EXISTS b_reservesSID cascade;
217
DROP INDEX IF EXISTS b_reservesBID cascade;
218
DROP INDEX IF EXISTS b_boat1 cascade;
219
DROP INDEX IF EXISTS b_boat2 cascade;
220
221
DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224
set enable_hashagg = off;
225
set enable_hashjoin = off;
226
set enable_seqscan = on;
227
-- Query 8 (STATISTICS)
228
229
230
231
explain analyze select s.sname
232
from sailors s
233
where s.sid in ( select r.sid
234
from reserves r
235
where r.bid in (select b.bid
236
from boat b
237
where b.color = 'red'));
238
239
-- Query 8 optimized (STATISTICS)
240
241
-- Query 8 view table of reserves
242
create MATERIALIZED VIEW query_8
243
as
244
select r1.sid
245
from (select r.sid
246
from reserves r
247
where exists (select * from
248
boat b
249
explain analyze select s.sname
250
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid));

```

Total rows: 6 of 6 Query complete 00:00:00.055

Successfully run. Total query runtime: 137 msec. 4 rows affected.

Ln 208, Col 1

pgAdmin 4

Jun 9 6:44 PM ● pgAdmin 4

Servers schema3/postgres Local [postgres] Data Query History Explain Analyze Explain Statistics Notifications

Explain Analyze (Shift F7)

```

graph LR
    A[b_boat2] --> B[boat]
    B --> C[Sort]
    C --> D[Unique]
    D --> E[Nested Loop Inner Join]
    E --> F[Sort]
    F --> G[Unique]
    G --> H[Nested Loop Inner Join]
    H --> I[b_reservesbid]
    I --> J[b_sailorsid]

```

explain analyze select s.sname
from sailors s
where s.sid in (select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
-- Query 8 optimized (STATISTICS)
-- Query 8 view table of reserves
create MATERIALIZED VIEW query_8
as
select r1.sid
from (select r.sid
from reserves r
where exists (select * from
boat b
explain analyze select s.sname
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid));

Total rows: 1 of 1 Query complete 00:00:00.104

Successfully run. Total query runtime: 137 msec. 4 rows affected.

Ln 231, Col 16

The screenshot shows the pgAdmin 4 interface with the following details:

- Servers:** schema3/postgres
- Local:** postgres
- Data / Query History:** Shows a list of 210-250 numbered statements, mostly DROP INDEX commands.
- Execute/Refresh:** A button at the top of the history list.
- Explain Plan:** A detailed breakdown of the query execution plan, listing 21 steps from Nested Loop to Execution Time.
- Messages:** A status bar at the bottom right indicates "Successfully run. Total query runtime: 137 msec. 4 rows affected. Ln 230, Col 1".

```

210 from pg_indexes
211 where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query';
212
213
214
215 DROP INDEX IF EXISTS b_sailorsSID cascade;
216 DROP INDEX IF EXISTS b_reservesSID cascade;
217 DROP INDEX IF EXISTS b_reservesBID cascade;
218 DROP INDEX IF EXISTS b_boat cascade;
219 DROP INDEX IF EXISTS b_boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_haagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = on;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
232 from sailors s
233 where s.sid in ( select r.sid
234 from reserves r
235 where r.bid in (select b.bid
236 from boat b
237 where b.color = 'red'));
238
239 -- Query 8 optimized (STATISTICS)
240
241 -- Query 8 view table of reserves sids that have red boats
242 create MATERIALIZED VIEW query_8
243 as
244 select r1.sid
245 from (select r.sid
246 from reserves r
247 where exists (select * from boat b where r.bid=b.bid and color='red')) as r1;
248
249 explain analyze select s.sname
250 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
Total rows: 21 of 21   Query complete 00:00:00.93
  
```

Explanation :

- Metrics :

Execution Time : 2.486 ms Total Expected Cost : 2373.87

- The Query Planner used the Merge semi join by using both the ZigZag join and the Hash based algorithm together on the condition (s.sid =r.sid) which improved the Execution time and the expected cost way more better which made the join in $O(n \log n)$.
- The Query Planner used the Merge semi join by using both the ZigZag join and the Hash based algorithm together on the condition (r.bid = b.bid) which improved the Execution time and the expected cost way more better which made the join in $O(n \log n)$.
- And It used the Hash indexed scan on color ='red' with performance of $O(1)$.
- The Mix indices helped in the performance it decreased the Execution Time and Expected Cost .

Optimized Query

```

-- Query 8 view table of reserves sids that have red boats

create MATERIALIZED VIEW query_8
as
select r1.sid
from (select r.sid
      from reserves r
      where exists (select * from boat b where r.bid=b.bid and
color='red')) as r1;
  
```

