

Query 1 with no indecies

Planning and execution time :

The screenshot shows the pgAdmin 4 interface with the title bar "pgAdmin 4". The left sidebar displays the database structure under "Servers" and "PostgreSQL". The main area shows a query editor with the following SQL code:

```
explain analyze select *  
from (select *  
      from student
```

Below the query, the "Data Output" tab is selected, showing the execution plan:

QUERY PLAN
text

```
5 [...] Rows Removed by Filter: 64900  
6 [...]-> Hash (cost=22.39..22.39 rows=125 width=40) (actual time=0.684..0.687 rows=250 loops=1)  
7 [...] Buckets: 1024 Batches: 1 Memory Usage: 26kB  
8 [...]-> Hash Join (cost=13.06..22.39 rows=125 width=40) (actual time=0.271..0.556 rows=250 loops=1)  
9 [...] Hash Cond: (t.section_id = s.section_id)  
10 [...]-> Seq Scan on takes t (cost=0..0.80 rows=500 width=12) (actual time=0.025..0.098 rows=500 loops=1)  
11 [...]-> Hash (cost=11.50..11.50 rows=125 width=28) (actual time=0.223..0.225 rows=250 loops=1)  
12 [...] Buckets: 1024 Batches: 1 Memory Usage: 23kB  
13 [...]-> Seq Scan on section s (cost=0..0.11 rows=125 width=28) (actual time=0.019..0.157 rows=250 loops=1)  
14 [...] Filter: ((semester = 1) AND (year = 2019))  
15 [...] Rows Removed by Filter: 250  
16 Planning Time: 0.657 ms  
17 Execution Time: 17.137 ms
```

Execution Plan and costs :

The screenshot shows the pgAdmin 4 interface with a query editor displaying the following SQL code:

```

> s 11 and
> p 12 year = 2019) as sem1_student
> s 13 on CS1_student.id = sem1_student.student_id;
> t

```

The Explain tab is selected, showing the following execution plan:

#	Node	Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Hash Full Join (cost=23.95..1394.23 rows=1137 width=68) (actual=0.43... Hash Cond: (student.id = t.student_id)	0.39 ms	11.416 ms	1 1.04	1100	1137	1
2.	→ Seq Scan on student as student (cost=0..1366 rows=1137 width=2... Filter: ((department).text ~ 'CSEN') Rows Removed by Filter: 64900	10.615 ms	10.615 ms	1 1.04	1100	1137	1
3.	→ Hash (cost=22.39..22.39 rows=125 width=40) (actual=0.41..0.412 ... Buckets: 1024 Batches: 1 Memory Usage: 26 kB	0.076 ms	0.412 ms	1 2	250	125	1
4.	→ Hash Inner Join (cost=13.06..22.39 rows=125 width=40) (actu... Hash Cond: (t.section_id = s.section_id)	0.151 ms	0.336 ms	1 2	250	125	1
5.	→ Seq Scan on takes as t (cost=0..8 rows=500 width=12) (a...	0.06 ms	0.06 ms	1 1	500	500	1
6.	→ Hash (cost=11..5.11.5 rows=125 width=28) (actual=0.12... Buckets: 1024 Batches: 1 Memory Usage: 23 kB	0.041 ms	0.126 ms	1 2	250	125	1
7.	→ Seq Scan on section as s (cost=0..11.5 rows=125 wi... Filter: ((semester = 1) AND (year = 2019)) Rows Removed by Filter: 250	0.086 ms	0.086 ms	1 2	250	125	1

Query 1 using B+ Tree on : Student(department) , section(year) , section(semester), takes(section_id)

Disabled flags: seqscan , bitmapscan

Planning and execution time:

```

pgAdmin 4
File Object Tools Help
Servers (1) PostgreSQL (1) schema1/postgres@PostgreSQL 13 * schema1/post...
Dashboard Properties SQL Statistics Dependencies Dependents No limit ▾
Data (1) Query Editor Query History
> S 1 create index stud on Student using btree(department);
> S 2 create index sec_y on section using btree(year);
> S 3 create index sec_sem on section using btree(semester);

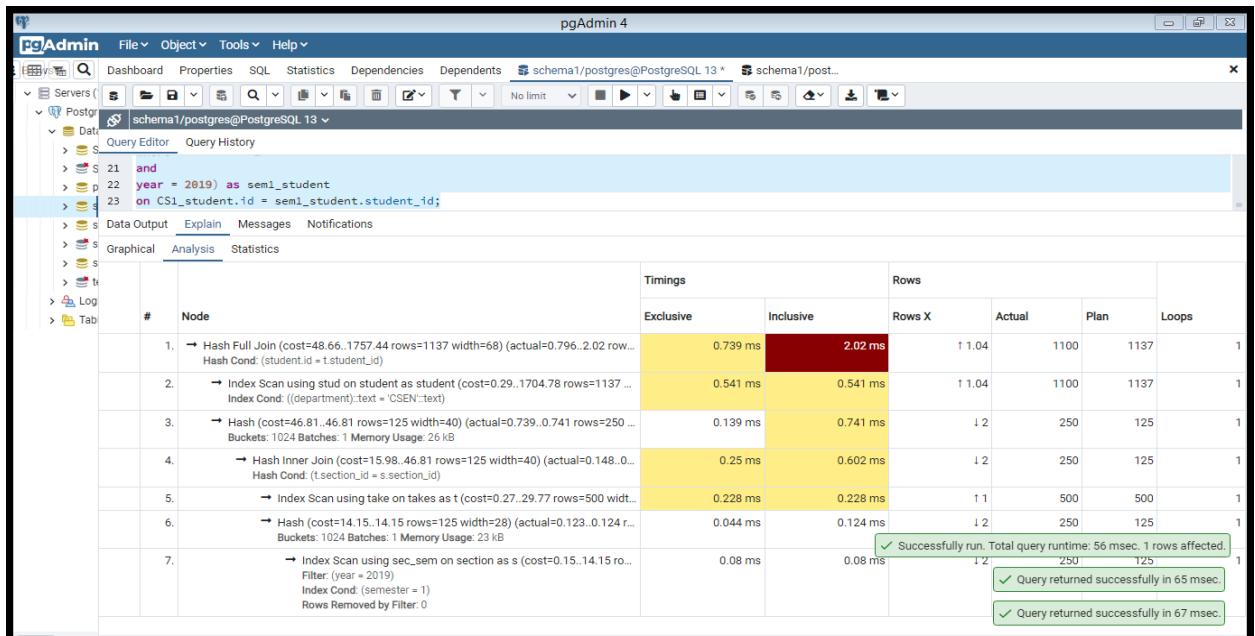
Data Output Explain Messages Notifications
> S QUERY PLAN
> S text
1 Hash Full Join (cost=48.66..1757.44 rows=1137 width=68) (actual time=0.498..1.695 rows=1100 loops=1)
   > S t
      2 [...]Hash Cond: (student.id = t.student_id)
         3 [...]> Index Scan using stud on student (cost=0.29..1704.78 rows=1137 width=28) (actual time=0.043..0.539 rows=1100 loops=1)
            4 [...]Index Cond: (department.text = CSEN:text)
            5 [...]> Hash (cost=46.81..46.81 rows=125 width=40) (actual time=0.444..0.446 rows=250 loops=1)
            6 [...]Buckets: 1024 Batches: 1 Memory Usage: 26kB
            7 [...]> Hash Join (cost=15.98..46.81 rows=125 width=40) (actual time=0.123..0.381 rows=250 loops=1)
            8 [...]Hash Cond: (t.section_id = s.section_id)
            9 [...]> Index Scan using takes_t on takes t (cost=0.27..29.77 rows=500 width=12) (actual time=0.014..0.132 rows=500 loops=1)
           10 [...]> Hash (cost=14.15..14.15 rows=125 width=28) (actual time=0.099..0.100 rows=250 loops=1)
           11 [...]Buckets: 1024 Batches: 1 Memory Usage: 23kB
           12 [...]> Index Scan using sec_sem on section s (cost=0.15..14.15 rows=125 width=28) (actual time=0.011..0.064 rows=250 loops=1)
           13 [...]Index Cond: (semester = 1)
           14 [...]Filter: (year = 2019)
           15 Planning Time: 0.432 ms
           16 Execution Time: 1.854 ms
  
```

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

Query returned successfully in 65 msec.

Query returned successfully in 67 msec.

Execution plan and costs:



Here the B+ tree optimized the query as we put it on the columns that we search on like semester and year of section as B+ tree search in O(logn) time so it is better than seqscan

Query 1 using hash indecies on : Student(department) , section(year) , section(semester), takes(section_id)

Disabled flags: seqscan, bitmapscan

Planning and execution time:

```

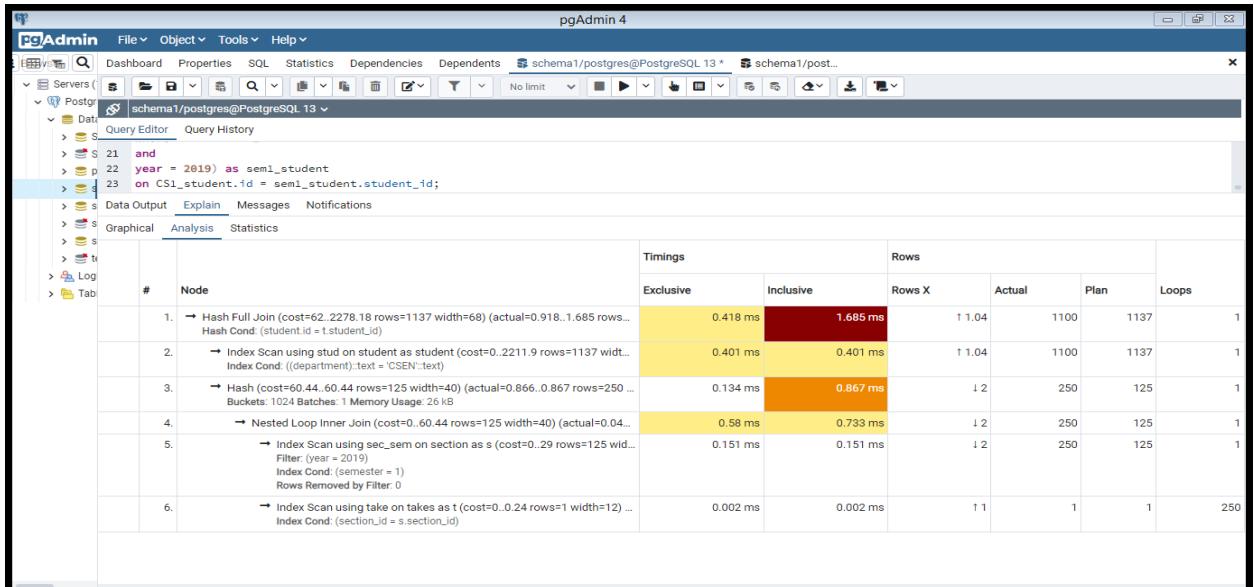
pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents schema1/postgres@PostgreSQL_13 * schema1/post...
Servers (1) Postgr Data (1) schema1/postgres@PostgreSQL_13 * Query Editor Query History
> S 21 and
> S 22 year = 2019) as sem1_student
> S 23 on CS1.student.id = sem1_student.student_id;
> S Data Output Explain Messages Notifications
> S QUERY PLAN
> S text
> S t1
1 Hash Full Join (cost=62.00..2278.18 rows=1137 width=68) (actual time=0.550..1.585 rows=1100 loops=1)
2 [.] Hash Cond: (student.id = t.student_id)
3 [.]> Index Scan using stud on student (cost=0.00..2211.90 rows=1137 width=28) (actual time=0.007..0.544 rows=1100 loops=1)
4 [.]> Index Cond: ((department).text = 'CSEN'.text)
5 [.]> Hash (cost=60.44..60.44 rows=125 width=40) (actual time=0.538..0.540 rows=250 loops=1)
6 [.] Buckets: 1024 Batches: 1 Memory Usage: 26kB
7 [.]> Nested Loop (cost=0.00..60.44 rows=125 width=40) (actual time=0.018..0.456 rows=250 loops=1)
8 [.]> Index Scan using sec_sem on section s (cost=0.00..29.00 rows=125 width=28) (actual time=0.011..0.081 rows=250 loops=1)
9 [.]> Index Cond: (semester = 1)
10 [.]> Filter: (year = 2019)
11 [.]> Index Scan using takes t (cost=0.00..0.24 rows=1 width=12) (actual time=0.001..0.001 rows=1 loops=250)
12 [.]> Index Cond: (section_id = s.section_id)
13 Planning Time: 0.381 ms
14 Execution Time: 1.680 ms

```

Successfully run. Total query runtime: 54 msec. 14 rows affected.

Successfully run. Total query runtime: 53 msec. 14 rows affected.

Execution plan and costs:



Here the hash index increased performance as it search in O(1) time so it helped in searching from semester=1 and year=2019

Query1 using BRIN indecies on : Student(Department) , section(year) , section(semester), takes(section_id)

Disabled flags : seqscan

Planning and execution time :

```

pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents schema1/postgres@PostgreSQL 13 * schema1/post...
Servers (1) Data (1) Log (1) Tables (1)
schema1/postgres@PostgreSQL 13
Query Editor Query History
S 27 and
S 28 year = 2019) as sem1_student
P 29 on CS1_student.id = sem1_student.student_id;
Data Output Explain Messages Notifications
QUERY PLAN
text
1. ...> Bitmap Index Scan on student (cost=12.35..899.98 rows=1137 width=28) (actual time=0.050..5.242 rows=1100 loops=1)
   4. ...> Recheck Cond: ((department)=text AND CSEN=text)
   5. ...> Rows Removed by Index Recheck: 15436
   6. ...> Heap Blocks: lossy=128
   7. ...> Bitmap Index Scan on stud (cost=0.00..12.07 rows=27730 width=0) (actual time=0.042..0.043 rows=1280 loops=1)
   8. ...> Index Cond: ((department)=text AND CSEN=text)
   9. ...> Hash (cost=58.85..58.85 rows=125 width=40) (actual time=0.526..0.528 rows=250 loops=1)
  10. ...> Buckets: 1024 Batches: 1 Memory Usage: 26kB
  11. ...> Hash Join (cost=25.40..58.85 rows=125 width=40) (actual time=0.246..0.456 rows=250 loops=1)
  12. ...> Hash Cond: (t.section_id = s.section_id)
  13. ...> Index Scan using takes_pkey on takes t (cost=0.27..32.40 rows=500 width=12) (actual time=0.017..0.100 rows=500 loops=1)
  14. ...> Hash (cost=23.56..23.56 rows=125 width=28) (actual time=0.217..0.218 rows=250 loops=1)
  15. ...> Hash (cost=23.56..23.56 rows=125 width=28) (actual time=0.217..0.218 rows=250 loops=1)
  16. ...> Bitmap Heap Scan on section s (cost=12.06..23.56 rows=125 width=28) (actual time=0.041..0.152 rows=250 loops=1)
  17. ...> Recheck Cond: (semester = 1)
  18. ...> Rows Removed by Index Recheck: 250
  19. ...> Heap Blocks: lossy=4
  20. ...> Bitmap Index Scan on sec_sem (cost=0.00..12.03 rows=500 width=0) (actual time=0.028..0.028 rows=40 loops=1)
  22. ...> Index Cond: (semester = 1)
  23. Planning Time: 0.448 ms
  24. Execution Time: 6.282 ms
  
```

Execution plan and costs:

#	Node	Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Hash Full Join (cost=72.76..964.67 rows=1137 width=68) (actual=0.536..6.034 rows=1100 loops=1) Hash Cond: (student.id = t.student_id)	0.392 ms	6.034 ms	↑ 1.04	1100	1137	1
2.	→ Bitmap Heap Scan on student as student (cost=12.35..899.98 rows=1137 width=28) (actual=0.047..0.456 rows=250 loops=1) Recheck Cond: ((department)=text AND CSEN=text) Heap Blocks: exact=0	5.121 ms	5.168 ms	↑ 1.04	1100	1137	1
3.	→ Bitmap Index Scan using stud (cost=0.12..12.07 rows=27730 width=0) (actual=0.047..0.047 rows=1280 loops=1) Index Cond: ((department)=text AND CSEN=text)	0.047 ms	0.047 ms	↑ 21.67	1280	27730	1
4.	→ Hash (cost=58.85..58.85 rows=125 width=40) (actual=0.47..0.474 rows=250 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 26 kB	0.08 ms	0.474 ms	↓ 2	250	125	1
5.	→ Hash Inner Join (cost=25.4..58.85 rows=125 width=40) (actual=0.187..0.394 rows=250 loops=1) Hash Cond: (t.section_id = s.section_id)	0.133 ms	0.394 ms	↓ 2	250	125	1
6.	→ Index Scan using takes_pkey on takes t (cost=0.27..32.40 rows=500 width=12) (actual=0.097..0.097 ms rows=500 loops=1)	0.097 ms	0.097 ms	↑ 1	500	500	1
7.	→ Hash (cost=23.56..23.56 rows=125 width=28) (actual=0.162..0.165 rows=250 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 23 kB	0.043 ms	0.165 ms	↓ 2	250	125	1
8.	→ Bitmap Heap Scan on section s (cost=12.06..23.56 rows=125 width=28) (actual=0.092..0.123 ms rows=250 loops=1) Recheck Cond: (year = 2019) Rows Removed by Filter: 0 Recheck Cond: (semester = 1) Heap Blocks: exact=0	0.092 ms	0.123 ms	↓ 2	250	125	1
9.	→ Bitmap Index Scan using sec_sem (cost=0..12.03 rows=500 width=0) (actual=0.031..0.031 ms rows=1 loops=1) Index Cond: (semester = 1)	0.031 ms	0.031 ms				

Successfully run. Total query runtime: 55 msec. 1 rows affected.

