

Query 9

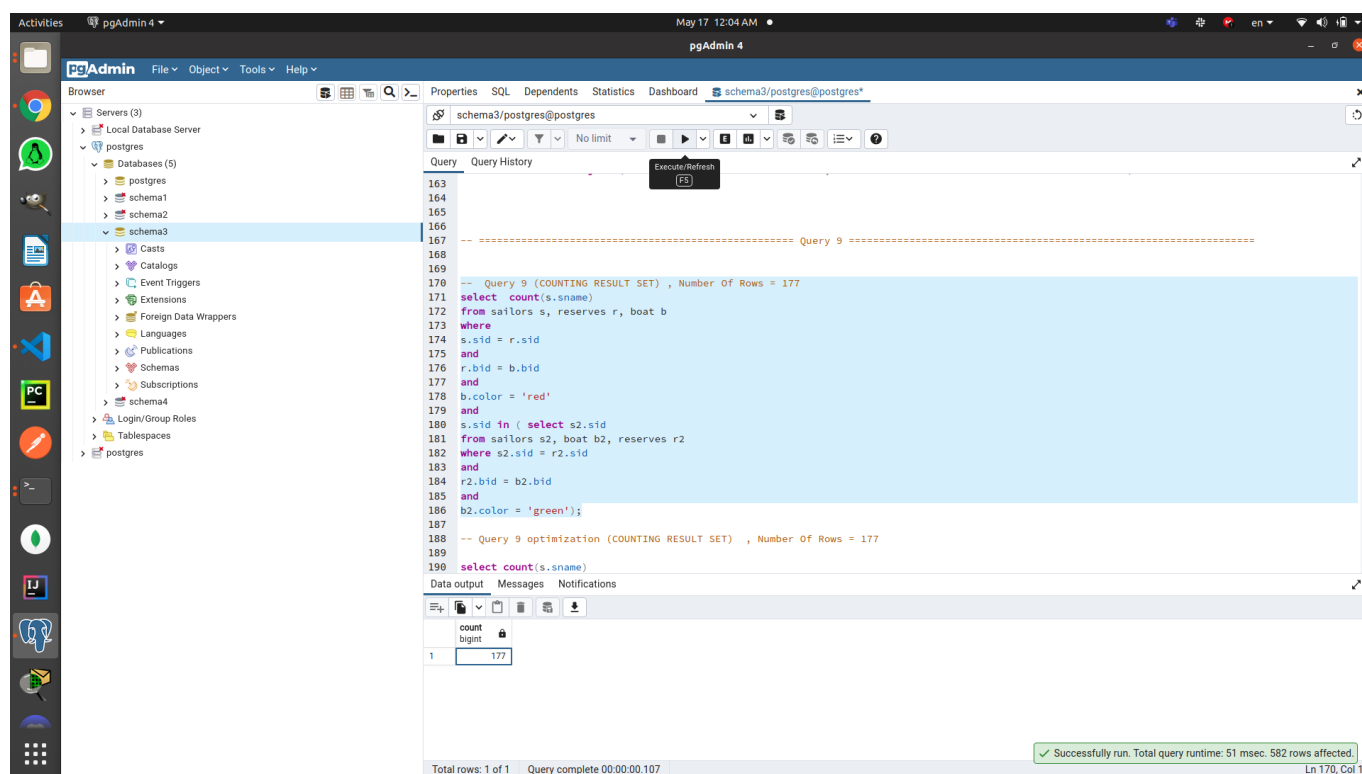
- Find the names of sailors who have reserved both a red and a green boat.

Original Query

```
select s.sname
from sailors s, reserves r, boat b
where
s.sid = r.sid
and
r.bid = b.bid
and
b.color = 'red'
and
s.sid in ( select s2.sid
from sailors s2, boat b2, reserves r2
where s2.sid = r2.sid
and
r2.bid = b2.bid
and
b2.color = 'red');
```

Result Set

- 177 Rows



The screenshot displays the pgAdmin 4 interface. The SQL editor shows the following query (lines 163-190):

```
163
164
165
166 ----- Query 9 -----
167
168
169
170 -- Query 9 (COUNTING RESULT SET) , Number Of Rows = 177
171 select count(s.sname)
172 from sailors s, reserves r, boat b
173 where
174 s.sid = r.sid
175 and
176 r.bid = b.bid
177 and
178 b.color = 'red'
179 and
180 s.sid in ( select s2.sid
181 from sailors s2, boat b2, reserves r2
182 where s2.sid = r2.sid
183 and
184 r2.bid = b2.bid
185 and
186 b2.color = 'green');
187
188 -- Query 9 optimization (COUNTING RESULT SET) , Number Of Rows = 177
189
190 select count(s.sname)
```