```
explain analyze select s.sname  
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
```

Result Set

- 673 Rows

Report

1. given query without an index :

Activities pgAdmin 4 Jun 9 6:07 PM ● pgAdmin 4

Servers Local PostgreSQL schema3/postgres* Jun 9 6:07 PM ● pgAdmin 4

Databases Query History Explain Analyze Shift (F7)

```

210
211 set enable_hashagg = off;
212 set enable_hashjoin = off;
213 set enable_seqscan = on;
214
-- Query 8 (STATISTICS)
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
-- Query 8 optimized (STATISTICS)
233
234 -- Query 9 view table of reserves sids that have red boats
235 create MATERIALIZED VIEW query_8
236 as
237 select r1.sid
238 from (select r.sid
239         from reserves r
240       where exists (select * from boat b where r.bid=b.bid and color='red' )) as r1 ;
241
242 explain analyze select s.sname
243 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
244
245
246
247
248
-- ====== Query 9 ======
249
250 -- Find the names of sailors who have reserved both a red and a green boat.
251
252 -- Query 9 (COUNTING RESULT SET) , Number Of Rows = 177
253
254
255 select count(s.sname)
256 from sailors s, reserves r, boat b
257 where
Total rows: 1 of 1 Query complete 00:00:00.051

```

Data output Messages Explain Notifications

Graphical Analysis Statistics

Successfully run. Total query runtime: 51 msec. 1 rows affected.

Successfully run. Total query runtime: 137 msec. 4 rows affected.

Ln 242, Col 17

Activities pgAdmin 4 Jun 9 6:07 PM ● pgAdmin 4

Servers Local PostgreSQL schema3/postgres* Jun 9 6:07 PM ● pgAdmin 4

Databases Query History Execute/Refresh (F5)

```

210
211 set enable_hashagg = off;
212 set enable_hashjoin = off;
213 set enable_seqscan = on;
214
-- Query 8 (STATISTICS)
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
-- Query 8 optimized (STATISTICS)
233
234 -- Query 9 view table of reserves sids that have red boats
235 create MATERIALIZED VIEW query_8
236 as
237 select r1.sid
238 from (select r.sid
239         from reserves r
240       where exists (select * from boat b where r.bid=b.bid and color='red' )) as r1 ;
241
242 explain analyze select s.sname
243 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
244
245
246
247
248
-- ====== Query 9 ======
249
250 -- Find the names of sailors who have reserved both a red and a green boat.
251
252 -- Query 9 (COUNTING RESULT SET) , Number Of Rows = 177
253
254
255 select count(s.sname)
256 from sailors s, reserves r, boat b
257 where
Total rows: 12 of 12 Query complete 00:00:00.069

```

Data output Messages Explain Notifications

QUERY PLAN text

- Merge Semi Join (cost=1774.32..1806.34 rows=673 width=21) (actual time=5.560..6.097 rows=673 loops=1)
- Merge Cond: (s.sid = r1.sid)
- Sort (cost=1699.30..1746.80 rows=19000 width=25) (actual time=5.314..5.455 rows=2999 loops=1)
- Sort Key: s.sid
- Sort Method: quicksort Memory: 225kB
- Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.010..1.995 rows=19000 loops=1)
- Sort (cost=75.01..77.85 rows=1136 width=4) (actual time=0.243..0.295 rows=1136 loops=1)
- Sort Key: r1.sid
- Sort Method: quicksort Memory: 102kB
- Seq Scan on query_8 r1 (cost=0.00..17.36 rows=1136 width=4) (actual time=0.009..0.087 rows=1136 loops=1)
- Planning Time: 0.078 ms
- Execution Time: 6.290 ms

Successfully run. Total query runtime: 69 msec. 12 rows affected.

Successfully run. Total query runtime: 137 msec. 4 rows affected.

Ln 242, Col 1

Explanation :

- Metrics :

Execution Time : 6.290 ms Total Expected Cost : 1806.34

- Reason :

- This Query Improved in the Execution time and Expected Cost than the Original Query from 5664.74 to 1806.34 .
 - Because I used Materialized Views which already made an Intermediate Ready Table with smaller Size that Optimized Query used it which decreased the number of steps needed(Filtration of the table over red boats reserve sids) for the Query to get Executed .
 - Because the loops ends faster and exits due to I used the Exist Operator instead of the In Operator that waits till the end.

2. given query with B+ trees indices only :

The screenshot shows the pgAdmin 4 interface with the following details:

- Top Bar:** Activities, pgAdmin 4, Jun 9 6:12 PM, pgAdmin 4.
- Servers:** Local, schema3/postgres@postgres.
- Data Query Tab:** Contains a list of queries numbered 92 to 100. The first few queries are:

```
select count(s.sname)
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);

CREATE INDEX b_sailorsSID ON sailors USING btree(sid );
CREATE INDEX b_reservesSID ON reserves USING btree(sid );
CREATE INDEX b_reservesBID ON reserves USING btree(bid);
CREATE INDEX b_boat ON boat USING btree(bid,color );
CREATE INDEX b_query_8 ON query_8 USING btree(sid );

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_8';

DROP INDEX  IF EXISTS b_sailorsSID cascade;
DROP INDEX  IF EXISTS b_reservesSID cascade;
DROP INDEX  IF EXISTS b_reservesBID cascade;
DROP INDEX  IF EXISTS R_reservesBID cascade;

set enable_hashagg = off;
set enable_hashjoin = off;
set enable_seqscan = on;
-- Query 8 (STATISTICS)
```
- Data Output Tab:** Shows the results of the last query, which created indexes. The table has columns: schemaname, tablename, indexname, and indexdef. The results are:

schemaname	tablename	indexname	tablespace	Indexdef
public	sailors	b_sailorsSID	[null]	CREATE INDEX b_sailorsSID ON public.sailors USING btree (sid)
public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesSID ON public.reserves USING btree (sid)
public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesBID ON public.reserves USING btree (bid)
public	boat	b_boat	[null]	CREATE INDEX b_boat ON public.boat USING btree (bid,color)
public	query_8	b_query_8	[null]	CREATE INDEX b_query_8 ON public.query_8 USING btree (sid)
- Status Bar:** Successfully run. Total query runtime: 137 msec. 4 rows affected. Ln 206, Col 1.

The screenshot shows the pgAdmin 4 interface. The left sidebar contains icons for various database management tasks. The main window has a title bar "pgAdmin 4" and a status bar "Jun 9 6:14 PM". The left pane shows a tree view of servers and databases, with "schema3/postgres@postgres*" selected. The central pane displays a query editor with several lines of SQL code. The right pane shows a "Query Plan" diagram for a specific query, illustrating the execution flow and data flow between tables like b_sailorssid and b_query_8 through a "Merge Semi Join". A message bar at the bottom indicates successful runs and total query runtime.

```
set enable_hashagg = off;
set enable_hashjoin = off;
set enable_seadscan = on;
-- Query 8 (STATISTICS)
explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
```

```
-- Query 8 optimized (STATISTICS)
-- Query 8 view table of reserves sids that have red boats
create MATERIALIZED VIEW query_8
as
select r1.sid
from (select r.sid
      from reserves r
      where exists (select * from boat b where r.bid=b.bid and color='red' )) as r1;

explain analyze select s.sname
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);

-- ====== Query 9 ======
-- Find the names of sailors who have reserved both a red and a green boat.
-- Query 9 (COUNTING RESULT SET) , Number Of Rows = 177

select count(s.sname)
from sailors s, reserves r, boat b
where
```

Data output Messages Explain Notifications

Graphical Analysis Statistics

b_sailorssid Merge Semi Join

b_query_8

Successfully run. Total query runtime: 58 msec. 1 rows affected.

Successfully run. Total query runtime: 137 msec. 4 rows affected.

```

219 set enable_hashagg = off;
220 set enable_hashjoin = off;
221 set enable_seqscan = on;
222 -- Query 8 (STATISTICS)
223
224
225
226 explain analyze select s.sname
227 from sailors s
228 where s.sid in ( select r.sid
229 from reserves r
230 where r.bid in (select b.bid
231 from boat b
232 where b.color = 'red'));
233
234 -- Query 8 optimized (STATISTICS)
235
236 -- Query 8 view table of reserves sids that have red boats
237 create MATERIALIZED VIEW query_8
238 as
239 select r1.sid
240 from (select r.sid
241 from reserves r
242 where exists (select * from boat b where r.bid=b.bid and color='red' )) as r1 ;
243
244 explain analyze select s.sname
245 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
246
247
248
249
250 -- ====== Query 9 ======
251
252 -- Find the names of sailors who have reserved both a red and a green boat.
253
254 -- Query 9 (COUNTING RESULT SET) , Number Of Rows = 177
255
256
257 select count(s.sname)
258 from sailors s, reserves r, boat b
259 where

```

Total rows: 7 of 7 Query complete 00:00:00.059

Successfully run. Total query runtime: 137 msec. 4 rows affected.
Ln 244, Col 1

Explanation :

- Metrics :

Execution Time : 1.157 ms Total Expected Cost :191.19

- The B-tree helped in the performance it decreased the Execution Time and Expected Cost (BEST OF THEM ALL).
- The Query Planner used the B-tree index because B-Tree is $O(\log n)$ performance with Exact Values .
- Here it showed the improvement due to for ($s.sid=R.sid$ (it used the index that was built on query_8 view)) condition of Joining in the Query the Merge Semi Join used index scan using ZigZag and algorithm and these columns where built on it an b-tree index.

- given query with hash indices only :

pgAdmin 4

Jun 9 6:19 PM ● pgAdmin 4

Servers Local PostgreSQL

Dat Query History