The BRIN index here increased performance a little bit as BRIN is used in small range queries in big ranges so it does not help that much in exact value queries

Query1 using mixed indecies : Hash indecies on : Student(Department),section(year) and B+ tree on : section(semester) , takes(section_id)

Disabled flags: seqscan and bitmapscan

Planning and execution time:

The screenshot shows the pgAdmin 4 interface with the title bar "pgAdmin 4". The left sidebar shows a tree view of servers and databases, with "PostgreSQL" selected. The main area is a "Query Editor" showing the following SQL query:

```
> S 24      and
> S 25      year = 2019) as sem1_student
> S 26      on CS1_student.id = sem1_student.student_id;
> S 27      Data Output Explain Messages Notifications
> S 28      QUERY PLAN
> S 29      text
1 Hash Full Join (cost=50.17..2266.35 rows=1137 width=68) (actual time=0.456..1.195 rows=1100 loops=1)
2 (...) Hash Cond: (student.id = t.student_id)
3 (...) Index Scan using std on student (cost=0.00..2211.90 rows=1137 width=28) (actual time=0.012..0.362 rows=1100 loops=1)
4 (...) Index Cond: ((department).text ~ 'CSEN%'::text)
5 (...) > Hash (cost=48.61..48.61 rows=125 width=40) (actual time=0.430..0.432 rows=250 loops=1)
6 (...) Buckets: 1024 Batches: 1 Memory Usage: 26kB
7 (...) > Hash Join (cost=15.98..48.61 rows=125 width=40) (actual time=0.153..0.362 rows=250 loops=1)
8 (...) Hash Cond: (t.section_id = s.section_id)
9 (...) > Index Scan using take on takes t (cost=0.27..29.77 rows=500 width=12) (actual time=0.031..0.103 rows=500 loops=1)
10 (...) > Hash (cost=14.15..14.15 rows=125 width=28) (actual time=0.107..0.108 rows=250 loops=1)
11 (...) Buckets: 1024 Batches: 1 Memory Usage: 23kB
12 (...) > Index Scan using sec_sem on section s (cost=0.15..14.15 rows=125 width=28) (actual time=0.018..0.073 rows=250 loops=1)
13 (...) Index Cond: (semester = 1)
14 (...) Filter: (year = 2019)
15 Planning Time: 0.526 ms
16 Execution Time: 1.333 ms
```

The "Explain" tab is selected, displaying the query plan with 16 numbered steps. The plan starts with a Hash Full Join, followed by various Hash Cond, Hash, and Index Scan operations across multiple tables: student, department, takes, section, and sec_sem. A filter condition for the year 2019 is applied at step 14. The execution time is 1.333 ms.

Execution plan and costs:

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

- Servers:** PostgreSQL 13*
- Current Database:** schema1/postgres@PostgreSQL 13*
- Query Editor:** Contains the following SQL query:


```
s 24 and
s 25 year = 2019) as sem1_student
s 26 on CS1_student.id = sem1_student.student_id;
```
- Analysis Tab:** Selected tab, showing the execution plan.
- Plan Details:**

Node	Timings		Rows			Loops
	Exclusive	Inclusive	Rows X	Actual	Plan	
1. → Hash Full Join (cost=50.17..2266.35 rows=1137 width=68) (actual=1.001..1.948 ro... Hash Cond: (student_id = t.student_id)	0.523 ms	1.948 ms	1 1.04	1100	1137	1
2. → Index Scan using stud on student as student (cost=0.2211.9 rows=1137 width=... Index Cond: ((department::text ~ CSEN::text))	0.514 ms	0.514 ms	1 1.04	1100	1137	1
3. → Hash (cost=48.61..48.61 rows=125 width=40) (actual=0.909..0.911 rows=250 ... Buckets: 1024 Batches: 1 Memory Usage: 26 kB	0.162 ms	0.911 ms	1 2	250	125	1
4. → Hash Inner Join (cost=15.98..48.61 rows=125 width=40) (actual=0.285..0... Hash Cond: (t.section_id ~ s.section_id)	0.268 ms	0.75 ms	1 2	250	125	1
5. → Index Scan using take on takes as t (cost=0.27..29.77 rows=500 width=... Index Cond: (year > 2019)	0.248 ms	0.248 ms	1 1	500	500	1
6. → Hash (cost=14.15..14.15 rows=125 width=28) (actual=0.232..0.234 r... Buckets: 1024 Batches: 1 Memory Usage: 23 kB	0.089 ms	0.234 ms	1 2	250	125	1
7. → Index Scan using sec_sem on section as s (cost=0.15..14.15 ro... Filter: (year > 2019) Index Cond: (semester > 1) Rows Removed by Filter: 0	0.146 ms	0.146 ms	1 2	250	125	1

Both B+ tree and hash increased performance as the search in $O(\log n)$ and $O(1)$ time

Query1 after optimization :

```
select * from  
((select * from section where year=2019 and semester=1) as s inner join takes t on t.section_id =  
s.section_id) as sem1  
full outer join  
(select * from student where department = 'CSEN') as CS1_student on CS1_student.id =  
sem1.student_id;
```

We pushed down selections

Query1 Optimized with no index:

Planning and execution time:

The screenshot shows the pgAdmin 4 interface with the title bar "pgAdmin 4". The left sidebar has "Servers" expanded, showing "PostgreSQL" and "Data". Under "Data", "Query Editor" is selected. The main pane displays the following SQL query and its execution plan:

```
explain analyze select * from
  (select * from section where year=2019 and semester=1) as s inner join takes t on t.section_id = s.section_id) as sem1
full outer join

a Data Output Explain Messages Notifications
```

QUERY PLAN

Step	Operation	Cost	Rows	Width	Actual Time
1	Hash Full Join	(cost=23.95..1394.23)	rows=1137	width=68	(actual time=0.430..12.453 rows=1100 loops=1)
2	[...] Hash Cond: (student_id = t.student_id)				
3	[...] Seq Scan on student (cost=0.00..1365.00)	rows=1137	width=28		(actual time=0.012..11.629 rows=1100 loops=1)
4	[...] Filter: ((department::text = 'CSEN)::text)				
5	[...] Rows Removed by Filter: 64900				
6	[...] Hash (cost=22.39..22.39)	rows=125	width=40		(actual time=0.408..0.412 rows=250 loops=1)
7	[...] Buckets: 1024 Batches: 1 Memory Usage: 26kB				
8	[...] Hash Join (cost=13.06..22.39)	rows=125	width=40		(actual time=0.154..0.339 rows=250 loops=1)
9	[...] Hash Cond: (taction_id = section.section_id)				
10	[...] Seq Scan on takes t (cost=0.00..8.00 rows=500 width=12)	rows=500	width=12		(actual time=0.020..0.063 rows=500 loops=1)
11	[...] Hash (cost=11.50..11.50)	rows=125	width=28		(actual time=0.124..0.126 rows=250 loops=1)
12	[...] Buckets: 1024 Batches: 1 Memory Usage: 23kB				
13	[...] Seq Scan on section (cost=0.00..11.50 rows=125 width=28)	rows=125	width=28		(actual time=0.014..0.089 rows=250 loops=1)
14	[...] Filter: ((year = 2019) AND (semester = 1))				
15	[...] Rows Removed by Filter: 250				
16	Planning Time: 0.355 ms				
17	Execution Time: 12.546 ms				

At the bottom right, a green checkmark indicates: "Query returned successfully in 68 msec."

Execution plan and costs :

The screenshot shows the pgAdmin 4 interface with a query editor containing the following SQL code:

```

full outer join
(select * from student where department = 'CSEN') as CS1_student on CS1_student.id = sem1.student_id;

```

The query plan is displayed in a table format:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Hash Full Join (cost=23.95..1394.23 rows=1137 width=68) (actual=0.396..14.255 rows=1... Hash Cond: (student.id = tstudent_id)	0.394 ms	14.255 ms	↑ 1.04	1100	1137	1
2.	→ Seq Scan on student as student (cost=0..1.1366 rows=1137 width=28) (actual=0.009.... Filter: ((department::text = 'CSEN'::text) Rows Removed by Filter: 64900	13.483 ms	13.483 ms	↑ 1.04	1100	1137	1
3.	→ Hash (cost=22.39..22.39 rows=125 width=40) (actual=0.376..0.379 rows=250 loops=1... Buckets: 1024 Batches: 1 Memory Usage: 26 kB	0.069 ms	0.379 ms	↓ 2	250	125	1
4.	→ Hash Inner Join (cost=13.06..22.39 rows=125 width=40) (actual=0.145..0.31 ro... Hash Cond: (tsection_id = section.section_id)	0.135 ms	0.31 ms	↓ 2	250	125	1
5.	→ Seq Scan on takes as t (cost=0..0.8 rows=500 width=12) (actual=0.018..0.05....	0.056 ms	0.056 ms	↑ 1	500	500	1
6.	→ Hash (cost=11.5..11.5 rows=125 width=28) (actual=0.117..0.119 rows=250.... Buckets: 1024 Batches: 1 Memory Usage: 23 kB	0.033 ms	0.119 ms	↓ 2	250	125	1
7.	→ Seq Scan on section as section (cost=0..11.5 rows=125 width=28) (act... Filter: ((year = 2019) AND (semester < 1)) Rows Removed by Filter: 250	0.086 ms	0.086 ms	↓ 2	250	125	1

At the bottom right, there are two status messages: "Successfully run. Total query runtime: 82 msec. 1 rows affected." and "Query returned successfully in 68 msec."

Query1 optimized using B+ tree on : Student(department) , section(year) , section(semester)

Disabled flags: seqscan and bitmapscan

Planning and execution time:

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

- Servers:** PostgreSQL 13 (selected)
- Databases:** Shopping, Shopping_test, postgres, schema1, schema2, schema3, schema4, test, Login/Group Roles, Tablespaces.
- Query Editor:** Contains the following SQL code:


```
11 full outer join
12   (select * from student where department = 'CSEN') as CS1_student on CS1_student.id = sem1.student_id;
13
```
- Explain Plan:** Shows the execution plan for the query, including:
 - Hash Full Join (cost=51.29..1760.06 rows=1137 width=68) (actual time=0.915..2.159 rows=1100 loops=1)
 - Hash Cond: (student.id = t.student_id)
 - Index Scan using student_dept on student (cost=0.29..1704.78 rows=1137 width=28) (actual time=0.045..0.557 rows=1100 loops=1)
 - Index Cond: ((department)=text + 'CSEN'=text)
 - Hash (cost=49.43..49.43 rows=125 width=40) (actual time=0.856..0.858 rows=250 loops=1)
 - Buckets: 1024 Batches: 1 Memory Usage: 26kB
 - Hash Join (cost=1.59..49.43 rows=125 width=40) (actual time=0.255..0.725 rows=250 loops=1)
 - Hash Cond: (t.section_id = section.section_id)
 - Index Scan using takes_pkey on takes (cost=0.27..32.40 rows=500 width=12) (actual time=0.024..0.247 rows=500 loops=1)
 - Hash (cost=14.15..14.15 rows=125 width=28) (actual time=0.216..0.217 rows=250 loops=1)
 - Buckets: 1024 Batches: 1 Memory Usage: 23kB
 - Index Scan using sec_sem on section (cost=0.15..14.15 rows=125 width=28) (actual time=0.018..0.140 rows=250 loops=1)
 - Hash Cond: (semester = 1)
 - Filter: (year = 2019)
 - Planning Time: 0.587 ms
 - Execution Time: 2.345 ms

Execution plan and costs:

The screenshot shows the pgAdmin 4 interface with the Explain Analysis tab selected, displaying the following execution plan details:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Hash Full Join (cost=51.29..1760.06 rows=1137 width=68) (actual=0.515..1.113 rows=1100)	0.423 ms	1.113 ms	↑ 1.04	1100	1137	1
2.	→ Index Scan using student_dept on student as student (cost=0.29..1704.78 rows=1137 width=28)	0.216 ms	0.216 ms	↑ 1.04	1100	1137	1
3.	→ Hash (cost=49.43..49.43 rows=125 width=40) (actual=0.474..0.475 rows=250 loops=1)	0.085 ms	0.475 ms	↓ 2	250	125	1
4.	→ Hash Inner Join (cost=15.98..49.43 rows=125 width=40) (actual=0.13..0.39 rows=250)	0.164 ms	0.39 ms	↓ 2	250	125	1
5.	→ Hash (cost=14.15..14.15 rows=125 width=28) (actual=0.102..0.103 rows=250)	0.124 ms	0.124 ms	↑ 1	500	500	1
6.	→ Hash (cost=14.15..14.15 rows=125 width=28) (actual=0.102..0.103 rows=250)	0.035 ms	0.103 ms	↓ 2	250	125	1
7.	→ Index Scan using sec_sem on section as section (cost=0.15..14.15 rows=125 width=28)	0.068 ms	0.068 ms	↓ 2	250	125	1

Query1 optimized using hash indecies on : Student(department) , section(year) , section(semester)

Disabled flags: seqscan and bitmapscan

Planning and execution time:

```

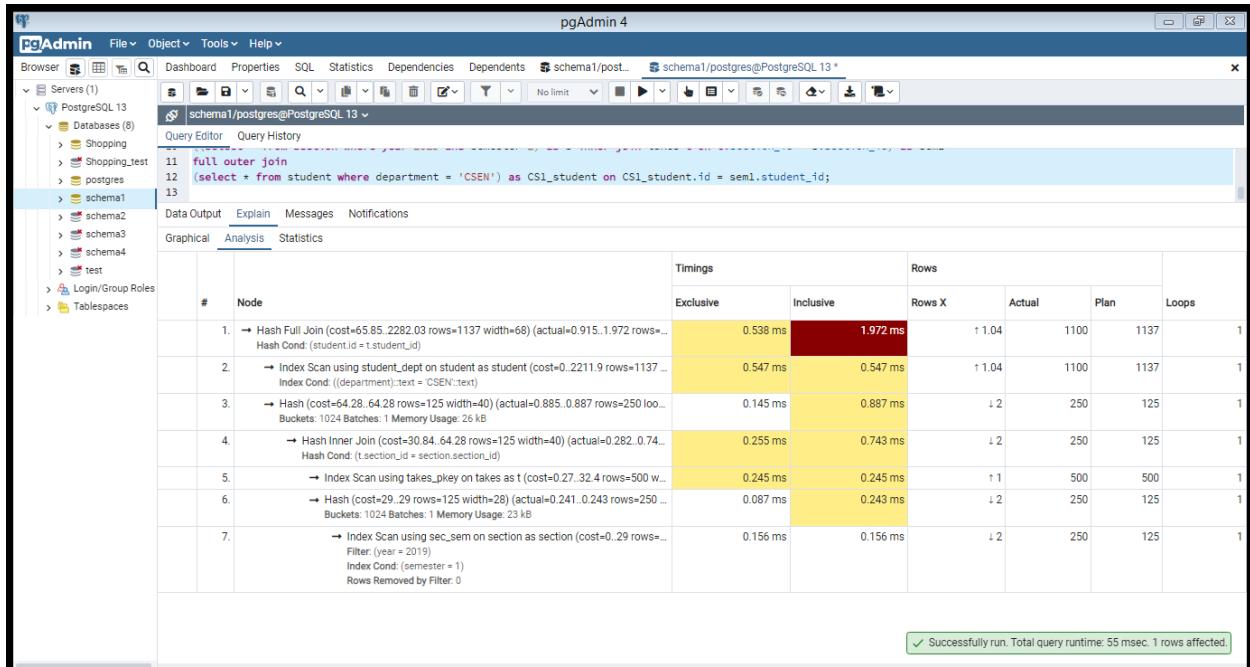
pgAdmin 4
File Object Tools Help
Servers (1) PostgreSQL 13
Databases (8)
> Shopping
> Shopping_test
> postgres
> schema1
> schema2
> schema3
> schema4
> test
Login/Group Roles
Tablespaces
schema1/postgres@PostgreSQL 13
Query Editor Query History
11 full outer join
12 (select * from student where department = 'CSEN') as CS1_student on CS1_student.id = sem1.student_id;
13
Data Output Explain Messages Notifications
QUERY PLAN
text
1 Hash Full Join (cost=65.85..2282.03 rows=1137 width=68) (actual time=0.516..1.241 rows=1100 loops=1)
2 [.] Hash Cond: (student.id = t.student_id)
3 [.]> Index Scan using student_dept on student as student (cost=0.00..2211.90 rows=1137 width=28) (actual time=0.009..0.360 rows=1100 loops=1)
4 [.] Index Cond: ((department)=text)
5 [.]> Hash (cost=64.28..64.28 rows=125 width=40) (actual time=0.492..0.493 rows=250 loops=1)
6 [.] Buckets: 1024 Batches: 1 Memory Usage: 26kB
7 [.]> Hash Join (cost=30.84..64.28 rows=125 width=40) (actual time=0.188..0.419 rows=250 loops=1)
8 [.] Hash Cond: (t.section_id = section.section_id)
9 [.]> Index Scan using takes_pkey on takes as t (cost=0.27..32.40 rows=500 width=12) (actual time=0.026..0.124 rows=500 loops=1)
10 [.]> Hash (cost=29.00..29.00 rows=125 width=28) (actual time=0.148..0.149 rows=250 loops=1)
11 [.] Buckets: 1024 Batches: 1 Memory Usage: 23kB
12 [.]> Index Scan using sec_sem on section as section (cost=0.00..29.00 rows=125 width=28) (actual time=0.014..0.095 rows=250 loops=1)
13 [.] Index Cond: (semester = 1)
14 [.] Filter: (year = 2019)
15 Planning Time: 0.393 ms
16 Execution Time: 1.417 ms

```

Successfully run. Total query runtime: 47 msec. 16 rows affected.

Successfully run. Total query runtime: 51 msec. 16 rows affected.