The Data output pane shows the result set:

count bigint
177

The status bar at the bottom indicates: "Successfully run. Total query runtime: 51 msec. 582 rows affected. Ln 170, Col 1"

Report

1. given query without an index,

pgAdmin 4

schema3/postgres@postgres*

Query

```
15
16 select count(sid)
17 from sailors;
18
19 select count(*)
20 from boat;
21
22 select count(*)
23 from reserves;
24
25 delete from Reserves;
26
27 delete from sailors;
28
29 delete from Boat;
30
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 Messages Notifications

schemaname	tablename	indexname	tablespace	indexdef
name	name	name	name	text

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

Total rows: 0 of 0 Query complete 00:00:00.117 Ln 35, Col 1

pgAdmin 4

schema3/postgres@postgres*

Query

214 and

Data output Messages Explain X Notifications

Graphical Analysis Statistics

Total rows: 1 of 1 Query complete 00:00:00.123 Ln 226, Col 1

The screenshot shows the pgAdmin 4 interface with a SQL query in the Query Editor and its execution plan in the Data output pane.

Query:

```

-- Query 9 (STATISTICS)
-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.

explain analyze select s.sname
from sailors s
where exists
(
  select rTotal.sid
  from (select r1.sid
        from reserves r1
        inner join
        (select r2.sid
         from reserves r2
         where s2.sid = r2.sid
         and
         r2.bid = b2.bid
         and
         b2.color = 'green');
        where exists
        (select b1
         from boats b1
         where color = 'red' and r1.bid = b1.bid )
        ) as r2
  on r2.sid = r1.sid ) as rTotal
where rTotal.sid=s.sid

```

Execution Plan:

Step	Operation	Cost	Width	Actual Time	Actual Width	Loops
1	Hash Join	(cost=1968.71..2721.55 rows=1309 width=21)		(actual time=24.046..33.483 rows=177 loops=1)		
2	Hash Cond	(r.sid = s.sid)				
3	Hash Join	(cost=67.84..788.91 rows=4982 width=4)		(actual time=0.693..10.021 rows=1136 loops=1)		
4	Hash Cond	(r2.bid = b2.bid)				
5	Seq Scan on reserves r	(cost=0.00..540.00 rows=35000 width=8)		(actual time=0.015..4.099 rows=35000 loops=1)		
6	Hash	(cost=62.50..62.50 rows=427 width=4)		(actual time=0.671..0.671 rows=427 loops=1)		
7	Buckets	1024 Batches: 1 Memory Usage: 24kB				
8	Seq Scan on boat b	(cost=0.00..62.50 rows=427 width=4)		(actual time=0.009..0.586 rows=427 loops=1)		
9	Filter	(color = 'red'::bpchar)				
10	Rows Removed by Filter	2573				
11	Hash	(cost=1838.46..1838.46 rows=4993 width=33)		(actual time=23.225..23.229 rows=2064 loops=1)		
12	Buckets	8192 Batches: 1 Memory Usage: 202kB				
13	Hash Semi Join	(cost=1384.03..1638.46 rows=4993 width=33)		(actual time=17.250..22.749 rows=2064 loops=1)		
14	Hash Cond	(s.sid = s2.sid)				
15	Seq Scan on sailors s	(cost=0.00..349.00 rows=19000 width=25)		(actual time=0.009..2.314 rows=19000 loops=1)		
16	Hash	(cost=1321.62..1321.62 rows=4993 width=8)		(actual time=17.170..17.173 rows=2064 loops=1)		
17	Buckets	8192 Batches: 1 Memory Usage: 145kB				
18	Hash Join	(cost=851.44..1321.62 rows=4993 width=8)		(actual time=11.126..16.700 rows=2064 loops=1)		
19	Hash Cond	(s2.sid = r2.sid)				
20	Seq Scan on sailors s2	(cost=0.00..349.00 rows=19000 width=4)		(actual time=0.006..2.397 rows=19000 loops=1)		
21	Hash	(cost=789.03..789.03 rows=4993 width=4)		(actual time=11.046..11.048 rows=2064 loops=1)		
22	Buckets	8192 Batches: 1 Memory Usage: 137kB				
23	Hash Join	(cost=67.85..789.03 rows=4993 width=4)		(actual time=0.924..10.594 rows=2064 loops=1)		
24	Hash Cond	(r2.bid = b2.bid)				
25	Seq Scan on reserves r2	(cost=0.00..540.00 rows=35000 width=8)		(actual time=0.007..4.261 rows=35000 loops=1)		
26	Hash	(cost=62.50..62.50 rows=428 width=4)		(actual time=0.674..0.675 rows=428 loops=1)		
27	Buckets	1024 Batches: 1 Memory Usage: 24kB				
28	Seq Scan on boat b2	(cost=0.00..62.50 rows=428 width=4)		(actual time=0.078..0.578 rows=428 loops=1)		
29	Filter	(color = 'green'::bpchar)				
30	Rows Removed by Filter	2572				
31	Planning Time	0.438 ms				
32	Execution Time	33.541 ms				

Explanation :

- Metrics :

Execution Time : 33.462 ms Total Expected Cost : 2721.55

- First , a Sequential Scan occurred on boat b2 Table to filter the boats with color green .
- Second , a Hash Table was built on the run on b2 attributes cause a Hash Join this resulted with 1024 Buckets created ,and Memory Usage of 24 KB .
- Thirdly , a Sequential Scan occurred on reserves r2 Table to be able to hash each row's sid to the Boat b2's buckets to full inner join on the condition r2.bid= b2.bid .
- Fourthly , on the intermediate results a Hash table was built on it.
- Fifth , a Sequential Scan occurred on reserves s2 Table to be able to hash each row's sid to the Boat b2's buckets to full inner join on the condition s2.sid= r2.sid .
- Sixth , we repeat the process again with s1,b1,r1 and inner join all the results together with a hash semi join and based on the conditions.

2. given query with B+ trees indices only,

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

Query History:

```

259 >
260 >
261 >
262 CREATE INDEX b_sailorsSID ON sailors USING btree(sid);
263 CREATE INDEX b_reservesSID ON reserves USING btree(sid);
264 CREATE INDEX b_reservesBID ON reserves USING btree(bid);
265 CREATE INDEX b_boat ON boat USING btree(bid,color);
266 >
267 select *
268 from pg_indexes
269 where tablename = 'sailors' or tablename='reserves' or tablename='boat';
270 >
271 -- Query 9 (STATISTICS)
272 >
273 explain analyze select s.sname
274 from sailors s, reserves r, boat b
275 where
276 s.sid = r.sid
277 and
278 r.bid = b.bid
279 and
280 b.color = 'red'
281 and
282 s.sid in ( select s2.sid
283 from sailors s2, boat b2, reserves r2
284 where s2.sid = r2.sid
285 and
286 r2.bid = b2.bid
287 and
288 b2.color = 'green');
289 >
290
291 -- Query 9 optimization (STATISTICS)
292 -- Find the names of sailors who have reserved both a red and a green boat.
293 >
294
295 explain analyze select s.sname
296 from sailors s
297 where exists
298 (
299 select rTotal.sid
300 from (select r1.sid
301 from
302 (select r.sid
303 from reserves r
304 where exists

```

Data output:

	schemaname name	tablename name	indexname name	tablespace name	indexdef text
1	public	sailors	b_sailorssid	[null]	CREATE I...
2	public	reserves	b_reserves...	[null]	CREATE I...
3	public	reserves	b_reserves...	[null]	CREATE I...
4	public	boat	b_boat	[null]	CREATE I...

Total rows: 4 of 4 Query complete 00:00:00.055 Ln 270, Col 1

The top screenshot shows the query execution results in the 'Data output' tab. The query is:

```
(select b.bid
from boat b
where color = 'green' and r.bid = b.bid)
```

The results show 1 row. The bottom screenshot shows the 'Query Plan' tab for the same query. The plan is as follows:

```

1  Nested Loop (cost=1514.57..2711.33 rows=1309 width=21) (actual time=11.315..17.272 rows=177 loops=1)
2   -> Hash Join (cost=1514.29..2267.13 rows=1309 width=12) (actual time=11.302..16.999 rows=177 loops=1)
3     Hash Join (r.sid = s2.sid)
4     Hash Join (cost=67.84..788.91 rows=4982 width=4) (actual time=0.365..6.006 rows=1136 loops=1)
5     Hash Join (r.bid = b.bid)
6     Seq Scan on reserves r (cost=0.00..540.00 rows=35000 width=8) (actual time=0.007..2.577 rows=35000 loops=1)
7     Hash (cost=62.50..62.50 rows=427 width=4) (actual time=0.354..0.354 rows=427 loops=1)
8     Buckets: 1024 Batches: 1 Memory Usage: 24kB
9     Seq Scan on boat b (cost=0.00..62.50 rows=427 width=4) (actual time=0.006..0.306 rows=427 loops=1)
10    Filter (color = 'red'::bpchar)
11    Rows Removed by Filter: 2573
12    Hash (cost=1384.04..1384.04 rows=4993 width=8) (actual time=10.875..10.878 rows=2064 loops=1)
13    Buckets: 8192 Batches: 1 Memory Usage: 145kB
14    HashAggregate (cost=1334.11..1384.04 rows=4993 width=8) (actual time=10.304..10.614 rows=2064 loops=1)
15    Group Key: s2.sid
16    Hash Join (cost=851.44..1321.62 rows=4993 width=8) (actual time=6.598..9.885 rows=2064 loops=1)
17    Hash Join (s2.sid = r2.sid)
18    Seq Scan on sailors s2 (cost=0.00..349.00 rows=19000 width=4) (actual time=0.004..1.438 rows=19000 loops=1)
19    Hash (cost=789.03..789.03 rows=4993 width=4) (actual time=6.543..6.545 rows=2064 loops=1)
20    Buckets: 8192 Batches: 1 Memory Usage: 137kB
21    Hash Join (cost=67.85..789.03 rows=4993 width=4) (actual time=0.488..6.305 rows=2064 loops=1)
22    Hash Join (r2.bid = b2.bid)
23    Seq Scan on reserves r2 (cost=0.00..540.00 rows=35000 width=8) (actual time=0.004..2.665 rows=35000 loops=1)
24    Hash (cost=62.50..62.50 rows=428 width=4) (actual time=0.348..0.348 rows=428 loops=1)
25    Buckets: 1024 Batches: 1 Memory Usage: 24kB
26    Seq Scan on boat b2 (cost=0.00..62.50 rows=428 width=4) (actual time=0.042..0.301 rows=428 loops=1)
27    Filter (color = 'green'::bpchar)
28    Rows Removed by Filter: 2572
29    Index Scan using b_sailorsSid on sailors s (cost=0.29..0.33 rows=1 width=25) (actual time=0.001..0.001 rows=1 loops=1)
30    Index Scan (sid = r.sid)
31    Planning Time: 1.220 ms
32    Execution Time: 17.333 ms

```

Explanation :

- Metrics :

Execution Time : 17.333 ms Total Expected Cost : 2711.33

- The B-tree helped in the performance it decreased the Execution Time and Expected Cost .
- The Query Planner used the B-tree index instead of

3. given query with hash indices only,
 -
4. given query with BRIN indices only,
 -
5. given query with mixed indices (any mix of your choice).
 -

Optimized Query

```
select s.sname
from sailors s
where exists
(
    select rTotal.sid
    from (select r1.sid
          from
            (select r.sid
             from reserves r
             where exists
               (select bid
                from boat b
                where color = 'green' and r.bid =b.bid )
            )as r1
          inner join
            (select r.sid
             from reserves r
             where exists
               (select bid
                from boat b
                where color = 'red' and r.bid =b.bid )
            ) as r2
          on r2.sid = r1.sid ) as rTotal
    where rTotal.sid=s.sid
)
```

Report

1. given query without an index,

pgAdmin 4

File Object Tools Help

schema3/postgres@postgres*

Servers (3)

Local Database Server

postgres

Databases (5)

postgres

schema1

schema2

schema3

Casts

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas

Subscriptions

schema4

Login/Group Roles

Tablespaces

postgres

Properties SQL Dependents Statistics Dashboard

schema3/postgres@postgres*

Query Query History

15

16 select count(sid)

17 from sailors;

18

19 select count(*)

20 from boat;

21

22 select count(*)

23 from reserves;

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 Messages Notifications

schema3name tablename indexname tablespace indexdef

name name name name text

Ln 35, Col 1

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

Total rows: 0 of 0 Query complete 00:00:00.117

The top screenshot shows the pgAdmin 4 interface with a query window displaying the following SQL query:

```

294
295 explain analyze select s.sname
296 from sailors s
297
298
299

```

The query plan diagram shows a Hash Semi Join operation on the 'sailors' table, followed by a Hash Inner Join operation. The plan also shows a Hash operation on the 'reserves' table, which is joined with the 'boat' table.

The bottom screenshot shows the same pgAdmin 4 interface with a more complex query plan. The query window displays the following SQL query:

```

299
300 -- Query 9 optimization (STATISTICS)
301 -- Find the names of sailors who have reserved both a red and a green boat.
302
303
304 explain analyze select s.sname
305 from sailors s
306 where exists
307 (
308   select rTotal.sid
309   from (select r1.sid
310         from reserves r1
311         where exists
312             (select bid
313              from boat b
314              where color = 'green' and r1.bid = b.bid)
315        ) as r1
316   inner join
317   (select r2.sid
318    from reserves r2
319    where exists
320        (select bid
321         from boat b
322         where color = 'red' and r2.bid = b.bid)
323        ) as r2
324   on r2.sid = r1.sid ) as rTotal
325 where rTotal.sid=s.sid
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344

```

The query plan diagram shows a Hash Semi Join operation on the 'sailors' table, followed by a Hash Inner Join operation. The plan also shows a Hash operation on the 'reserves' table, which is joined with the 'boat' table.

The 'Data output' section shows the following query plan statistics:

Step	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Actual Loops
1	Hash Semi Join	(cost=1621.20..2034.80 rows=1324 width=21)			(actual time=13.866..16.943 rows=177 loops=1)		
2	Hash Cond	(s.sid = r1.sid)					
3	Seq Scan on sailors s	(cost=0.00..349.00 rows=19000 width=25)			(actual time=0.007..1.498 rows=19000 loops=1)		
4	Seq Scan on reserves r	(cost=1604.65..1604.65 rows=1324 width=8)			(actual time=13.819..13.824 rows=177 loops=1)		
5	Buckets	2048 Batches: 1 Memory Usage: 23K					
6	Hash Join	(cost=885.26..1604.65 rows=1324 width=8)			(actual time=7.398..13.794 rows=177 loops=1)		
7	Hash Cond	(r1.sid = r2.sid)					
8	Hash Semi Join	(cost=67.85..755.27 rows=4993 width=4)			(actual time=0.503..6.706 rows=2064 loops=1)		
9	Hash Cond	(r.bid = b.bid)					
10	Seq Scan on reserves r	(cost=0.00..540.00 rows=35000 width=8)			(actual time=0.005..2.795 rows=35000 loops=1)		
11	Hash	(cost=62.50..62.50 rows=428 width=4)			(actual time=0.360..0.360 rows=428 loops=1)		
12	Buckets	1024 Batches: 1 Memory Usage: 24K					
13	Seq Scan on boat b	(cost=0.00..62.50 rows=428 width=4)			(actual time=0.046..0.311 rows=428 loops=1)		
14	Filter	(color = 'green'::bpchar)					
15	Rows Removed by Filter	2572					
16	Hash	(cost=755.14..755.14 rows=4982 width=4)			(actual time=6.886..6.887 rows=1136 loops=1)		
17	Buckets	8192 Batches: 1 Memory Usage: 104K					
18	Hash Semi Join	(cost=67.84..755.14 rows=4982 width=8)			(actual time=0.357..6.742 rows=1136 loops=1)		
19	Hash Cond	(r1.sid = b1.sid)					
20	Seq Scan on reserves r1	(cost=0.00..540.00 rows=35000 width=8)			(actual time=0.004..2.869 rows=35000 loops=1)		
21	Hash	(cost=62.50..62.50 rows=427 width=4)			(actual time=0.350..0.350 rows=427 loops=1)		
22	Buckets	1024 Batches: 1 Memory Usage: 24K					
23	Seq Scan on boat b1	(cost=0.00..62.50 rows=427 width=4)			(actual time=0.004..0.303 rows=427 loops=1)		
24	Filter	(color = 'red'::bpchar)					
25	Rows Removed by Filter	2573					
26	Planning Time	0.338 ms					
27	Execution Time	16.981 ms					

Explanation :

- First , a Sequential Scan occurred on Reserves Table to filter the reserved boats with bid 103 and it costs 0..627 rows, and read through 35000 Rows and 582 Rows are left after the filtration of the where condition.
- Second , a Hash Table was built on the run on r.bid of the 582 Rows and it costed 627.50...627.50 rows, this resulted with 1024 Buckets created , and Memory Usage of 29 KB.
- Thirdly , a Sequential Scan occurred on Sailors Table to be able to hash each row's sid to the Reserves's buckets to full inner join on the condition and it costed 00..349 rows, and it read through

19000 rows.

- Fourthly, a Hash Semi Join occurred to produce the result set of the condition of $s.sid = r.sid$, and the reason it is Hash Semi Join In the first query, only the $r.sid$ needs to be saved from the reserves into the hash table, because that is the only data needed to implement the semi-join, it costs 634.77 ... 1084.60 rows.

1. given query with B+ trees indices only,

-

3. given query with hash indices only,

-

4. given query with BRIN indices only,

-

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

-