```

> p 196
> s 197
> s 198
> s 199
> s 200 CREATE INDEX b_sailorsSID ON sailors USING hash(sid );
> s 201 CREATE INDEX b_reservesID ON reserves USING hash(sid );
> s 202 CREATE INDEX b_reservesBID ON reserves USING hash(bid );
> s 203 CREATE INDEX b_boat1 ON boat USING hash(bid );
> s 204 CREATE INDEX b_boat2 ON boat USING hash(color );
> s 205
> s 206 CREATE INDEX b_query_8 ON query_8 USING hash(sid );
> s 207
> s 208
> s 209 select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_8';
> s 210
> s 211
> s 212
> Log
> Tab 214
> postgres 215 DROP INDEX IF EXISTS b_sailorsSID cascade;
216 DROP INDEX IF EXISTS b_reservesID cascade;
217 DROP INDEX IF EXISTS b_reservesBID cascade;
218 DROP INDEX IF EXISTS b_boat1 cascade;
219 DROP INDEX IF EXISTS b_boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = on;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red')) );
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
Total rows: 6 of 6 Query complete 00:00:00.169

```

Data output Messages Explain Notifications

schema name	table name	index name	tablespace	indexdef
public	sailors	b_sailorsSID	[null]	CREATE INDEX b_sailorsSID ON public.sailors USING hash (sid)
public	reserves	b_reservesID	[null]	CREATE INDEX b_reservesID ON public.reserves USING hash (sid)
public	reserves	b_reservesBID	[null]	CREATE INDEX b_reservesBID ON public.reserves USING hash (bid)
public	boat	b_boat1	[null]	CREATE INDEX b_boat1 ON public.boat USING hash (bid)
public	boat	b_boat2	[null]	CREATE INDEX b_boat2 ON public.boat USING hash (color)
public	query_8	b_query_8	[null]	CREATE INDEX b_query_8 ON public.query_8 USING hash (sid)

Successfully run. Total query runtime: 137 msec. 4 rows affected. Ln 198, Col 1

pgAdmin 4

Jun 9 6:21 PM ● pgAdmin 4

Servers Local PostgreSQL

Dat Query History

```

> p 216 DROP INDEX IF EXISTS b_reservesID cascade;
> s 217 DROP INDEX IF EXISTS b_reservesBID cascade;
> s 218 DROP INDEX IF EXISTS b_boat1 cascade;
> s 219 DROP INDEX IF EXISTS b_boat2 cascade;
> s 220
> s 221 DROP INDEX IF EXISTS b_query_8 cascade;
> s 222
> s 223
> s 224 set enable_hashagg = off;
> s 225 set enable_hashjoin = off;
> s 226 set enable_seqscan = on;
> s 227 -- Query 8 (STATISTICS)
> s 228
> s 229
> s 230
> s 231 explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red')) );
> Log
> Tab 232
> postgres 233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
-- Query 8 optimized (STATISTICS)
-- Query 8 view table of reserves sids that have red boats
create MATERIALIZED VIEW query_8
as
select r1.sid
from (select r.sid
      from reserves r
      where exists (select * from boat b where r.bid=b.bid and color='red' ) )
explain analyze select s.sname
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);

```

Data output Messages Explain Notifications

Graphical Analysis Statistics

```

graph LR
    sailors[sailors] --> NestedLoopSemiJoin[Nested Loop Semi Join]
    NestedLoopSemiJoin --> b_query_8[b_query_8]

```

Successfully run. Total query runtime: 56 msec. 1 rows affected. Ln 249, Col 17

Successfully run. Total query runtime: 137 msec. 4 rows affected. Ln 249, Col 17