Execution plan and costs:



Query1 optimized using BRIN indecies on : Student(Department) , section(year) , section(semester),takes(section_id)

Disabled flags : seqscan

Planning and execution time:

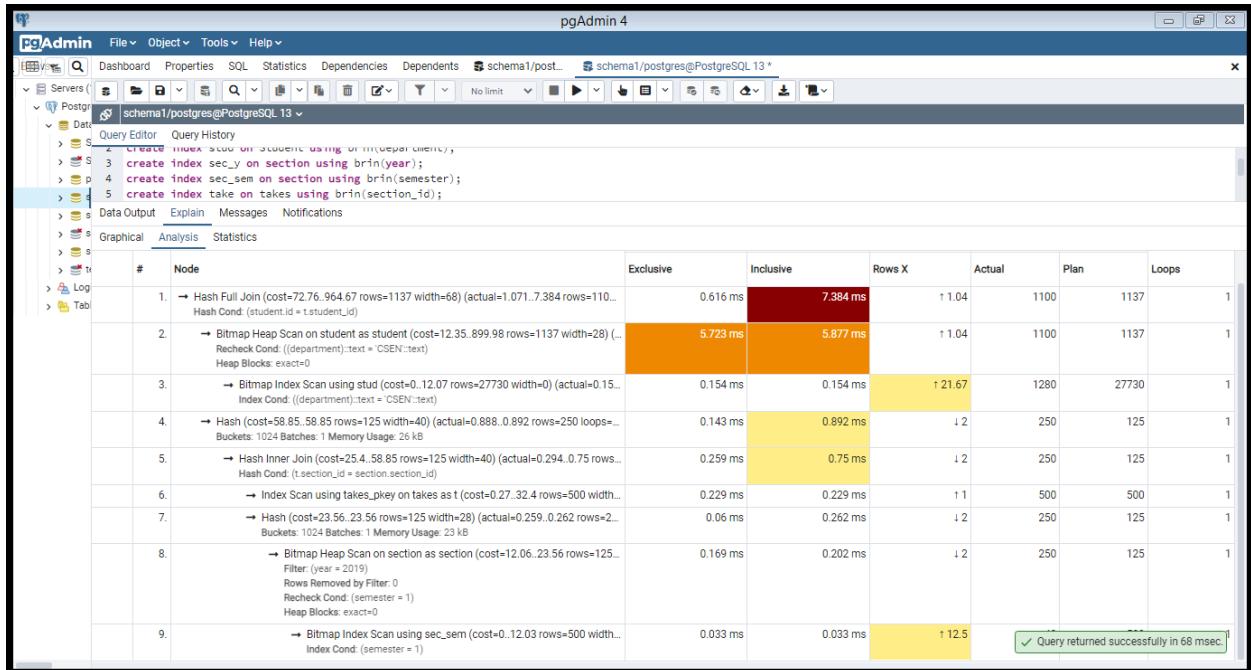
```

pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents schema1/post...
Servers ( PostgreSQL schema1/postgres@PostgreSQL 13
Data
Query Editor Query History
S full outer join
  8 (select * from student where department = 'CSEN') as CS1_student on CS1_student.id = sem1.student_id;
  9
  10
  11
  12
  13
  14
  15
  16
  17
  18
  19
  20
  21
  22
  23 Planning Time: 0.695 ms
  24 Execution Time: 4.798 ms
  ✓ Query returned successfully in 68 msec.

Data Output Explain Messages Notifications
QUERY PLAN
text
3 [...]--> Bitmap Heap Scan on student (cost=12.35..899.98 rows=1137 width=28) (actual_time=0.058..3.627 rows=1100 loops=1)
4 [...] Recheck Cond: ((department).text = 'CSEN'.text)
5 [...] Rows Removed by Index Recheck: 15438
6 [...] Heap Blocks: lossy=128
7 [...]--> Bitmap Index Scan on stud (cost=0.00..12.07 rows=27730 width=0) (actual_time=0.049..0.050 rows=1280 loops=1)
8 [...] Index Cond: ((department).text = 'CSEN'.text)
9 [...]--> Hash (cost=58.85..58.85 rows=125 width=40) (actual_time=0.541..0.544 rows=250 loops=1)
10 [...] Buckets: 1024 Batches: 1 Memory Usage: 24kB
11 [...]--> Hash Join (cost=25.40..58.85 rows=125 width=40) (actual_time=0.185..0.450 rows=250 loops=1)
12 [...] Hash Cond: (t.section_id = section.section_id)
13 [...]--> Index Scan using takes_pkey on takes (cost=0.27..32.40 rows=500 width=12) (actual_time=0.018..0.131 rows=500 loops=1)
14 [...]--> Hash (cost=23.56..23.56 rows=125 width=28) (actual_time=0.156..0.158 rows=250 loops=1)
15 [...] Buckets: 1024 Batches: 1 Memory Usage: 23kB
16 [...]--> Bitmap Heap Scan on section (cost=12.06..23.56 rows=125 width=28) (actual_time=0.037..0.124 rows=250 loops=1)
17 [...] Recheck Cond: (semester = 1)
18 [...] Rows Removed by Index Recheck: 250
19 [...] Filter: (year = 2019)
20 [...] Heap Blocks: lossy=4
21 [...]--> Bitmap Index Scan on sec_sem (cost=0.00..12.03 rows=500 width=0) (actual_time=0.027..0.027 rows=40 loops=1)
22 [...] Index Cond: (semester = 1)
23 Planning Time: 0.695 ms
24 Execution Time: 4.798 ms
  ✓ Query returned successfully in 68 msec.

```

Execution plan and costs:



Query1 optimized using mixed indecies :

Hash indecies on : Student(Department),section(year) and B+ tree on : section(semester),takes(section_id)

Disabled flags : seqscan and bitmapscan

Planning and execution time:

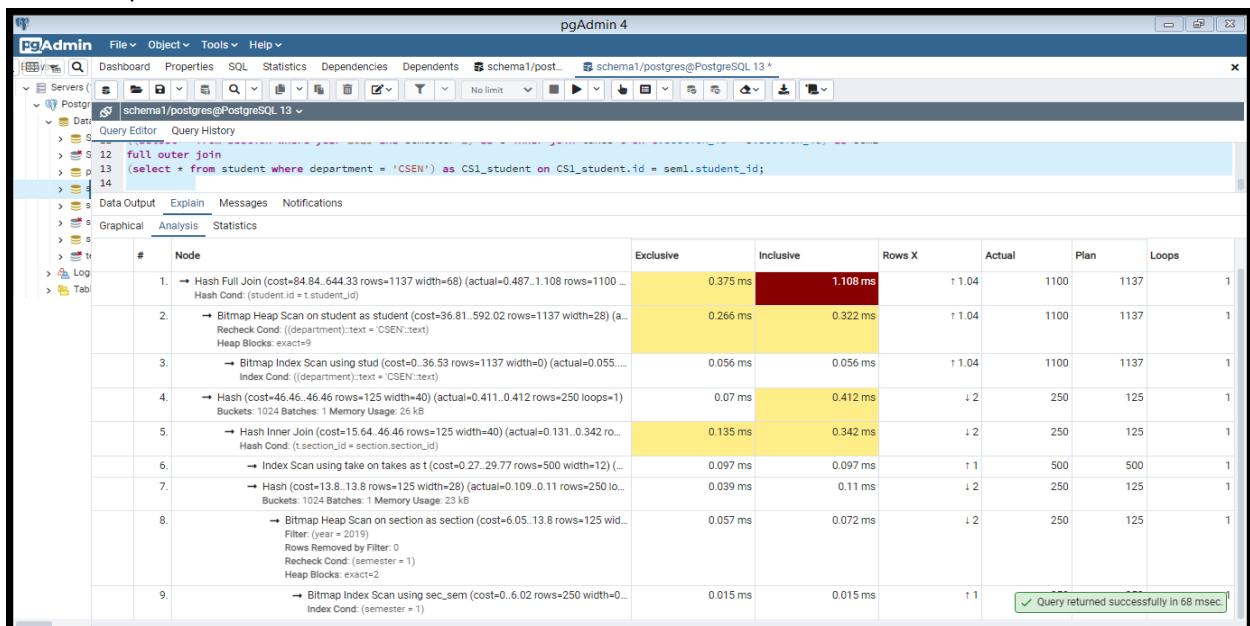
```

pgAdmin 4
File ▾ Object ▾ Tools ▾ Help ▾
Dashboard Properties SQL Statistics Dependencies Dependents schema1/post... schema1/postgres@PostgreSQL 13 *
Servers (1) Data (1) Query Editor (1) Log (1) Tables (1)
schema1/postgres@PostgreSQL 13 ▾
Data (1) S 12 full outer join (1) p 13 (select * from student where department = 'CSEN') as CS1_student on CS1_student.id = sem1.student_id; (1) t 14
Data Output Explain Messages Notifications
QUERY PLAN
text
1. Hash Full Join (cost=84.84..644.33 rows=1137 width=68) (actual time=1.008..2.034 rows=1100 loops=1)
   2. [...] Hash Cond: (student.id = t.student_id)
   3. [...] Bitmap Heap Scan on student (cost=36.81..592.02 rows=1137 width=28) (actual time=0.117..0.553 rows=1100 loops=1)
      Recheck Cond: ((department).text = 'CSEN'.text)
      Heap Blocks: exact<9
   4. [...] Hash Join (cost=15.64..46.46 rows=125 width=40) (actual time=0.238..0.731 rows=250 loops=1)
      Hash Cond: (t.section_id = section.section_id)
   5. [...] Index Scan using takes_t on takes (cost=0.27..29.77 rows=300 width=12) (actual time=0.022..0.253 rows=500 loops=1)
   6. [...] Hash (cost=13.80..13.80 rows=125 width=28) (actual time=0.200..0.200 rows=250 loops=1)
      Buckets: 1024 Batches: 1 Memory Usage: 23kB
   7. [...] Bitmap Heap Scan on section (cost=6.05..13.80 rows=125 width=28) (actual time=0.068..0.139 rows=250 loops=1)
      Recheck Cond: (semester = 1)
   8. [...] Filter: (year = 2019)
      Heap Blocks: exact<2
   9. [...] Bitmap Index Scan on sec_sem (cost=0.00..6.02 rows=250 width=0) (actual time=0.050..0.050 rows=250 loops=1)
      Index Cond: (semester = 1)
Planning Time: 2.857 ms
Execution Time: 2.245 ms

```

Query returned successfully in 68 msec.

Execution plan and costs:



Query2 with no index:

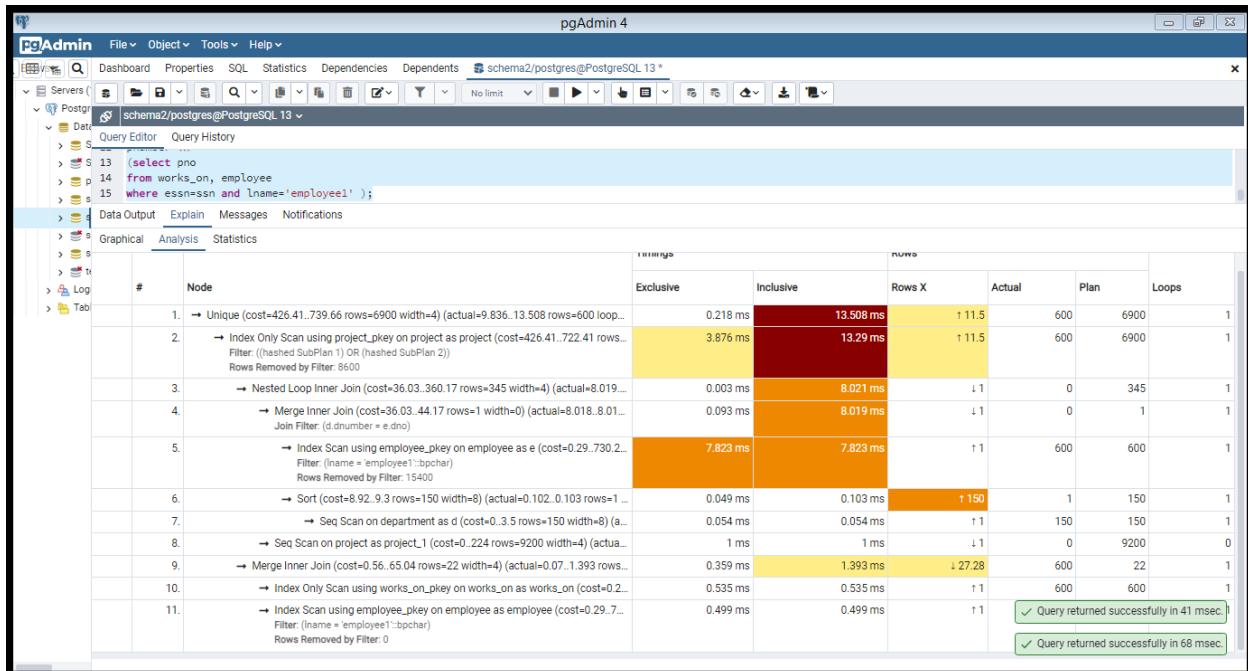
Planning and execution time:

```

explain analyze select distinct pnumber
from project
where pnumber in
    (select pno
     from works_on, employee
      where essn=ssn and lname='employee1' );
  
```

The screenshot shows the pgAdmin 4 interface with the query editor open. The query is displayed in the top pane, and the execution plan is shown in the bottom pane under the 'Explain' tab. The plan is a detailed text representation of the query's execution strategy, including nested loops, merge joins, and various filters and sorts. Two green status bars at the bottom right indicate successful query returns.

Execution plan and costs:



Query2 using B+ tree on : Department(Dnumber),Department(mgr_snn), Project(Pnumber) , Employee(Lname), Employee(dno), works_on(essn)

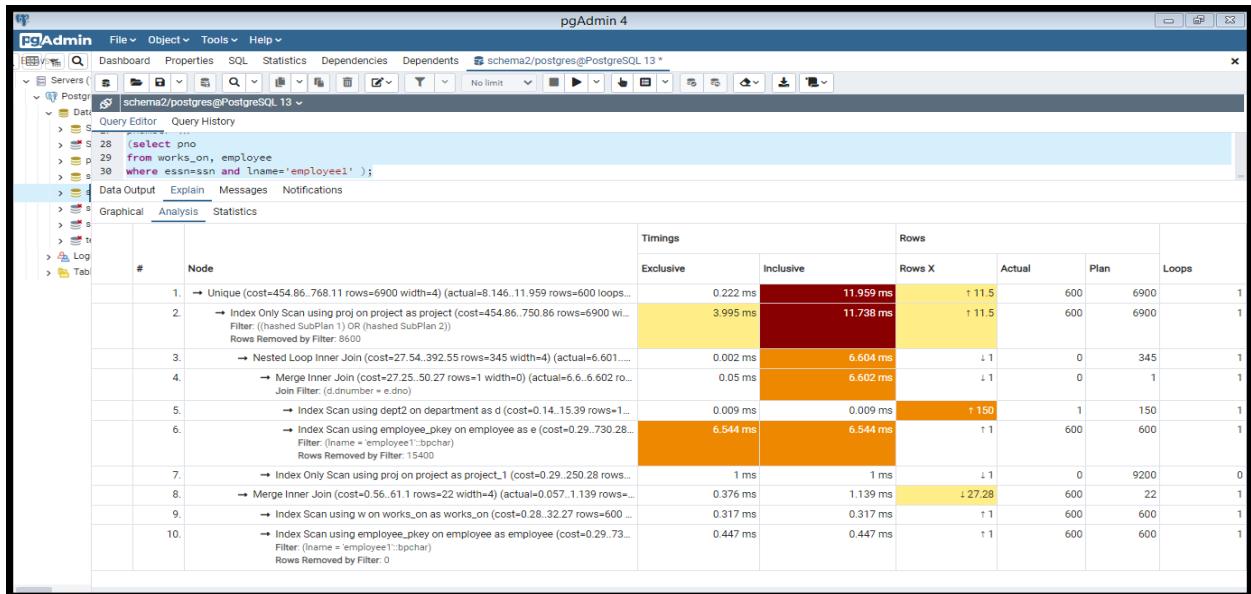
Disabled flags : seqscan , bitmapscan

Planning and execution time:

```

pgAdmin 4
File ▾ Object ▾ Tools ▾ Help ▾
Dashboard Properties SQL Statistics Dependencies Dependents schema2/postgres@PostgreSQL 13*
Servers (1) Postgres (1) Data (1) Query Editor (1) Query History (1)
S 28 (select pno
      from works_on, employee
      where essn=ssn and lname='employee1' );
Data Output Explain Messages Notifications
S 28
  3 (...) Filter: ((hashed SubPlan 1) OR (hashed SubPlan 2))
  4 (...) Rows Removed by Filter: 8600
  5 (...) Heap Fetches: 0
  6 (...) SubPlan 1
  7 (...) > Nested Loop (cost=27.54..392.55 rows=345 width=4) (actual time=4.200..4.201 rows=0 loops=1)
    8 (...) > Merge Join (cost=27.25..50.27 rows=1 width=0) (actual time=4.200..4.201 rows=0 loops=1)
    9 (...) > Merge Cond: (d.mgr_ssn = e.ssn)
    10 (...) > Join Filter: (d.dnumber = e.dno)
    11 (...) > Index Scan using dept2 on department d (cost=0.14..15.39 rows=150 width=8) (actual time=0.008..0.009 rows=1 loops=1)
    12 (...) > Index Scan using employee_pkey on employee e (cost=0.29..730.28 rows=600 width=8) (actual time=0.012..4.145 rows=600 loops=1)
    13 (...) Filter: (lname = 'employee1'.bpchar)
    14 (...) Rows Removed by Filter: 15400
    15 (...) > Index Only Scan using proj on project project_1 (cost=0.29..250.28 rows=9200 width=4) (never executed)
    16 (...) > Merge Join (cost=0.56..61.10 rows=22 width=4) (actual time=0.037..0.528 rows=600 loops=1)
    17 (...) > SubPlan 2
    18 (...) > > Merge Join (cost=0.28..32.27 rows=600 width=8) (actual time=0.022..0.131 rows=600 loops=1)
    19 (...) > > Merge Cond: (works_on.essn = employee.ssn)
    20 (...) > > Index Scan using w on works_on (cost=0.28..32.27 rows=600 width=8) (actual time=0.022..0.131 rows=600 loops=1)
    21 (...) > > Index Scan using employee_pkey on employee (cost=0.29..730.28 rows=600 width=8) (actual time=0.013..0.198 rows=600 loops=1)
    22 (...) Filter: (lname = 'employee1'.bpchar)
    23 Planning Time: 0.841 ms
    24 Execution Time: 8.306 ms
  