```

Activities pgAdmin 4 Jun 9 6:21 PM ●
File Object Tools Help
Servers Properties SQL Dependents Statistics Dashboard schema3/postgres@postgres*
Local [ postgres Data Query History Execute/Refresh
216 DROP INDEX IF EXISTS b.reserve CASCADE;
217 DROP INDEX IF EXISTS b.reserve$ID cascade;
218 DROP INDEX IF EXISTS b.boat1 cascade;
219 DROP INDEX IF EXISTS b.boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = on;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
232
233
234
235
236
237
238
239 -- Query 8 optimized (STATISTICS)
240
241 -- Query 8 view table of reserves sids that have red boats
242 create MATERIALIZED VIEW query_8
243 as
244 select r1.sid
245 from (select r.sid
246 from reserves r
247 where exists (select * from boat b where r.bid=b.bid and color='red' )) r1;
248
249 explain analyze select s.sname
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
250
251
252
253
254
255 -- ====== Query 9 ======
256
Total rows: 6 of 6 Query complete 00:00:00.063

```

QUERY PLAN
text

- 1 Nested Loop Semi Join (cost=0.00..721.68 rows=673 width=21) (actual time=0.016..11.861 rows=673 loops=1)
 - 2 -> Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.006..1.457 rows=19000 loops=1)
 - 3 -> Index Scan using b_query_8 on query_8 r1 (cost=0.00..0.04 rows=2 width=4) (actual time=0.000..0.000 rows=0 loops=19000)
 - 4 Index Cond: (sid = s.sid)
 - 5 Planning Time: 0.150 ms
 - 6 Execution Time: 11.896 ms

Successfully run. Total query runtime: 63 msec. 6 rows affected.

Successfully run. Total query runtime: 137 msec. 4 rows affected.

Explanation :

- Metrics :

Execution Time : 11.896 ms Total Expected Cost : 721.68

- The Hash helped in the performance it decreased the Execution Time (but in the execution time processor was overwhelmed) and Expected Cost .
- The Query Planner used the Hash index because Hash is O(1) performance with Exact Values and considered the best for exact values queries.
- Here it showed the improvement due to for every condition of Joining in the Query the Nested Loop Semi Join using index scan using Hash based algorithm on the condition (sid(from query_8 view)=s.sid) which approximatly maded to be O(n) performance.

4. given query with BRIN indices only :

pgAdmin 4

Jun 9 6:22 PM • pgAdmin 4

Servers schema3/postgres Local PostgreSQL Data Query History

```

193 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
194
195
196
197
198
199
200 CREATE INDEX b_sailorsSID ON sailors USING brin(sid );
201 CREATE INDEX b_reservesSID ON reserves USING brin(sid );
202 CREATE INDEX b_reservesBID ON reserves USING brin(bid );
203 CREATE INDEX b_boat1 ON boat USING brin(bid );
204 CREATE INDEX b_boat2 ON boat USING brin(color );
205
206 CREATE INDEX b_query_8 ON query_8 USING brin(sid );
207
208
209 select *
210 from pg_indexes
211 where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_8';
212
213
214
215 DROP INDEX IF EXISTS b_sailorsSID cascade;
216 DROP INDEX IF EXISTS b_reservesSID cascade;
217 DROP INDEX IF EXISTS b_reservesBID cascade;
218 DROP INDEX IF EXISTS b_boat1 cascade;
219 DROP INDEX IF EXISTS b_boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = on;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
232 from sailors s
233 where s.sid in ( select r.sid
234
Total rows: 6 of 6 Query complete 00:00:00.067

```

Data output Messages Explain Notifications

schemaname	tablename	indexname	tablespace	Indexdef
1 public	sailors	b_sailorsSID	[null]	CREATE INDEX b_sailorsSID ON public.sailors USING brin(sid)
2 public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesSID ON public.reserves USING brin(sid)
3 public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesBID ON public.reserves USING brin(bid)
4 public	boat	b_boat1	[null]	CREATE INDEX b_boat1 ON public.boat USING brin(bid)
5 public	boat	b_boat2	[null]	CREATE INDEX b_boat2 ON public.boat USING brin(color)
6 public	query_8	b_query_8	[null]	CREATE INDEX b_query_8 ON public.query_8 USING brin(sid)

Successfully run. Total query runtime: 137 msec. 4 rows affected. Ln 208, Col 1

pgAdmin 4

Jun 9 6:25 PM • pgAdmin 4

Servers schema3/postgres Local PostgreSQL Data Query History

```

217 DROP INDEX IF EXISTS b_reservesBID cascade;
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

```

Explain Analyze [Shift+F7]

Graphical Analysis Statistics

```

graph LR
    A[query_8] --> B[Sort]
    B --> C[Unique]
    C --> D[Nested Loop Inner Join]
    E[b_sailorsSID] --> F[sailors]

```

Successfully run. Total query runtime: 137 msec. 4 rows affected. Ln 249, Col 17

```

217 DROP INDEX IF EXISTS b_reser cascade;
218 DROP INDEX IF EXISTS b_boat1 cascade;
219 DROP INDEX IF EXISTS b_boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = off;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
232 from sailors s
233 where s.sid in ( select r.sid
234 from reserves r
235 where r.bid in (select b.bid
236 from boat b
237 where b.color = 'red'));
238
239 -- Query 8 optimized (STATISTICS)
240
241 -- Query 8 view table of reserves sids that have red boats
242 create MATERIALIZED VIEW query_8
243 as
244 select r1.sid
245 from (select r.sid
246 from reserves r
247 where exists (select * from boat b where r.bid=b.bid and color='red')) as r1;
248
249 explain analyze select s.sname
250 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
251
252
253
254
255
256
257 -- ====== Query 9 ======
-- Find the names of sailors who have reserved both a red and a green boat.

Total rows: 18 of 18   Query complete 00:00:00.875

```

Successfully run. Total query runtime: 137 msec. 4 rows affected.
Ln 249, Col 1

Explanation :

- Metrics :

Execution Time : 859.141 ms Total Expected Cost : 10005517867.87

- Here the BRIN was not used in the original Query Plan settings (Hashjoin and HashAgg are off) so I've made seqscan=off too.
- The Execution Time and Expected Cost became the (WORST OF THEM ALL) .
- This happened because the Query Optimizer didn't use it from the first place due to BRIN Usage here was not suitable so we have used it to simulate seqscan behaviour only we traversed it all and followed all its pointers so it is worst index to use in this case.
- Because there were no Aggregation used and was not low selectivity Query so it was not helpful to the performance.
- The plan performs a (Bit map index scan) on all indices and then to (Bit map heap scan) to select relevant rows .

- given query with mixed indices (any mix of your choice) :

pgAdmin 4

Jun 9 6:43 PM ● pgAdmin 4

Activities pgAdmin 4 ▾

Servers schema3/postgres Local [Data Query History Execute/Refresh]

```

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
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
747
748
749
749
750
751
752
753
754
755
756
757
757
758
759
759
760
761
762
763
764
765
766
767
767
768
769
769
770
771
772
773
774
775
776
777
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
837
838
839
839
840
841
842
843
844
845
846
847
847
848
849
849
850
851
852
853
854
855
856
857
857
858
859
859
860
861
862
863
864
865
866
867
867
868
869
869
870
871
872
873
874
875
876
877
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
927
928
929
929
930
931
932
933
934
935
936
937
937
938
939
939
940
941
942
943
944
945
946
947
947
948
949
949
950
951
952
953
954
955
956
957
957
958
959
959
960
961
962
963
964
965
966
966
967
968
968
969
970
971
972
973
974
975
976
977
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1066
1067
1068
1068
1069
1070
1071
1072
1073
1074
1075
1076
1076
1077
1078
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1098
1099
1100
1101
1102
1103
1104
1105
1105
1106
1107
1107
1108
1109
1110
1111
1112
1113
1114
1115
1115
1116
1117
1117
1118
1119
1120
1121
1122
1123
1124
1125
1125
1126
1127
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1136
1137
1138
1138
1139
1140
1141
1142
1143
1144
1145
1145
1146
1147
1147
1148
1149
1150
1151
1152
1153
1154
1155
1155
1156
1157
1157
1158
1159
1160
1161
1162
1163
1164
1165
1165
1166
1167
1167
1168
1169
1170
1171
1172
1173
1174
1175
1175
1176
1177
1177
1178
1179
1180
1181
1182
1183
1184
1185
1185
1186
1187
1187
1188
1189
1190
1191
1192
1193
1194
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1205
1206
1207
1207
1208
1209
1210
1211
1212
1213
1214
1214
1215
1216
1216
1217
1218
1219
1219
1220
1221
1221
1222
1223
1223
1224
1225
1225
1226
1227
1227
1228
1229
1229
1230
1231
1231
1232
1233
1233
1234
1235
1235
1236
1237
1237
1238
1239
1239
1240
1241
1241
1242
1243
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1251
1252
1253
1253
1254
1255
1255
1256
1257
1257
1258
1259
1259
1260
1261
1261
1262
1263
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1271
1272
1273
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1281
1282
1283
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1291
1292
1293
1293
1294
1295
1295
1296
1297
1297
1298
1299
1299
1300
1301
1301
1302
1303
1303
1304
1305
1305
1306
1307
1307
1308
1309
1309
1310
1311
1311
1312
1313
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1321
1322
1323
1323
1324
1325
1325
1326
1327
1327
1328
1329
1329
1330
1331
1331
1332
1333
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1341
1342
1343
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1351
1352
1353
1353
1354
1355
1355
1356
1357
1357
1358
1359
1359
1360
1361
1361
1362
1363
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1371
1372
1373
1373
1374
1375
1375
1376
1377
1377
1378
1379
1379
1380
1381
1381
1382
1383
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1391
1392
1393
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1401
1402
1403
1403
1404
1405
1405
1406
1407
1407
1408
1409
1409
1410
1411
1411
1412
1413
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1421
1422
1423
1423
1424
1425
1425
1426
1427
1427
1428
1429
1429
1430
1431
1431
1432
1433
1433
1434
1435
1435
1436
1437
1437
1438
1439
1439
1440
1441
1441
1442
1443
1443
1444