```

Execution plan and costs:

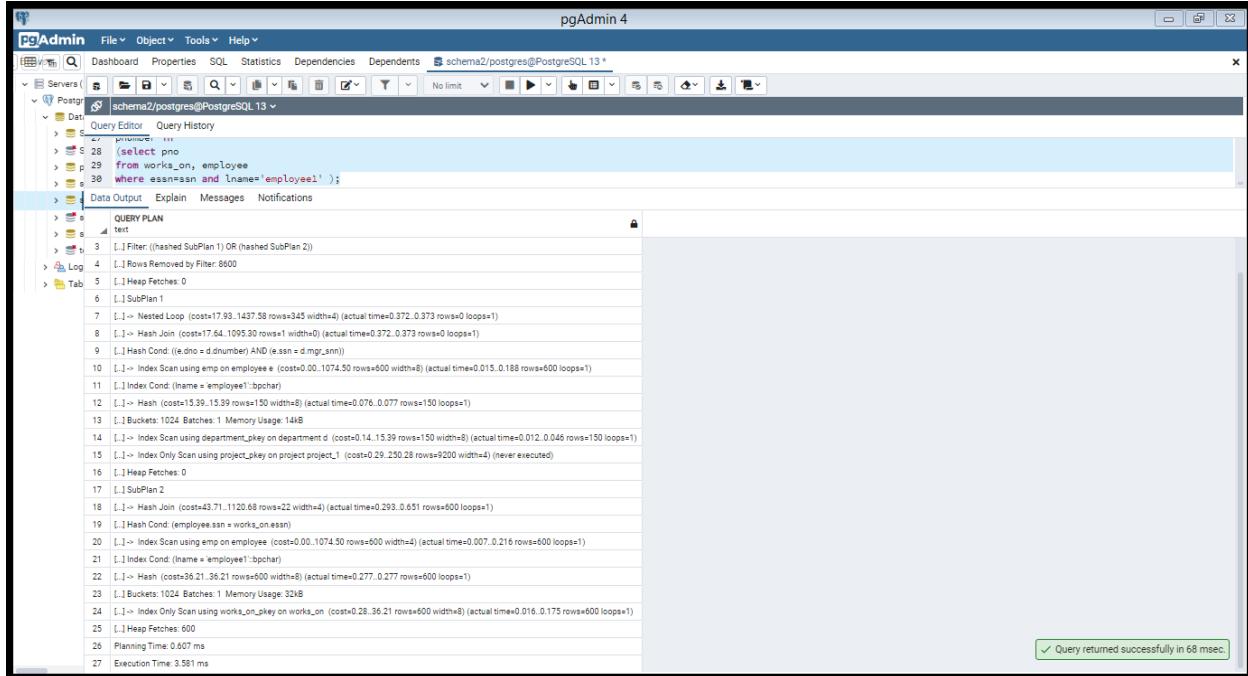


Here the B+ tree helped in the (in) statement because we check every Pnumber also it helped in the join between Employee and department as B+ tree search in O(logn)

Query2 using hash indecies on : Employee(Lname), Employee(dno)

Disabled flags: seqscan,bitmapscan

Planning and execution time:

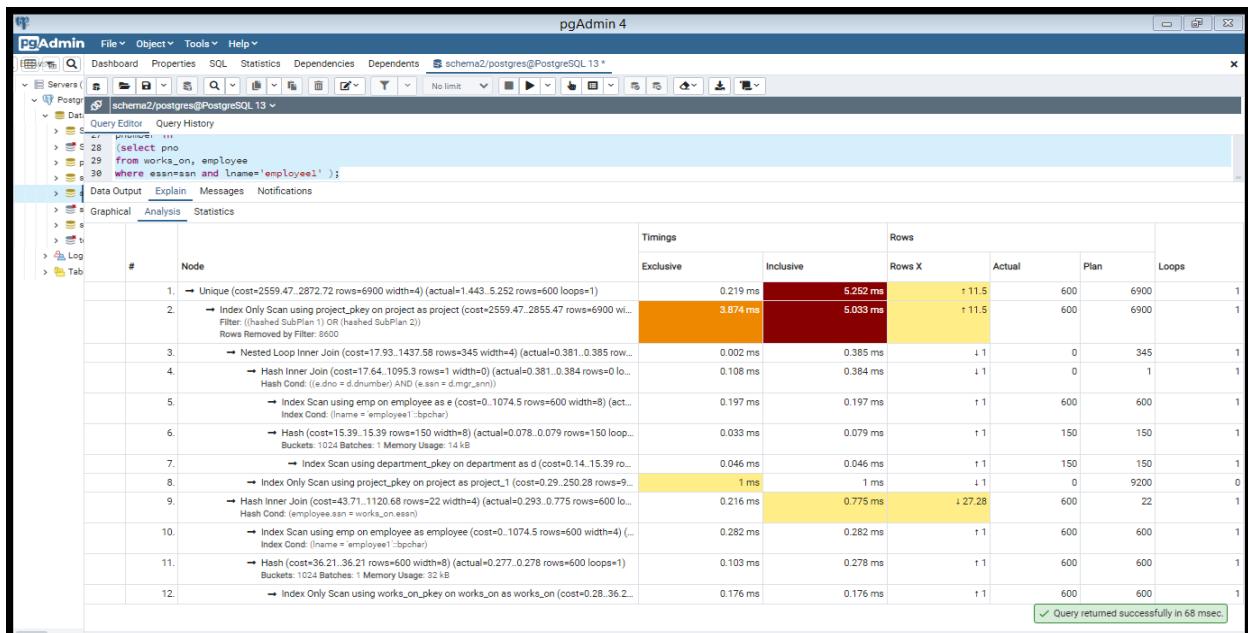


```

pgAdmin 4
File Object Tools Help
Servers PostgreSQL schema2/postgres@PostgreSQL 13+
Query Editor Query History
3.28 (select pno
      from works_on, employee
      where essn=ssn and lname='employee1' );
Data Output Explain Messages Notifications
QUERY PLAN
3.28
  3.28.1 Filter: ((hashed SubPlan 1) OR (hashed SubPlan 2))
  3.28.2 4.1 Rows Removed by Filter: 8600
  3.28.3 5.1 Heap Fetches: 0
  3.28.4 6.1 SubPlan 1
  3.28.5 7.1 Nested Loop (costs=17.93.1437.58 rows=245 width=4) (actual time=0.372.0.373 rows=600 loops=1)
    3.28.5.1 8.1--> Hash Join (costs=17.64.1095.30 rows=1 width=0) (actual time=0.372.0.373 rows=600 loops=1)
      3.28.5.2 9.1--> Hash Cond: ((dno = d.dnumber) AND (essn = dmgr_essn))
      3.28.5.3 10.1--> Index Scan using emp on employee e (cost=0.00.1074.50 rows=600 width=8) (actual time=0.015.0.188 rows=600 loops=1)
        3.28.5.4 11.1--> Index Cond: (lname = employee1_bpchar)
        3.28.5.5 12.1--> Hash (costs=15.39.15.39 rows=150 width=8) (actual time=0.076.0.077 rows=150 loops=1)
        3.28.5.6 13.1--> Hash (costs=15.39.15.39 rows=150 width=8) (actual time=0.076.0.077 rows=150 loops=1)
        3.28.5.7 14.1--> Index Scan using department_pkey on department d (costs=0.14.15.39 rows=150 width=8) (actual time=0.012.0.046 rows=150 loops=1)
        3.28.5.8 15.1--> Index Only Scan using project_pkey on project project_1 (costs=0.29.250.28 rows=9200 width=4) (never executed)
        3.28.5.9 16.1--> Hash Fetches: 0
        3.28.5.10 17.1--> SubPlan 2
        3.28.5.11 18.1--> Hash Join (costs=43.71..1120.68 rows=22 width=4) (actual time=0.293.0.651 rows=600 loops=1)
        3.28.5.12 19.1--> Hash Cond: (employee.ssn = works_on.essn)
        3.28.5.13 20.1--> Index Scan using emp on employee e (costs=0.00.1074.50 rows=600 width=8) (actual time=0.007.0.216 rows=600 loops=1)
        3.28.5.14 21.1--> Index Cond: (lname = employee1_bpchar)
        3.28.5.15 22.1--> Hash (costs=36.21.36.21 rows=600 width=8) (actual time=0.277.0.277 rows=600 loops=1)
        3.28.5.16 23.1--> Hash (costs=15.39.15.39 rows=150 width=8) (actual time=0.076.0.077 rows=150 loops=1)
        3.28.5.17 24.1--> Index Only Scan using works_on_pkey on works_on (costs=0.28.36.21 rows=600 width=8) (actual time=0.016.0.175 rows=600 loops=1)
        3.28.5.18 25.1--> Hash Fetches: 600
        3.28.5.19 26. Planning Time: 0.407 ms
        3.28.5.20 27. Execution Time: 3.561 ms
    
```

Query returned successfully in 68 msec.

Execution plan and costs:



#	Node	Timings		Rows				
		Exclusive	Inclusive	Rows X	Actual	Plan		
1.	Unique (cost=2559.47.2872.72 rows=600 width=4) (actual=1.443.5.252 rows=600 loops=1)		0.219 ms	5.252 ms	11.5	600	6900	1
2.	Index Only Scan using project_pkey on project as project (cost=2559.47..2855.47 rows=6900 width=4) Filter: ((hashed SubPlan 1) OR (hashed SubPlan 2)) Rows Removed by Filter: 8600		3.874 ms	5.033 ms	11.5	600	6900	1
3.	→ Nested Loop Inner Join (cost=17.93.1437.58 rows=345 width=4) (actual=0.381.0.385 rows=150 loops=1)		0.002 ms	0.385 ms	1 1	0	345	1
4.	→ Hash Inner Join (cost=17.64.1095.3 rows=1 width=0) (actual=0.381.0.384 rows=0 loops=1)		0.108 ms	0.384 ms	1 1	0	1	1
5.	→ Index Scan using emp on employee as e (cost=0.00.1074.5 rows=600 width=8) (actual=0.197 ms.0.197 ms rows=600 loops=1)		0.197 ms	0.197 ms	1 1	600	600	1
6.	→ Hash (cost=15.39.15.39 rows=150 width=8) (actual=0.078.0.079 rows=150 loops=1)		0.033 ms	0.079 ms	1 1	150	150	1
7.	→ Index Scan using department_pkey on department as d (cost=0.14.15.39 rows=150 width=8) (actual=0.046 ms.0.046 ms rows=150 loops=1)		0.046 ms	0.046 ms	1 1	150	150	1
8.	→ Index Only Scan using project_pkey on project as project_1 (cost=0.29.250.28 rows=9 rows=9 loops=1)		1 ms	1 ms	1 1	0	9200	0
9.	→ Hash Inner Join (cost=43.71..1120.68 rows=22 width=4) (actual=0.293.0.775 rows=600 loops=1)		0.216 ms	0.775 ms	1 27.28	600	22	1
10.	→ Index Scan using emp on employee as employee (cost=0.00.1074.5 rows=600 width=8) (actual=0.282 ms.0.282 ms rows=600 loops=1)		0.282 ms	0.282 ms	1 1	600	600	1
11.	→ Hash (cost=36.21.36.21 rows=600 width=8) (actual=0.277.0.278 rows=600 loops=1)		0.103 ms	0.278 ms	1 1	600	600	1
12.	→ Index Only Scan using works_on_pkey on works_on as works_on (cost=0.28.36.21 rows=600 width=8) (actual=0.176 ms.0.176 ms rows=600 loops=1)		0.176 ms	0.176 ms	1 1	600	600	1

Query returned successfully in 68 msec.

Here the hash helped in finding the Employee with lname="employee1" as it search in O(1) time

Query2 using BRIN indecies on : Department(Dnumber) , Employee(Lname),Employee(dno)

Disabled flags: seqscan

Planning and execution time:

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

- Server:** PostgreSQL 13
- Database:** schema2/postgres@PostgreSQL 13
- Query Editor:** Contains the following SQL code:

```
1 set enable_seqscan=off;
2 set enable_bitmapscan=on;
3 update pg_index set invalid=false where indexrelid='employee_pkey'::regclass;
```
- Result Set:** Displays the query plan for the update statement, showing various stages of execution including Bitmap Index Scan, Hash Cond, and Index Only Scan.
- Status Bar:** Shows "Query returned successfully in 68 msec."

Execution plan and costs:

The screenshot shows the pgAdmin 4 interface with a query editor window. The query being analyzed is:

```

SELECT p.name, e.name, d.name, j.name, w.hours
FROM project AS p
JOIN employee AS e ON e.id = p.employee_id
JOIN department AS d ON d.id = p.department_id
JOIN job AS j ON j.id = p.job_id
JOIN works_on AS w ON w.project_id = p.id AND w.employee_id = e.id;
    
```

The execution plan (shown in the 'Explain' tab) details the following steps:

1. Unique (cost=1360.96..1674.21 rows=6900 width=400) (actual=7,112.10..226 rows=600 loops=1)
2. → Index Only Scan using project_pkey on project as project (cost=1360.96..1656.96 rows=6900 width=400) (actual=7,112.10..226 rows=600 loops=1)
 Filter: (hashed SubPlan[1]) OR (hashed SubPlan[2])
 Rows Removed by Filter: 2600
3. → Nested Loop Inner Join (cost=30.18..838.33 rows=345 width=400) (actual=2,602..2,609 rows=0 loops=1)
4. → Hash Inner Join (cost=29.89..494.04 rows=1 width=0) (actual=2,602..2,608 rows=0 loops=1)
 Hash Cond: (e.id = d.id) AND (e.id = d.mgr_id AND d.mgr_id IS NOT NULL)
5. → Bitmap Heap Scan on employee as e (cost=12.25..475.25 rows=600 width=400) (actual=0... rows=0 loops=1)
 Recheck Cond: (name = employee.t_bpcar)
 Heap Blocks: exact0
6. → Bitmap Index Scan using emp_idx (cost=0..12.11 rows=16000 width=0) (actual=0..139..0 rows=0 loops=1)
 Index Cond: (name = employee.t_bpcar)
7. → Hash (cost=18.39..15.39 rows=150 width=8) (actual=0..118..0.121 rows=150 loops=1)
 Buckets: 1024 Batches: 1 Memory Usage: 14 kB
8. → Index Scan using department_pkey on department as d (cost=14..15.39 rows=150..150 width=400) (actual=0..150..150 loops=1)
9. → Index Scan using department_pkey on department as d (cost=14..15.39 rows=150..150 width=400) (actual=0..150..150 loops=1)
10. → Hash Inner Join (cost=56.56..521.43 rows=22 width=4) (actual=0..456..4,094 rows=600 loops=1)
 Hash Cond: (employees.ssn = works_on.emp_ssn)
11. → Bitmap Heap Scan on employee as employee (cost=12..25..475.25 rows=600 width=400) (actual=0..11..0 loops=1)
 Recheck Cond: (name = employee.t_bpcar)
 Heap Blocks: exact0
12. → Bitmap Index Scan using emp_idx (cost=0..12.11 rows=16000 width=0) (actual=0..042..0.042 loops=1)
 Index Cond: (name = employee.t_bpcar)
13. → Hash (cost=56.21..36.21 rows=600 width=8) (actual=0..391..0.392 rows=600 loops=1)
 Buckets: 1024 Batches: 1 Memory Usage: 32 kB
14. → Index Only Scan using works_on_pkey on works_on as works_on (cost=0..28..36.21 rows=600 width=400) (actual=0..232..0.232 loops=1)

The 'Timings' section provides detailed performance metrics for each step:

Node	Timings		Rows			Loops
	Exclusive	Inclusive	Rows X	Actual	Plan	
1.	0.221 ms	10.226 ms	111.5	600	6900	1
2.	3.303 ms	10.006 ms	111.5	600	6900	1
3.	0.001 ms	2.609 ms	1..1	0	345	1
4.	0.19 ms	2.608 ms	1..1	0	1	1
5.	2.217 ms	2.357 ms	1..1	600	600	1
6.	0.14 ms	0.14 ms	112.5	1280	16000	1
7.	0.049 ms	0.121 ms	1..1	150	150	1
8.	0.072 ms	0.072 ms	1..1	150	150	1
9.	1 ms	1 ms	1..1	0	9200	0
10.	0.315 ms	4.094 ms	1..27.28	600	22	1
11.	3.346 ms	3.388 ms	1..1	600	600	1
12.	0.042 ms	0.042 ms	112.5	1280	16000	1
13.	0.16 ms	0.392 ms	1..1	600	600	1
14.	0.232 ms	0.232 ms	1..1	60	60	1

A note at the bottom right indicates: "Query returned successfully in 58 msec."