1445
1445
1446
1447
1447
1448
1449
1449
1450
1451
1451
1452
1453
1453
1454
1455
1455
1456
1457
1457
1458
1459
1459
1460
1461
1461
1462
1463
1463
1464
1465
1465
1466
1467
1467
1468
1469
1469
1470
1471
1471
1472
1473
1473
1474
1475
1475
1476
1477
1477
1478
1479
1479
1480
1481
1481
1482
1483
1483
1484
1485
1485
1486
1487
1487
1488
1489
1489
1490
1491
1491
1492
1493
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1501
1502
1503
1503
1504
1505
1505
1506
1507
1507
1508
1509
1509
1510
1511
1511
1512
1513
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1521
1522
1523
1523
1524
1525
1525
1526
1527
1527
1528
1529
1529
1530
1531
1531
1532
1533
1533
1534
1535
1535
1536
1537
1537
1538
1539
1539
1540
1541
1541
1542
1543
1543
1544
1545
1545
1546
1547
1547
1548
1549
1549
1550
1551
1551
1552
1553
1553
1554
1555
1555
1556
1557
1557
1558
1559
1559
1560
1561
1561
1562
1563
1563
1564
1565
1565
1566
1567
1567
1568
1569
1569
1570
1571
1571
1572
1573
1573
1574
1575
1575
1576
1577
1577
1578
1579
1579
1580
1581
1581
1582
1583
1583
1584
1585
1585
1586
1587
1587
1588
1589
1589
1590
1591
1591
1592
1593
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1601
1602
1603
1603
1604
1605
1605
1606
1607
1607
1608
1609
1609
1610
1611
1611
1612
1613
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1621
1622
1623
1623
1624
1625
1625
1626
1627
1627
1628
1629
1629
1630
1631
1631
1632
1633
1633
1634
1635
1635
1636
1637
1637
1638
1639
1639
1640
1641
1641
1642
1643
1643
1644
1645
1645
1646
1647
1647
1648
1649
1649
1650
1651
1651
1652
1653
1653
1654
1655
1655
1656
1657
1657
1658
1659
1659
1660
1661
1661
1662
1663
1663
1664
1665
1665
1666
1667
1667
1668
1669
1669
1670
1671
1671
1672
1673
1673
1674
1675
1675
1676
1677
1677
1678
1679
1679
1680
1681
1681
1682
1683
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1691
1692
1693
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1701
1702
1703
1703
1704
1705
1705
1706
1707
1707
1708
1709
1709
1710
1711
1711
1712
1713
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1721
1722
1723
1723
1724
1725
1725
1726
1727
1727
1728
1729
1729
1730
1731
1731
1732
1733
1733
1734
1735
1735
1736
1737
1737
1738
1739
1739
1740
1741
1741
1742
1743
1743
1744
1745
1745
1746
1747
1747
1748
1749
1749
1750
1751
1751
1752
1753
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1761
1762
1763
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1771
1772
1773
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1781
1782
1783
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1791
1792
1793
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1801
1802
1803
1803
1804
1805
1805
1806
1807
1807
1808
1809
1809
1810
1811
1811
1812
1813
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1821
1822
1823
1823
1824
1825
1825
1826
1827
1827
1828
1829
1829
1830
1831
1831
1832
1833
1833
1834
1835
1835
1836
1837
1837
1838
1839
1839
1840
1841
1841
1842
1843
1843
1844
1845
1845
1846
1847
1847
1848
1849
1849
1850
1851
1851
1852
1853
1853
1854
1855
1855
1856
1857
1857
1858
1859
1859
1860
1861
1861
1862
1863
1863
1864
1865
1865
1866
1867
1867
1868
1869
1869
1870
1871
1871
1872
1873
1873
1874
1875
1875
1876
1877
1877
1878
1879
1879
1880
1881
1881
1882
1883
1883
1884
1885
1885
1886
1887
1887
1888
1889
1889
1890
1891
1891
1892
1893
1893
1894
1895
1895
1896
1897
1897
1898
1899
1899
1900
1901
1901
1902
1903
1903
1904
1905
1905
1906
1907
1907
1908
1909
1909
1910
1911
1911
1912
1913
1913
1914
1915
1915
1916
1917
1917
1918
1919
1919
1920
1921
1921
1922
1923
1923
1924
1925
1925
1926
1927
1927
1928
1929
1929
1930
1931
1931
1932
1933
1933
1934
1935
1935
1936
1937
1937
1938
1939
1939
1940
1941
1941
1942
1943
1943
1944
1945
1945
1946
1947
1947
1948
1949
1949
1950
1951
1951
1952
1953
1953
1954
1955
1955
1956
1957
1957
1958
1959
1959
1960
1961
1961
1962
1963
1963
1964
1965
1965
1966
1967
1967
1968
1969
1969
1970
1971
1971
1972
1973
1973
1974
1975
1975
1976
1977
1977
1978
1979
1979
1980
1981
1981
1982
1983
1983
1984
1985
1985
1986
1987
1987
1988
1989
1989
1990
1991
1991
1992
1993
1993
1994
1995
1995
1996
1997
1997
1998
1999
1999
2000
2001
2001
2002
2003
2003
2004
2005
2005
2006
2007
2007
2008
2009
2009
2010
2011
2011
2012
2013
2013
2014
2015
2015
2016
2017
2017
2018
2019
2019
2020
2021
2021
2022
2023
2023
2024
2025
2025
2026
2027
2027
2028
2029
2029
2030
2031
2031
2032
2033
2033
2034
2035
2035
2036
2037
2037
2038
2039
2039
2040
2041
2041
2042
2043
2043
2044
2045
2045
2046
2
```

```

Activities pgAdmin 4 Jun 9 6:45 PM ● pgAdmin 4
File Object Tools Help
Servers Properties SQL Dependents Statistics Dashboard schema3/postgres@postgres*
Local [ No limit ] Data Query History Execute/Refresh
> P 221 DROP INDEX IF EXISTS b_query;
> S 222
> S 223
> S 224 set enable_hashagg = off;
> S 225 set enable_hashjoin = off;
> S 226 set enable_seqscan = on;
-- Query 8 (STATISTICS)
> S 227 explain analyze select s.sname
> S 228 from sailors s
> S 229 where s.sid in ( select r.sid
> S 230 from reserves r
> S 231 where r.bid in (select b.bid
> S 232 from boat b
> S 233 where b.color = 'red'));
> S 234
-- Query 8 optimized (STATISTICS)
> S 235
> S 236
> S 237
> S 238
-- Query 8 view table of reserves sids that have red boats
> S 239 create MATERIALIZED VIEW query_8
> S 240 as
> S 241 select r1.sid
> S 242 from (select r.sid
> S 243 from reserves r
> S 244 where exists (select * from boat b where r.bid=b.bid and color='red' ) ) a
> S 245
> S 246 explain analyze select s.sname
> S 247 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
> S 248
> S 249
> S 250
-- ====== Query 9 ======
> S 251 -- Find the names of sailors who have reserved both a red and a green boat.
> S 252
> S 253
-- Query 9 (COUNTING RESULT SET) , Number Of Rows = 177
> S 254
> S 255
> S 256
-- Query 9 optimized (STATISTICS)
> S 257
> S 258
-- Query 9 (COUNTING RESULT SET) , Number Of Rows = 177
> S 259
> S 260
> S 261
Total rows: 10 of 10 Query complete 00:00:00.071

```

QUERY PLAN
text

- 1 Nested Loop (cost=75.01..999.20 rows=673 width=21) (actual time=0.254..1.173 rows=673 loops=1)
 - 2 -> Unique (cost=75.01..80.69 rows=673 width=4) (actual time=0.245..0.414 rows=673 loops=1)
 - 3 -> Sort (cost=75.01..77.85 rows=1136 width=4) (actual time=0.245..0.304 rows=1136 loops=1)
 - 4 Sort Key:r1.sid
 - 5 Sort Method: quicksort Memory: 102K
 - 6 -> Seq Scan on query_8 r1 (cost=0.00..17.36 rows=1136 width=4) (actual time=0.009..0.089 rows=1136 loops=1)
 - 7 -> Index Scan using b_sailorssid on sailors s (cost=0.00..1.35 rows=1 width=25) (actual time=0.001..0.001 rows=1 loops=673)
 - 8 Index Cond: (sid = r1.sid)
 - 9 Planning Time: 0.135 ms
 - 10 Execution Time: 1.209 ms

Successfully run. Total query runtime: 137 msec. 4 rows affected.
Ln 249, Col 1

Explanation :

- Metrics :

Execution Time : 1.209 ms Total Expected Cost : 999.20

- The Query Planner used the Merge join by using both the ZigZag join and the Hash based algorithm together which improved the Execution time and the expected cost way more better which made the join in $O(n \log n)$.
- The Mix indices helped in the performance it decreased the Execution Time and Expected Cost .