Here the Brin helped in finding the Employee name as it sorted the names and then found the needed name : employee1 as brin is used in small range queries in sorted data

Query2 using mixed indecies :

Disabled flags: seqscan and bitmapscan

Hash index on : Employee(dno)

B+ tree on : Employee(Lname), Project(Pnumber),Department(dnumber)

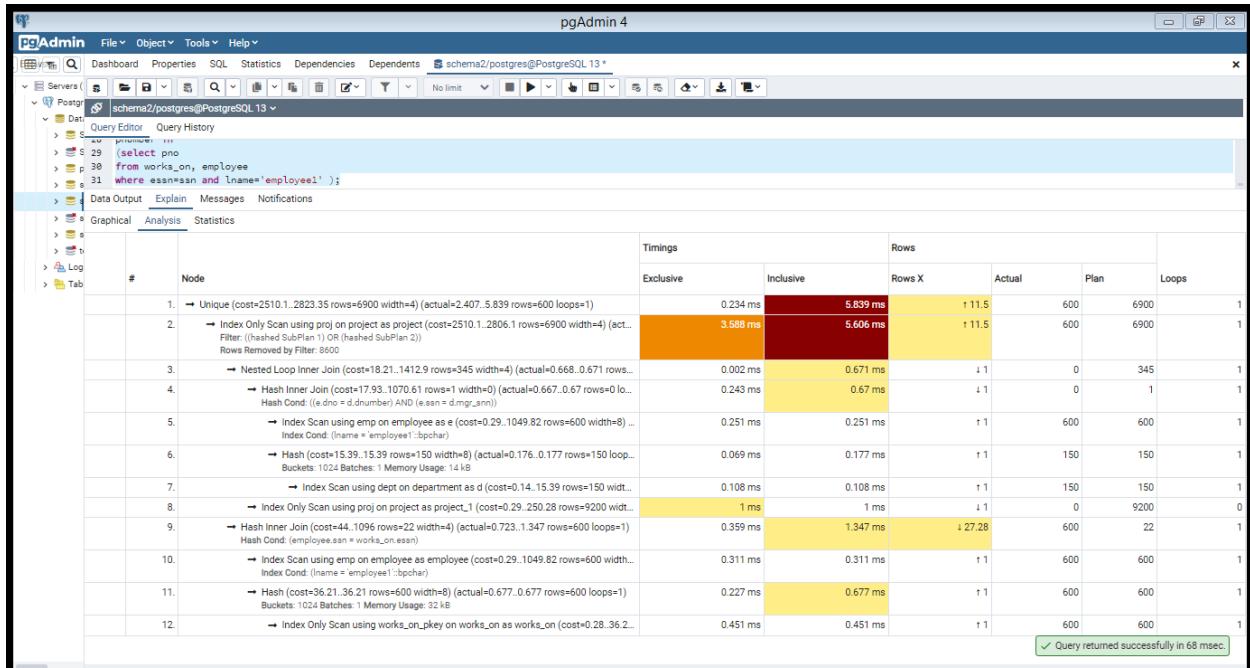
Planning and execution time:

```

pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents schema2/postgres@PostgreSQL 13+
Servers (1) PostgreSQL schema2/postgres@PostgreSQL 13+
Data (1) Query Editor Query History
> S 27 pnumber in
> S 28 pnumber
> p 29 (select pno
> s 30 from works_on, employee
> Data Output Explain Messages Notifications
> s 31
  QUERY PLAN
    text
  6 (...) SubPlan 1
  7 (...) >> Nested Loop (cost=18.21..1412.90 rows=945 width=8) (actual_time=0.362..0.367 rows=0 loops=1)
  8 (...) >> Hash Join (cost=17.93..1070.61 rows=1 width=0) (actual_time=0.362..0.365 rows=0 loops=1)
  9 (...) >> Hash Cond ((e.dno = d.dnumber) AND (e.esn = d.mgr_ssn))
10 (...) >> Index Scan using emp on employee e (cost=0.29..1049.82 rows=500 width=8) (actual_time=0.032..0.135 rows=500 loops=1)
11 (...) Index Cond: (lname = 'employee1'::bpchar)
12 (...) >> Hash (cost=15.39..15.39 rows=150 width=8) (actual_time=0.080..0.082 rows=150 loops=1)
13 (...) Buckets: 1024 Batches: 1 Memory Usage: 14kB
14 (...) >> Index Scan using dept on department d (cost=0.14..15.39 rows=150 width=8) (actual_time=0.015..0.048 rows=150 loops=1)
15 (...) >> Index Only Scan using proj on project project_1 (cost=0.29..250.28 rows=9200 width=4) (never executed)
16 (...) Heap Fetches: 0
17 (...) SubPlan 2
18 (...) >> Hash Join (cost=44.00..1096.00 rows=22 width=4) (actual_time=0.356..0.626 rows=500 loops=1)
19 (...) >> Hash Cond (employee.ssn = works_on.esn)
20 (...) >> Index Scan using emp on employee (cost=0.29..1049.82 rows=500 width=4) (actual_time=0.020..0.115 rows=500 loops=1)
21 (...) >> Hash Cond: (lname = 'employee1'::bpchar)
22 (...) >> Hash (cost=36.21..36.21 rows=500 width=8) (actual_time=0.325..0.326 rows=500 loops=1)
23 (...) Buckets: 1024 Batches: 1 Memory Usage: 32kB
24 (...) >> Index Only Scan using works_on_pkey on works_on (cost=0.28..36.21 rows=500 width=8) (actual_time=0.018..0.220 rows=500 loops=1)
25 (...) Heap Fetches: 600
26 Planning Time: 0.772 ms
27 Execution Time: 4.247 ms
  ✓ Query returned successfully in 68 msec.

```

Execution plan and costs:



Query2 after optimization:

```
select distinct pnumber
from project
where pnumber in
(select pnumber
from ((select Pnumber,Dnumber as p1 from project) as p inner join department d on p.p1=d.Dnumber)
as d inner join (select * from employee e where e.lname='employee1') as e
on e.dno=d.dnumber and d.mgr_snn=e.ssn
)
or
pnumber in
(select pno
from works_on w inner join (select * from employee e where e.lname='employee1') as e
on w.essn=e.ssn );
```

We changed the cross product with inner joins

Query2 optimized with no index:

Planning and execution time :

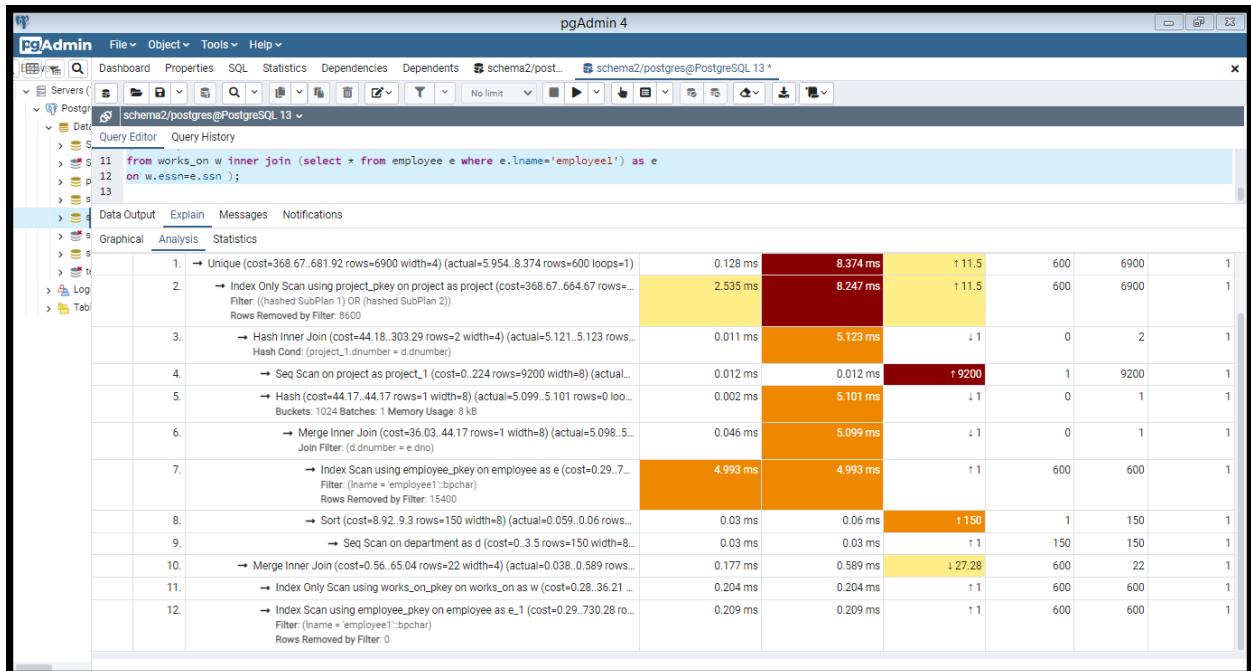
```

pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents schema2/post... schema2/postgres@PostgreSQL 13 *
Servers (1) PostgreSQL (1) schema2/postgres@PostgreSQL 13 *
Data (1) Query Editor (1) Query History
> S 11 from works_on w inner join (select * from employee e where e.lname='employee1') as e
> p 12 on w.essn=e.ssn ;
> s 13
> Data Output Explain Messages Notifications
> S QUERY PLAN text
> S 10 [...] >> Hash (cost=44.17..44.17 rows=1 width=8) (actual time=3.944..3.946 rows=0 loops=1)
> Log 11 [...] Buckets: 1024 Batches: 1 Memory Usage: 8 kB
> Tab 12 [...] >> Merge Join (cost=36.03..44.17 rows=1 width=8) (actual time=3.943..3.943 rows=0 loops=1)
13 [...] Merge Cond: (e.essn = d.mgr_ssn)
14 [...] Join Filter: (d.dnumber = d.hno)
15 [...] >> Index Scan using employee_pkey on employee e (cost=0.29..730.28 rows=600 width=8) (actual time=0.014..3.839 rows=600 loops=1)
16 [...] Filter: (lname = 'employee1'::bpchar)
17 [...] Rows Removed by Filter: 15400
18 [...] >> Sort (cost=8.92..9.30 rows=150 width=8) (actual time=0.057..0.058 rows=1 loops=1)
19 [...] Sort Key: d.mgr_ssn
20 [...] Sort Method: quicksort Memory: 32 kB
21 [...] >> Seq Scan on department d (cost=0.00..3.50 rows=150 width=8) (actual time=0.010..0.030 rows=150 loops=1)
22 [...] SubPlan 2
23 [...] >> Merge Join (cost=0.56..65.04 rows=22 width=4) (actual time=0.033..0.588 rows=600 loops=1)
24 [...] Merge Cond: (w.essn = e_1.ssn)
25 [...] >> Index Only Scan using works_on_pkey on works_on w (cost=0.28..36.21 rows=600 width=8) (actual time=0.020..0.200 rows=600 loops=1)
26 [...] Heap Fetches: 600
27 [...] >> Index Scan using employee_pkey on employee e_1 (cost=0.29..730.28 rows=600 width=4) (actual time=0.011..0.216 rows=600 loops=1)
28 [...] Filter: (lname = 'employee1'::bpchar)
29 Planning Time: 0.805 ms
30 Execution Time: 7.212 ms

```

Successfully run. Total query runtime: 58 msec. 30 rows affected.

Execution plan and costs:



Query2 optimized using B+ tree on : Department(Dnumber) , Project(Dnumber) , Project(Pnumber) , Employee(Lname)

Disabled flags: seqscan and bitmapscan

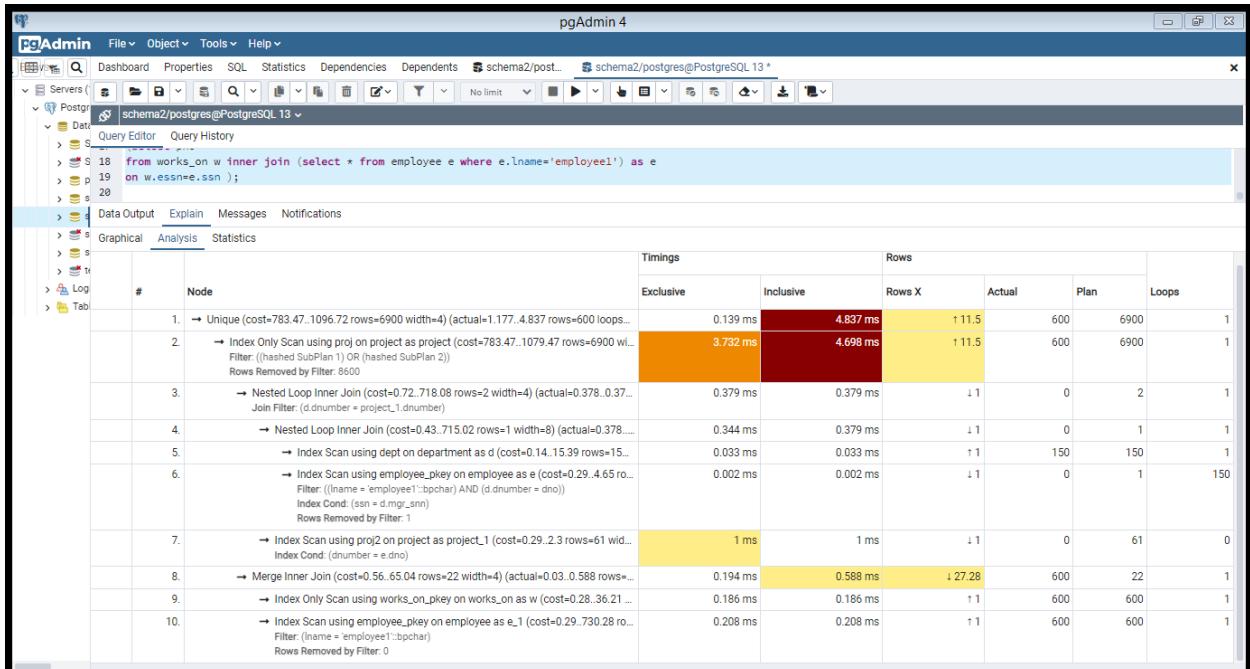
Planning and execution time:

```

pgAdmin 4
File Object Tools Help
Servers (1) PostgreSQL schema2/postgres@PostgreSQL 13 *
Dashboard Properties SQL Statistics Dependencies Dependents schema2/post...
Query Editor Query History
S 18 from works_on w inner join (select * from employee e where e.lname='employee1') as e
on w.essn=e.ssn ;
p 19
s 20
Data Output Explain Messages Notifications
QUERY PLAN
text
4 [...] Rows Removed by Filter: 8600
5 [...] Heap Fetches: 0
6 [...] SubPlan 1
7 [...] -> Nested Loop (cost=0.72..718.08 rows=2 width=4) (actual time=0.437..0.438 rows=0 loops=1)
8 [...] Join Filter: (d.dnumber = project_1.dnumber)
9 [...] -> Nested Loop (cost=0.43..715.02 rows=1 width=8) (actual time=0.436..0.437 rows=0 loops=1)
10 [...] -> Index Scan using dept on department d (cost=0.14..15.39 rows=150 width=8) (actual time=0.009..0.039 rows=150 loops=1)
11 [...] -> Index Scan using employee_pkey on employee e (cost=0.29..4.65 rows=1 width=8) (actual time=0.002..0.002 rows=0 loops=1)
12 [...] Index Cond: (ssn = d.mgr_ssn)
13 [...] Filter: ((lname = 'employee1'.bpchar) AND (d.dnumber = dno))
14 [...] Rows Removed by Filter: 1
15 [...] -> Index Scan using proj2 on project project_1 (cost=0.29..2.30 rows=61 width=8) (never executed)
16 [...] Index Cond: (dnumber = e.dno)
17 [...] SubPlan 2
18 [...] -> Merge Join (cost=0.56..65.04 rows=22 width=4) (actual time=0.036..0.596 rows=600 loops=1)
19 [...] Merge Cond: (w.essn = e_1.ssn)
20 [...] -> Index Only Scan using works_on_pkey on works_on w (cost=0.28..36.21 rows=600 width=8) (actual time=0.021..0.196 rows=600 loops=1)
21 [...] Heap Fetches: 600
22 [...] -> Index Scan using employee_pkey on employee e_1 (cost=0.29..730.28 rows=600 width=4) (actual time=0.013..0.217 rows=600 loops=1)
23 [...] Filter: ((lname = 'employee1'.bpchar)
24 Planning Time: 1.170 ms
25 Execution Time: 3.753 ms
Successfully run. Total query runtime: 53 msec. 25 rows affected.

```

Execution plan and costs:



Query2 optimized using hash indecies on : Department(Dnumber) , Project(Pnumber) , Project(Dnumber), Employee(Lname)

Disabled flags: seqscan and bitmapscan

planning and execution time:

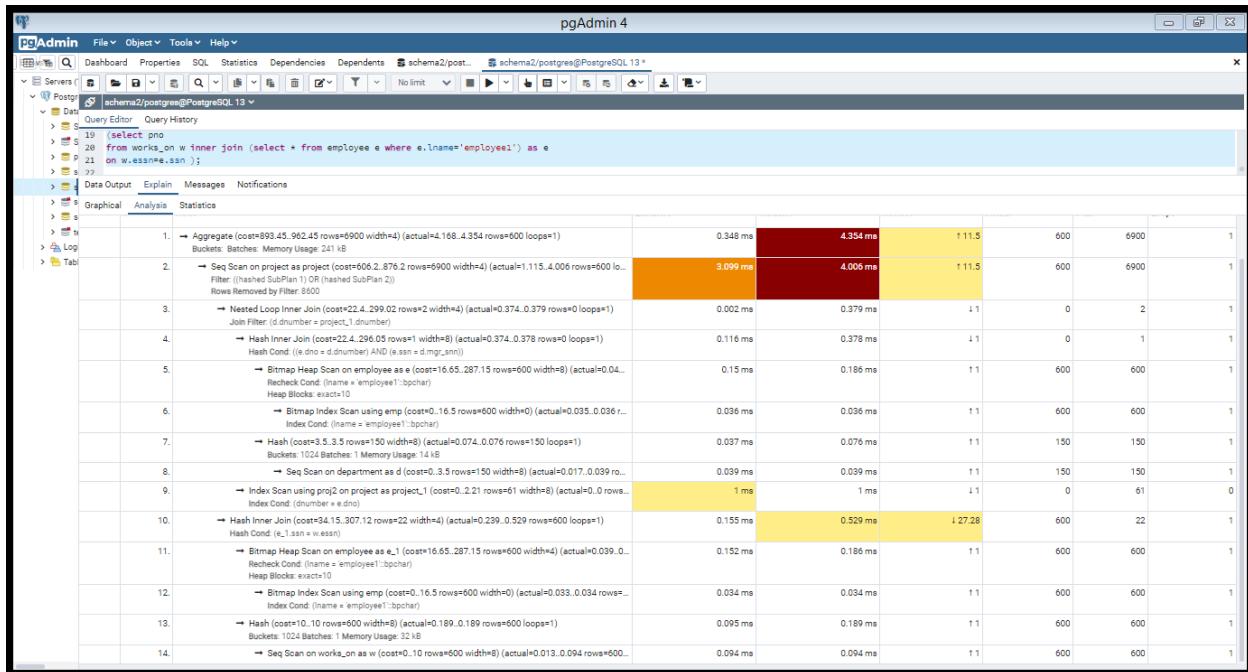
The screenshot shows the pgAdmin 4 interface with the following details:

- Query Plan:**

```

    SELECT * FROM works_on w INNER JOIN (SELECT * FROM employee e WHERE e.lname='employee1') AS e
    ON w.empno=s.empno;
  
```
- Execution Time:** 75 msec.
- Rows Affected:** 34
- Success Message:** Successfully run. Total query runtime: 75 msec. 34 rows affected.

execution plan and costs:



Query2 optimize using BRIN indecies on : Department(Dnumber) , Project(Pnumber) , Project(Dnumber), Employee(Lname)

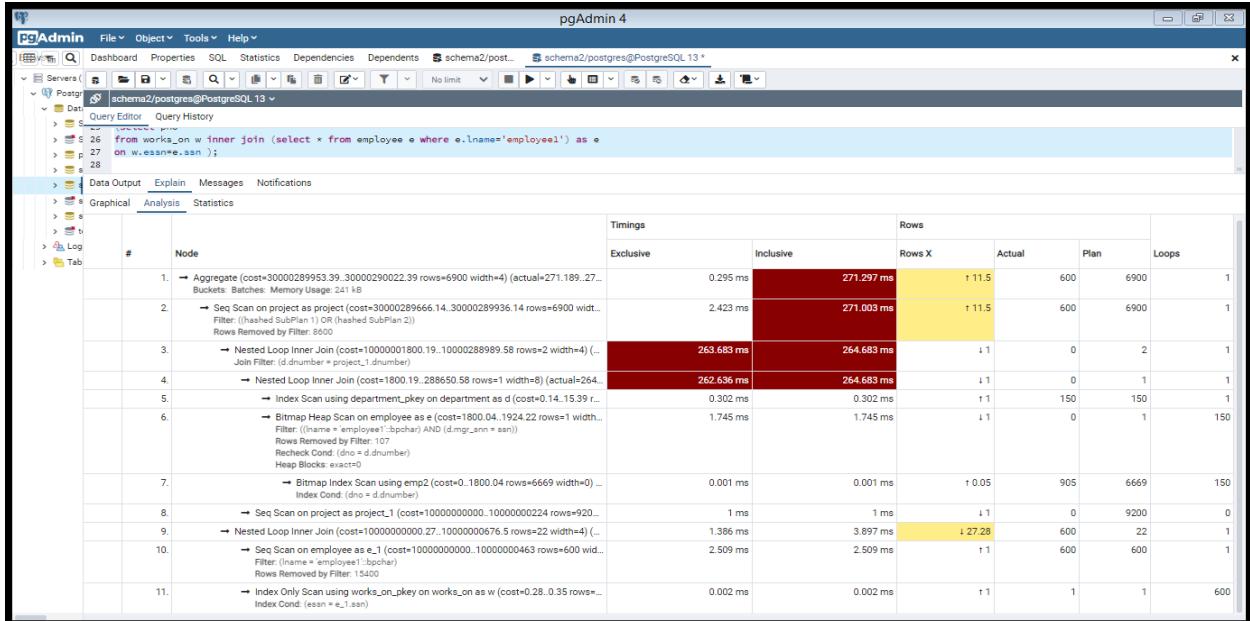
Planning and execution time:

```

pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents schema2/post... schema2/postgres@PostgreSQL 13 *
Servers Data Query Editor History
1 set enable_seqscan=off;
2 set enable_sortoff;
3 set enable_bitmapscan=on;
4 QUERY PLAN
5 tesp
6 ... Rows Removed by Filter: 8600
7 SubPlan 1
8 ... Nested Loop (costs=100000001.800.19... 10000288989.58 rows=2 width=4) (actual time=212.930..212.931 rows=0 loops=1)
9 ... Join Filter: (d.dnumber = project_1.dnumber)
10 ... Nested Loop (costs=1800.19... 288650.58 rows=1 width=8) (actual time=212.930..212.931 rows=0 loops=1)
11 ... Index Scan using department_pkey on department d (cost=0.14... 15.39 rows=150 width=8) (actual time=0.012..0.198 rows=150 loops=1)
12 ... Bitmap Heap Scan on employee e (cost=1800.04... 1924.22 rows=1 width=8) (actual time=1.405..1.405 rows=0 loops=150)
13 ... Recheck Cond: (dno = d.dnumber)
14 ... Rows Removed by Index Recheck: 5414
15 ... Filter: (lname = employee1.bpcchar) AND (dmgr_ssn = ssn)
16 ... Rows Removed by Filter: 107
17 ... Heap Blocks: lossy=13575
18 ... Bitmap Index Scan on emp2 (cost=0.00..1800.04 rows=6669 width=0) (actual time=0.016..0.016 rows=905 loops=150)
19 ... Index Cond: (dno = d.dnumber)
20 ... Seq Scan on project project_1 (cost=1000000000.00... 10000000224.00 rows=9200 width=8) (never executed)
21 ... SubPlan 2
22 ... Nested Loop (cost=1000000000.27... 10000000676.50 rows=22 width=4) (actual time=0.045..4.843 rows=600 loops=1)
23 ... Seq Scan on employee e_1 (cost=1000000000.00... 10000000463.00 rows=600 width=4) (actual time=0.015..3.427 rows=600 loops=1)
24 ... Filter: (lname = employee1.bpcchar)
25 ... Rows Removed by Filter: 15400
26 ... Index Only Scan using works_on_pkey on works_on w (cost=0.28..0.35 rows=1 width=8) (actual time=0.002..0.002 rows=1 loops=600)
27 ... Index Cond: (esnn = e_1.ssn)
28 ... Heap Fetches: 600
29 Planning Time: 0.982 ms
30 Execution Time: 220.788 ms

```

Execution plan and costs:



Using brin with seqscan off was very bad in time and costs as here the BRIN did not help in finding the Employee with Lname=employee1 as it does not help in joins as BRIN is used in small range queries

Using brin with seqscan on

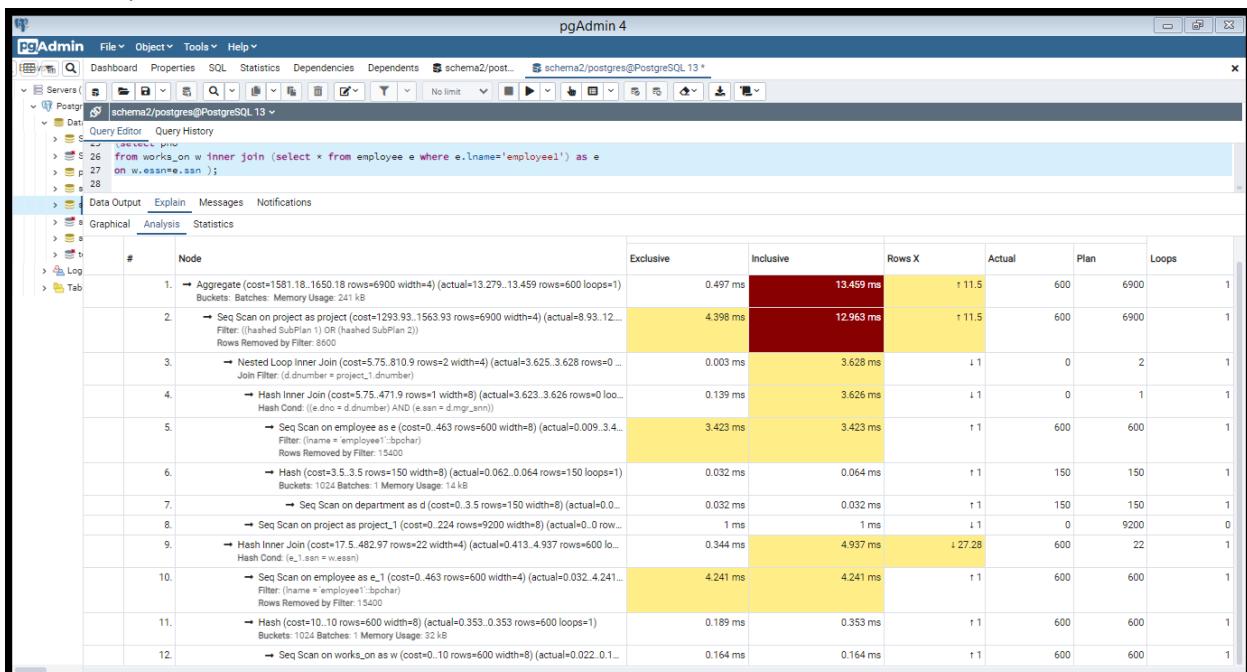
Planning and execution time:

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

- Query Editor:** Contains the SQL query:


```
26 from works_on w inner join (select * from employee e where e.lname='employee1') as e
      27 on w.essn=e.ssn;
      28
```
- Data Output:** Shows the results of the query, indicating 29 rows affected.
- Explain Plan:** Displays the execution plan with the following steps:
 - Step 1: Seq Scan on employee e (cost=0.00..463.00 rows=600 width=8) (actual time=0.008..2.544 rows=600 loops=1)
 - Step 2: Filter: (lname = 'employee1'::bpchar)
 - Step 3: Hash Cond: ((e.dno = d.dnumber) AND (e.ssn = d.mgr.ann))
 - Step 4: Hash Join (cost=5.75..810.90 rows=2 width=4) (actual time=2.732..2.736 rows=0 loops=1)
 - Join Filter: (d.dnumber = project_1.dnumber)
 - Step 5: Hash Join (cost=5.75..471.90 rows=1 width=8) (actual time=2.732..2.735 rows=0 loops=1)
 - Join Filter: (e.dno = d.dnumber)
 - Step 6: Hash Cond: (e.dno = d.dnumber) AND (e.ssn = d.mgr.ann)
 - Step 7: SubPlan 1
 - Step 8: Nested Loop (cost=5.75..810.90 rows=2 width=4) (actual time=2.732..2.736 rows=0 loops=1)
 - Join Filter: (d.dnumber = project_1.dnumber)
 - Step 9: Hash Join (cost=5.75..471.90 rows=1 width=8) (actual time=2.732..2.735 rows=0 loops=1)
 - Join Filter: (e.dno = d.dnumber)
 - Step 10: Hash Cond: (e.dno = d.dnumber) AND (e.ssn = d.mgr.ann)
 - Step 11: Hash Scan on employee e (cost=0.00..463.00 rows=600 width=8) (actual time=0.008..2.544 rows=600 loops=1)
 - Step 12: Filter: (lname = 'employee1'::bpchar)
 - Step 13: Rows Removed by Filter: 15400
 - Step 14: Hash (cost=3..50.30 rows=150 width=8) (actual time=0.071..0.072 rows=150 loops=1)
 - Step 15: Buckets: 1024 Batches: 1 Memory Usage: 14kB
 - Step 16: Hash (cost=0.00..3.50 rows=150 width=8) (actual time=0.011..0.037 rows=150 loops=1)
 - Step 17: Seq Scan on department d (cost=0.00..3.50 rows=150 width=8) (actual time=0.011..0.037 rows=150 loops=1)
 - Step 18: Seq Scan on project project_1 (cost=0.00..224.00 rows=9200 width=8) (never executed)
 - Step 19: SubPlan 2
 - Step 20: Hash Join (cost=17.50..482.97 rows=22 width=4) (actual time=0.225..2.862 rows=500 loops=1)
 - Step 21: Hash Cond: (e_1.ssn = e.ssn)
 - Step 22: Seq Scan on employee e_1 (cost=0.00..463.00 rows=600 width=4) (actual time=0.024..2.486 rows=600 loops=1)
 - Step 23: Filter: (lname = 'employee1'::bpchar)
 - Step 24: Rows Removed by Filter: 15400
 - Step 25: Hash (cost=10.00..10.00 rows=600 width=8) (actual time=0.183..0.184 rows=600 loops=1)
 - Step 26: Buckets: 1024 Batches: 1 Memory Usage: 32kB
 - Step 27: Seq Scan on works_on w (cost=0.00..10.00 rows=600 width=8) (actual time=0.014..0.092 rows=600 loops=1)
 - Step 28: Planning Time: 0.740 ms
 - Step 29: Execution Time: 10.289 ms
 - Status Bar:** Successfully run. Total query runtime: 73 msec. 29 rows affected.

Execution plan and costs:



Query2 optimized using mixed indecies : Hash index on : Project(Pnumber), Employee(Lname)
B+ tree on : Department(Dnumber), Project(Dnumber)

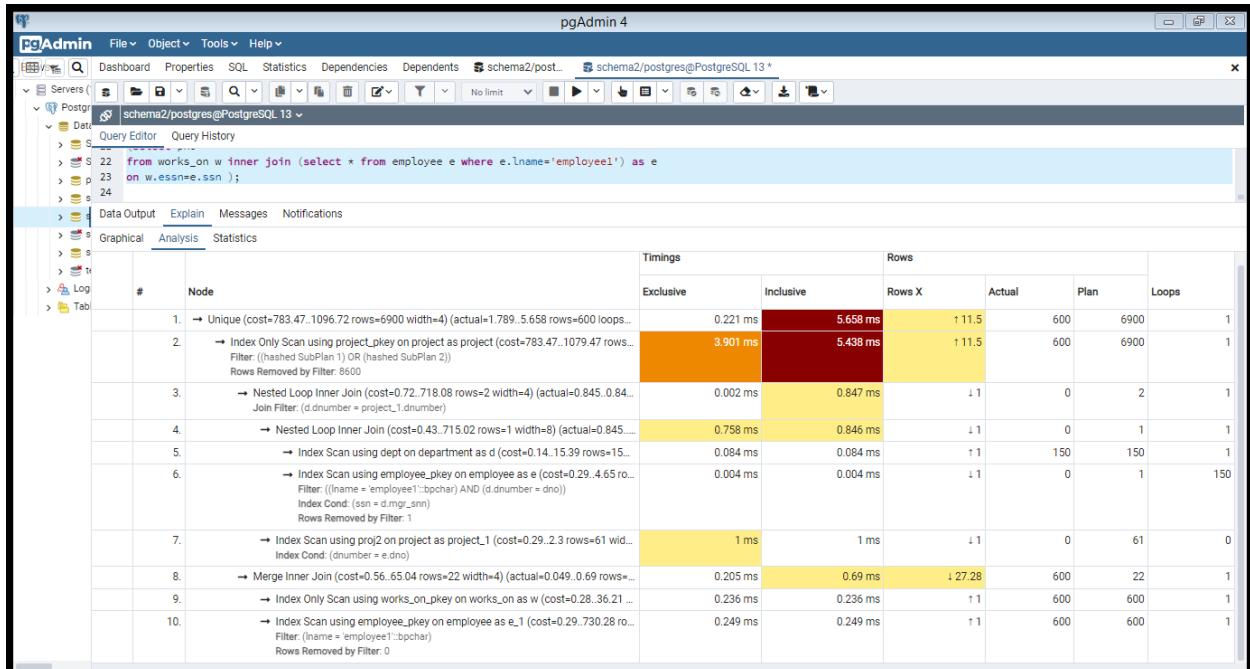
Planning and execution times:

```

pgAdmin 4
File Object Tools Help
Servers (1) PostgreSQL schema2/postgres@PostgreSQL 13 *
Dashboard Properties SQL Statistics Dependencies Dependents schema2/post...
Query Editor Query History
S 22 from works_on w inner join (select * from employee e where e.lname='employee1') as e
> p 23 on w.essn=e.ssn ;
> s 24
Data Output Explain Messages Notifications
> s QUERY PLAN
> s text
4 [...] Rows Removed by Filter: 8600
5 [...] Heap Fetches: 0
6 [...] SubPlan 1
7 [...] -> Nested Loop (cost=0.72..718.08 rows=2 width=4) (actual time=0.518..0.519 rows=0 loops=1)
8 [...] Join Filter: (d.dnumber = project_1.dnumber)
9 [...] -> Nested Loop (cost=0.43..715.02 rows=1 width=8) (actual time=0.517..0.518 rows=0 loops=1)
10 [...] -> Index Scan using dept on department d (cost=0.14..15.39 rows=150 width=8) (actual time=0.014..0.053 rows=150 loops=1)
11 [...] -> Index Scan using employee_pkey on employee e (cost=0.29..4.65 rows=1 width=8) (actual time=0.003..0.003 rows=1 loops=1)
12 [...] Index Cond: (ssn = d.mgr_ssn)
13 [...] Filter: ((lname = 'employee1')::bpchar) AND (d.dnumber = dno)
14 [...] Rows Removed by Filter: 1
15 [...] -> Index Scan using proj2 on project project_1 (cost=0.29..2.30 rows=61 width=8) (never executed)
16 [...] Index Cond: (dnumber = e.dno)
17 [...] SubPlan 2
18 [...] -> Merge Join (cost=0.56..65.04 rows=22 width=4) (actual time=0.034..0.765 rows=600 loops=1)
19 [...] Merge Cond: (w.essn = e_1.ssn)
20 [...] -> Index Only Scan using works_on_pkey on works_on w (cost=0.28..36.21 rows=600 width=8) (actual time=0.019..0.261 rows=600 loops=1)
21 [...] Heap Fetches: 600
22 [...] -> Index Scan using employee_pkey on employee e_1 (cost=0.29..730.28 rows=600 width=4) (actual time=0.012..0.274 rows=600 loops=1)
23 [...] Filter: ((lname = 'employee1')::bpchar)
24 Planning Time: 1.053 ms
25 Execution Time: 4.072 ms
Successfully run. Total query runtime: 63 msec. 25 rows affected.

```

Execution plan and costs:



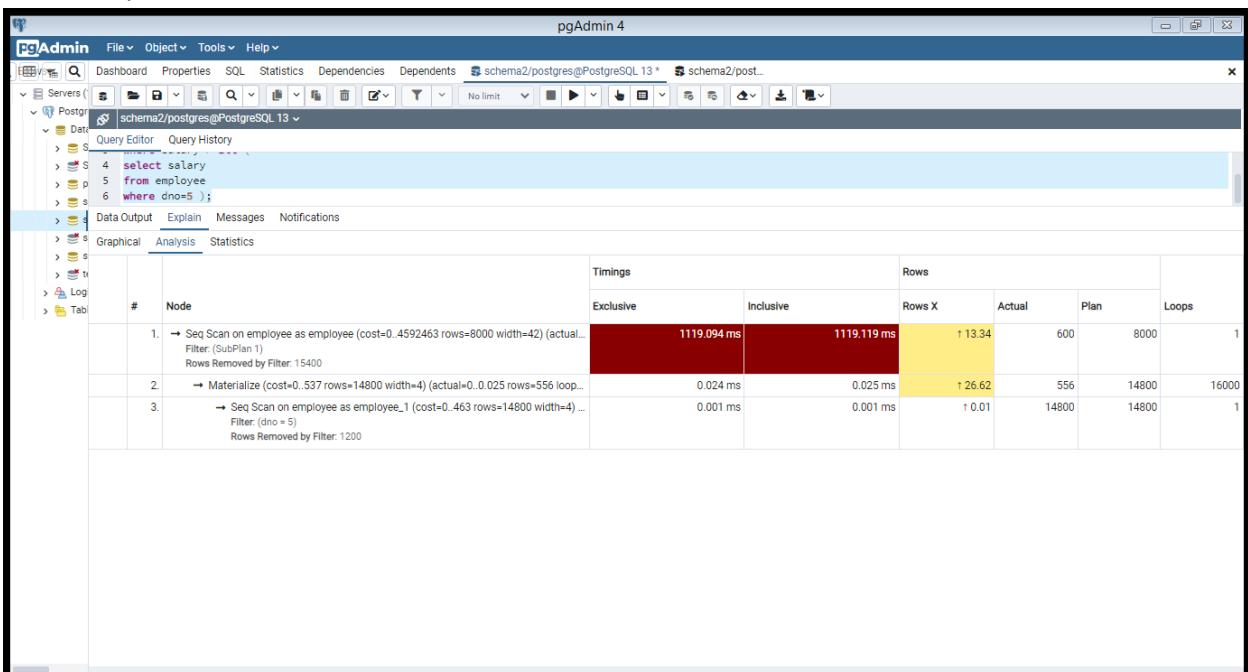
Query3 with no index

Planning and execution times:

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

- Servers:** PostgreSQL - schema2/postgres@PostgreSQL 13*
- Query Editor:** Explain Analyze select lname, fname from employee where salary > all (
- Result pane (QUERY PLAN):**
 - Seq Scan on employee (cost=0.00..4592463.00 rows=8000 width=42) (actual time=10.381..1123.709 rows=600 loops=1)
 - Filter: (SubPlan 1)
 - Materialize (cost=0.00..537.00 rows=14800 width=4) (actual time=0.00..0.025 rows=556 loops=16000)
 - Seq Scan on employee employee_1 (cost=0.00..463.00 rows=14800 width=4) (actual time=0.235..4.682 rows=14800 loops=1)
 - Filter: (dno = 5)
 - Rows Removed by Filter: 1200
 - Planning Time: 0.121 ms
 - Execution Time: 1124.204 ms
- Status bar:** Successfully run. Total query runtime: 1 secs 142 msec. 10 rows affected.

Execution plan and costs:



Query3 using B+ tree on : Employee(salary) , Employee(dno)

With seqscan disabled:

Planning and execution time:

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab selected. The query window contains the following SQL code:

```
1 set enable_seqscan=off;
2 set enable_bitmapscan=off;
3 create index emp on Employee using btree(dno);
4 create index emp2 on Employee using btree(salary);
5 drop index emp;
6 drop index emp2;
7 explain analyze select lname, fname from employee where salary > all (select salary from employee where dno=5);
```

The 'Data Output' tab is active, displaying the execution plan:

QUERY PLAN
1 Seq Scan on employee (cost=1000000000.28..10007738163.03 rows=8000 width=42) (actual time=13.351..1039.289 rows=600 loops=1) 2 [.] Filter: (SubPlan 1) 3 [.] Rows Removed by Filter: 15400 4 [.] SubPlan 1 5 [.]-> Materialize (cost=0.29..930.50 rows=14800 width=4) (actual time=0.000..0.023 rows=556 loops=16000) 6 [.]-> Index Scan using emp on employee employee_1 (cost=0.29..856.50 rows=14800 width=4) (actual time=0.035..6.730 rows=14800 loops=1) 7 [.] Index Cond: (dno = 5) 8 Planning Time: 0.894 ms 9 Execution Time: 1039.617 ms

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 1 secs 59 msec. 9 rows affected."

Execution plan and costs:

The screenshot shows the pgAdmin 4 interface with the 'Explain' tab selected. The query window contains the same SQL code as the previous screenshot.

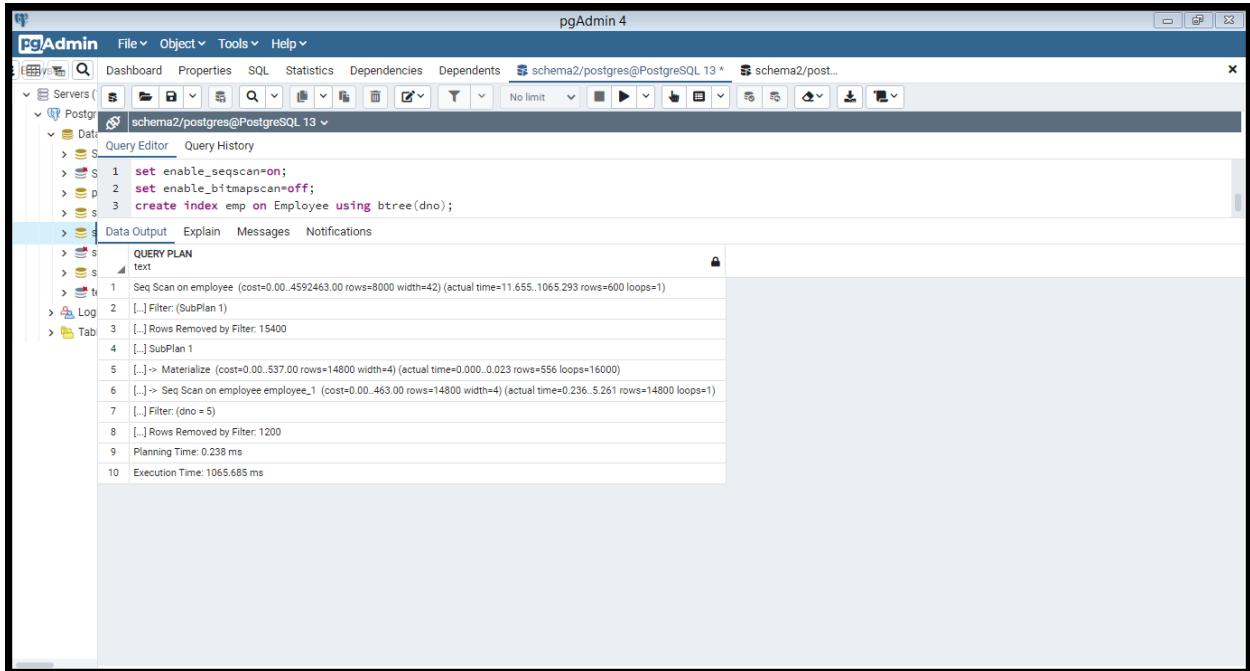
The 'Graphical' tab is active, displaying a detailed execution plan with timing information:

#	Node	Timings		Rows			
		Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Seq Scan on employee as employee (cost=1000000000.28..10007738163.03 rows... Filter: (SubPlan 1) Rows Removed by Filter: 15400	1111.814 ms	1111.838 ms	↑ 13.34	600	8000	1
2.	→ Materialize (cost=0.29..930.5 rows=14800 width=4) (actual=0..0.024 rows=556...	0.023 ms	0.024 ms	↑ 26.62	556	14800	16000
3.	→ Index Scan using emp on employee as employee_1 (cost=0.29..856.5 rows... Index Cond: (dno = 5)	0.001 ms	0.001 ms	↑ 0.01	14800	14800	1

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 1 secs 137 msec. 1 rows affected."

With seqscan enabled:

Planning and execution time:



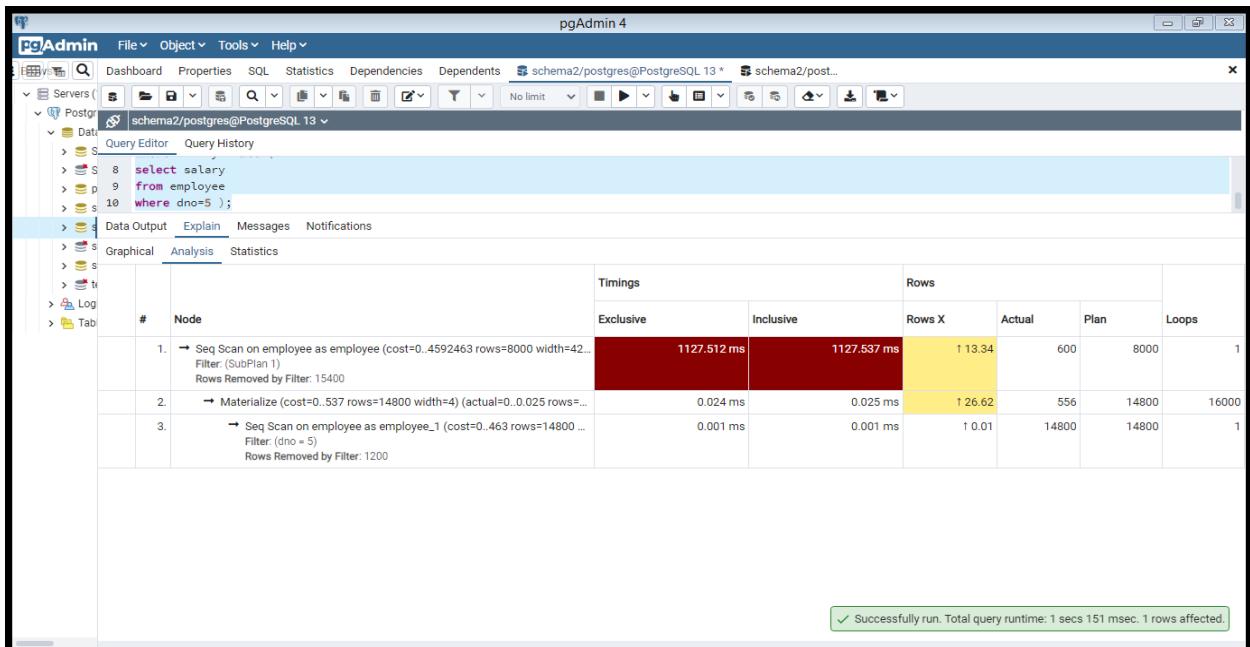
The screenshot shows the pgAdmin 4 interface with the following details:

- Servers:** PostgreSQL - schema2/postgres@PostgreSQL 13*
- Query Editor:** Contains the following SQL code:

```
1 set enable_seqscan=on;
2 set enable_bitmapscan=off;
3 create index emp on Employee using btree(dno);
```
- Explain Plan:** Shows the execution plan for the third statement:

```
1 Seq Scan on employee (cost=0.00..4592463.00 rows=8000 width=42) (actual time=11.655..1065.293 rows=600 loops=1)
  2 [...] Filter (SubPlan 1)
    3 [...] Rows Removed by Filter: 15400
    4 [...] SubPlan 1
    5 [...] >> Materialize (cost=0.00..537.00 rows=14800 width=4) (actual time=0.000..0.023 rows=556 loops=16000)
    6 [...] >> Seq Scan on employee employee_1 (cost=0.00..463.00 rows=14800 width=4) (actual time=0.236..5.261 rows=14800 loops=1)
    7 [...] Filter: (dno = 5)
    8 [...] Rows Removed by Filter: 1200
    9 Planning Time: 0.238 ms
   10 Execution Time: 1065.685 ms
```

Execution plan and costs:



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

- Servers:** PostgreSQL - schema2/postgres@PostgreSQL 13*
- Query Editor:** Contains the following SQL code:

```
8 select salary
  9   from employee
 10  where dno=5;
```
- Explain Graphical:** Shows the execution plan for the query:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Seq Scan on employee as employee (cost=0..4592463 rows=8000 width=42) Filter: (SubPlan 1) Rows Removed by Filter: 15400	1127.512 ms	1127.537 ms	113.34	600	8000	1
2.	→ Materialize (cost=0..537 rows=14800 width=4) (actual=0..0.025 rows=...)	0.024 ms	0.025 ms	126.62	556	14800	16000
3.	→ Seq Scan on employee as employee_1 (cost=0..463 rows=14800 ...) Filter: (dno = 5) Rows Removed by Filter: 1200	0.001 ms	0.001 ms	1.01	14800	14800	1

A green message bar at the bottom right says: "Successfully run. Total query runtime: 1 secs 151 msec. 1 rows affected."

Here the cost without seqscan was so high and the execution time was enhanced but with seqscan the cost was normal because in this query due to (all) it will check every salary in the table so the index did not help

Query3 using hash indecies on : Employee(salary) , Employee(dno)

With seqscan disabled

Planning and execution time:

```

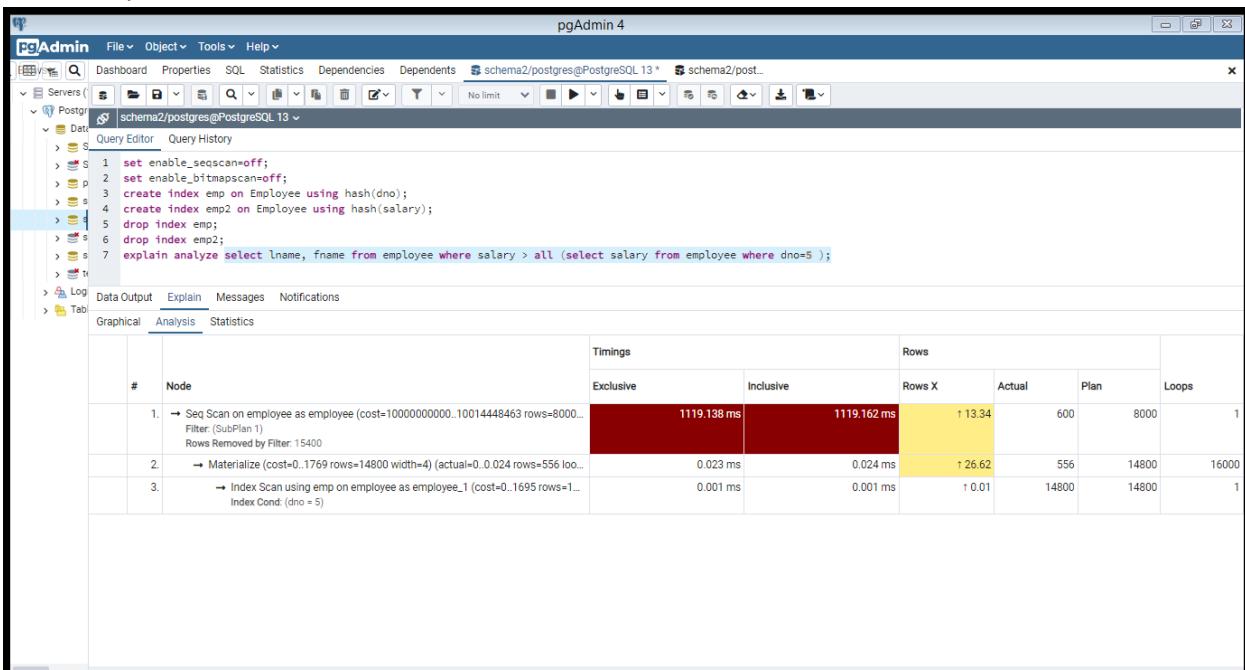
pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents schema2/postgres@PostgreSQL 13* schema2/post...
Servers (1) PostgreSQL (1) schema2/postgres@PostgreSQL 13*
Data (1) Query Editor (1) Query History
> S 1 set enable_seqscan=off;
> S 2 set enable_bitmaps=off;
> p 3 create index emp on Employee using hash(dno);
> s 4 create index emp2 on Employee using hash(salary);
> D 5 drop index emp;
> S 6 drop index emp2;
> S 7 explain analyze select lname, fname from employee where salary > all (select salary from employee where dno=5);

Data Output Explain Messages Notifications
Tab: QUERY PLAN
text
1 Seq Scan on employee (cost=1000000000.00..10014448463.00 rows=8000 width=42) (actual time=20.947..1051.486 rows=600 loops=1)
  2 [.] Filter: (SubPlan 1)
  3 [.] Rows Removed by Filter: 15400
  4 [.] SubPlan 1
  5 [.]-> Materialize (cost=0.00..1769.00 rows=14800 width=4) (actual time=0.001..0.023 rows=556 loops=16000)
  6 [.]-> Index Scan using emp on employee employee_1 (cost=0.00..1695.00 rows=14800 width=4) (actual time=11.201..16.084 rows=14800 loops=1)
  7 [.] Index Cond: (dno = 5)
  8 Planning Time: 0.774 ms
  9 Execution Time: 1051.828 ms

Successfully run. Total query runtime: 1 secs 71 msec. 9 rows affected.

```

Execution plan and costs:



With seqscan enabled :

Planning and execution time:

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers' and 'PostgreSQL', the 'schema2/postgres@PostgreSQL 13' connection is selected. In the main area, the 'Query Editor' tab is active, displaying the following SQL query:

```
select salary
from employee
where dno=5;
```

Below the query, the 'Explain' tab is selected, showing the execution plan:

```
QUERY PLAN
text
1 Seq Scan on employee (cost=0.4592463.00 rows=8000 width=42) (actual time=12.178..1042.525 rows=600 loops=1)
2 [...] Filter: (SubPlan 1)
3 [...] Rows Removed by Filter: 15400
4 [...] SubPlan 1
5 [...] > Materialize (cost=0.537.00 rows=14800 width=4) (actual time=0.000..0.023 rows=556 loops=16000)
6 [...] > Seq Scan on employee employee_1 (cost=0.463.00 rows=14800 width=4) (actual time=0.358..5.670 rows=14800 loops=1)
7 [...] Filter: (dno = 5)
8 [...] Rows Removed by Filter: 1200
9 Planning Time: 0.755 ms
10 Execution Time: 1042.892 ms
```

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 1 secs 65 msec. 10 rows affected."

Execution plan and costs:

The screenshot shows the pgAdmin 4 interface with the 'Explain' tab selected. Below the execution plan from the previous screenshot, there are two tabs: 'Graphical' and 'Analysis'. The 'Analysis' tab is currently active, displaying a detailed execution plan table:

#	Node	Timings		Rows			
		Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Seq Scan on employee as employee (cost=0.4592463 rows=8000 width=42, Filter: (SubPlan 1) Rows Removed by Filter: 15400	1137.896 ms	1137.921 ms	113.34	600	8000	1
2.	→ Materialize (cost=0.537 rows=14800 width=4) (actual=0.025 rows=556)	0.024 ms	0.025 ms	126.62	556	14800	16000
3.	→ Seq Scan on employee as employee_1 (cost=0.463 rows=14800 width=4, Filter: (dno = 5) Rows Removed by Filter: 1200	0.001 ms	0.001 ms	1.01	14800	14800	1

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 1 secs 161 msec. 1 rows affected."

Here the cost without seqscan was so high and the execution time was enhanced but with seqscan the cost was normal because in this query due to (all) it will check every salary in the table so the index did not help

Query 3 using BRIN indecies on : Employee(dno) and Employee(salary)

With seqscan disabled

Planning and execution time:

```

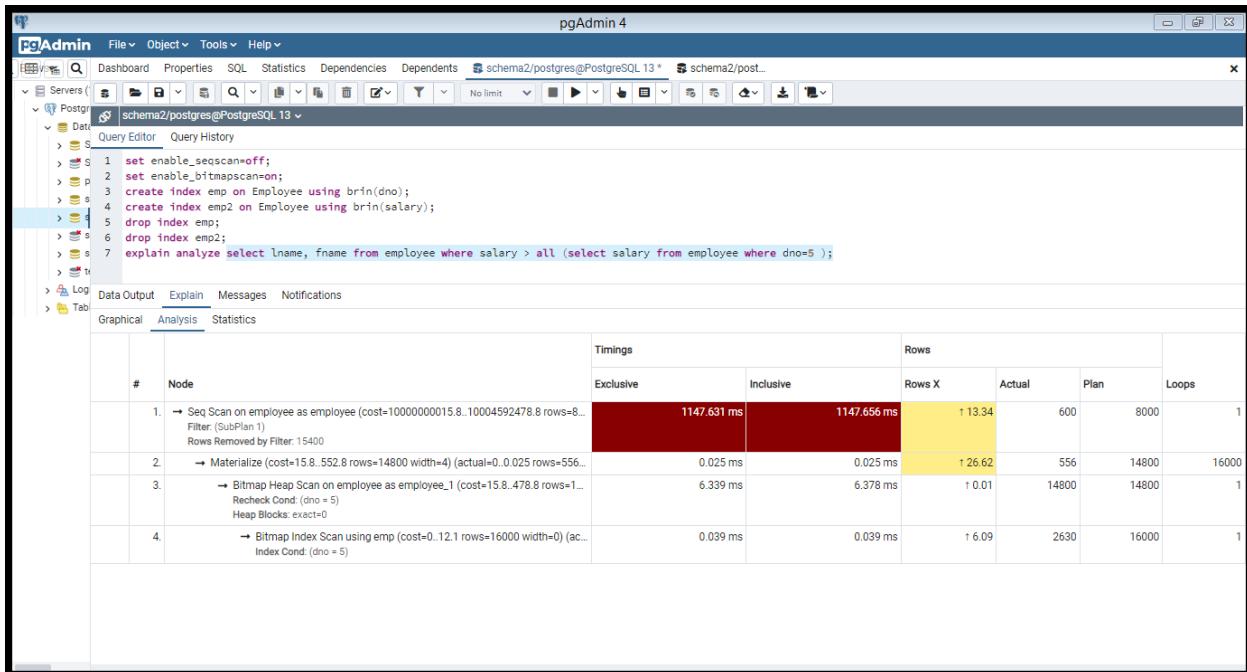
pgAdmin 4
File ▾ Object ▾ Tools ▾ Help ▾
Dashboard Properties SQL Statistics Dependencies Dependents schema2/postgres@PostgreSQL 13* schema2/post...
Servers (1) Postgres (1) schema2/postgres@PostgreSQL 13*
Data (1) Query Editor (1) Query History (1)
S 1 set enable_seqscan=off;
S 2 set enable_bitmapscan=on;
P 3 create index emp on Employee using brin(dno);
S 4 create index emp2 on Employee using brin(salary);
D 5 drop index emp;
S 6 drop index emp2;
T 7 explain analyze select lname, fname from employee where salary > all (select salary from employee where dno=5 );
Log Data Output Explain Messages Notifications Tab (1)

QUERY PLAN
text
1 Seq Scan on employee (cost=10000000015.80..10004592478.80 rows=8000 width=42) (actual_time=10.216..1040.087 rows=600 loops=1)
2 (...) Filter: (SubPlan 1)
3 (...) Rows Removed by Filter: 15400
4 (...) SubPlan 1
5 (...) -> Materialize (cost=15.80..552.80 rows=14800 width=4) (actual_time=0.000..0.023 rows=556 loops=16000)
6 (...) -> Bitmap Heap Scan on employee_1 (cost=15.80..478.80 rows=14800 width=4) (actual_time=0.321..4.935 rows=14800 loops=1)
7 (...) Recheck Cond: (dno = 5)
8 (...) Rows Removed by Index Recheck: 1200
9 (...) Heap Blocks: lossy=263
10 (...) -> Bitmap Index Scan on emp (cost=0.00..12.10 rows=16000 width=0) (actual_time=0.040..0.040 rows=2630 loops=1)
11 (...) Index Cond: (dno = 5)
12 Planning Time: 0.174 ms
13 Execution Time: 1040.474 ms

Success! Successfully run. Total query runtime: 1 secs 61 msec. 13 rows affected.

```

Execution plan and costs:



With seqscan enabled

Planning and execution time:

The screenshot shows the pgAdmin 4 interface with the 'Explain' tab selected. The query being analyzed is:

```
select salary
from employee
where dno=5;
```

The explain plan output is:

```
1 Seq Scan on employee (cost=0.00..4592463.00 rows=8000 width=42) (actual time=11.253..1114.900 rows=600 loops=1)
2 [...] Filter: (SubPlan 1)
3 [...] Rows Removed by Filter: 15400
4 [...] SubPlan 1
5 [...] -> Materialize (cost=0.00..537.00 rows=14800 width=4) (actual time=0.000..0.024 rows=556 loops=16000)
6 [...] -> Seq Scan on employee employee_1 (cost=0.00..463.00 rows=14800 width=4) (actual time=0.234..5.103 rows=14800 loops=1)
7 [...] Filter: (dno = 5)
8 [...] Rows Removed by Filter: 1200
9 Planning Time: 0.155 ms
10 Execution Time: 1115.411 ms
```

Execution plan and costs:

The screenshot shows the pgAdmin 4 interface with the 'Analysis' tab selected. The same query is run again, showing the detailed execution costs:

#	Node	Timings		Rows			
		Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	Seq Scan on employee as employee (cost=0..4592463 rows=8000 width=42) Filter: (SubPlan 1) Rows Removed by Filter: 15400	1115.125 ms	1115.15 ms	113.34	600	8000	1
2.	→ Materialize (cost=0..537 rows=14800 width=4) (actual=0..0.025 rows=5...	0.024 ms	0.025 ms	126.62	556	14800	16000
3.	→ Seq Scan on employee as employee_1 (cost=0..463 rows=14800 width=4) Filter: (dno = 5) Rows Removed by Filter: 1200	0.001 ms	0.001 ms	10.01	14800	14800	1

A green message bar at the bottom right indicates: "Successfully run. Total query runtime: 1 secs 137 msec. 1 rows affected."

Query3 using mixed indecies :

B+ tree on Employee(dno) and hash on Employee(salary)

With seqscan disabled

Planning and execution time:

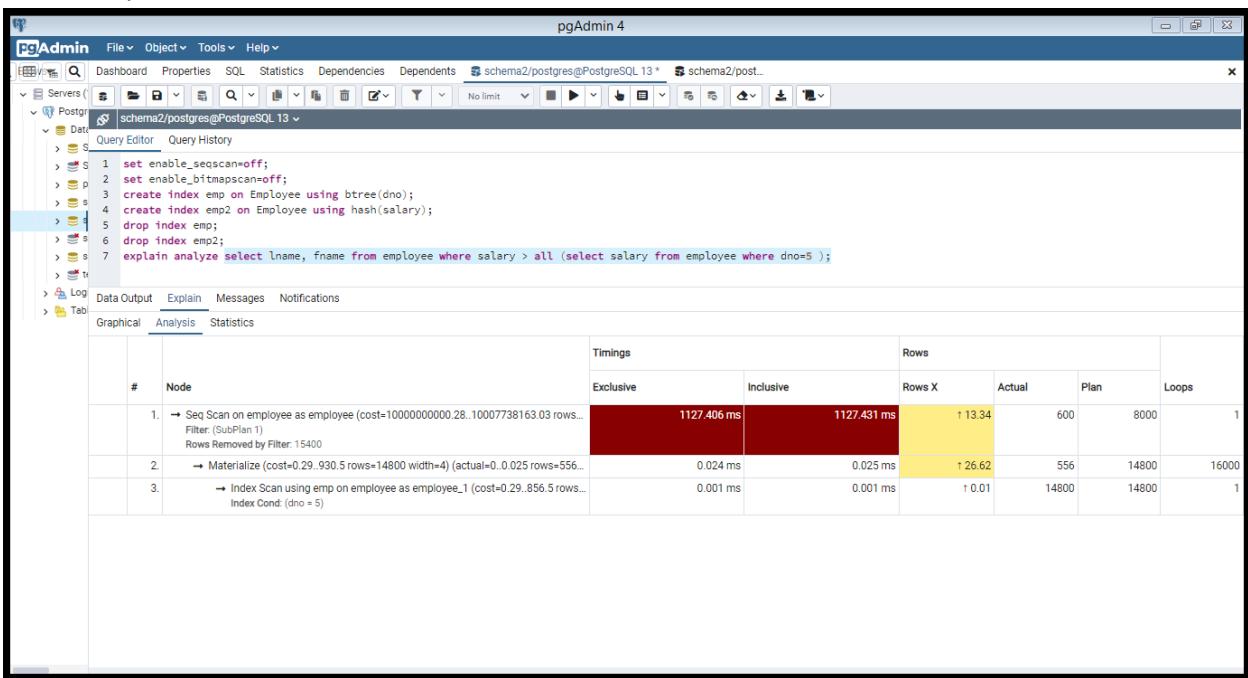
The screenshot shows the pgAdmin 4 interface with a query editor containing the following SQL code:

```
> S_ 1 set enable_seqscan=off;
> S_ 2 set enable_bitmapscan=off;
> P_ 3 create index emp on Employee using btree(dno);
> S_ 4 create index emp2 on Employee using hash(salary);
> D_ 5 drop index emp;
> S_ 6 drop index emp2;
> S_ 7 explain analyze select lname, fname from employee where salary > all (select salary from employee where dno=5 );
```

The "Explain" tab is selected, displaying the following query plan:

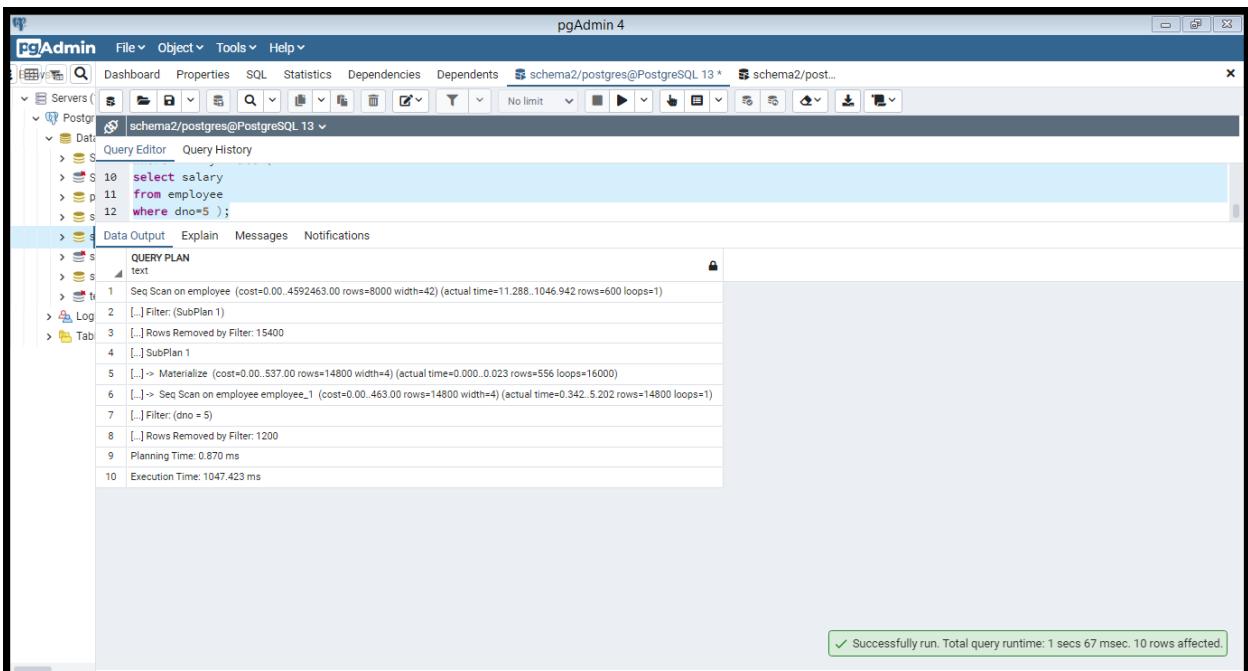
QUERY PLAN
1 Seq Scan on employee (cost=1000000000.28..10007738163.03 rows=8000 width=42) (actual time=13.550..1055.318 rows=600 loops=1) 2 [.] Filter: (SubPlan 1) 3 [.] Rows Removed by Filter: 15400 4 [.] SubPlan 1 5 [.] -> Materialize (cost=0.29..930.50 rows=14800 width=4) (actual time=0.000..0.023 rows=556 loops=16000) 6 [.] -> Index Scan using emp on employee_employee_1 (cost=0.29..856.50 rows=14800 width=4) (actual time=0.019..6.740 rows=14800 loops=1) 7 [.] Index Cond: (dno = 5) 8 Planning Time: 0.188 ms 9 Execution Time: 1055.685 ms

Execution plan and costs:

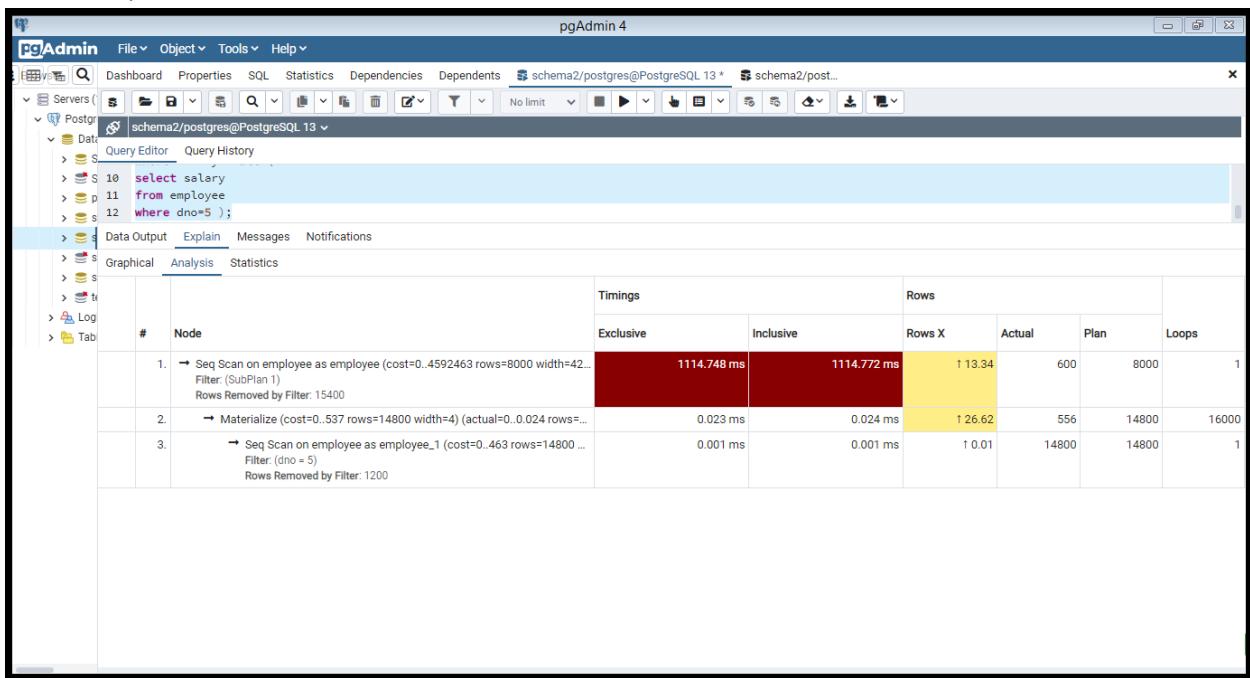


With seqscan enabled:

Planning and execution time:



Execution plan and costs:



Here in query 3 using indecies made the cost worse but it decreased the execution time

But if we used seqscan instead with no indecies it decrease the cost but increase the execution time

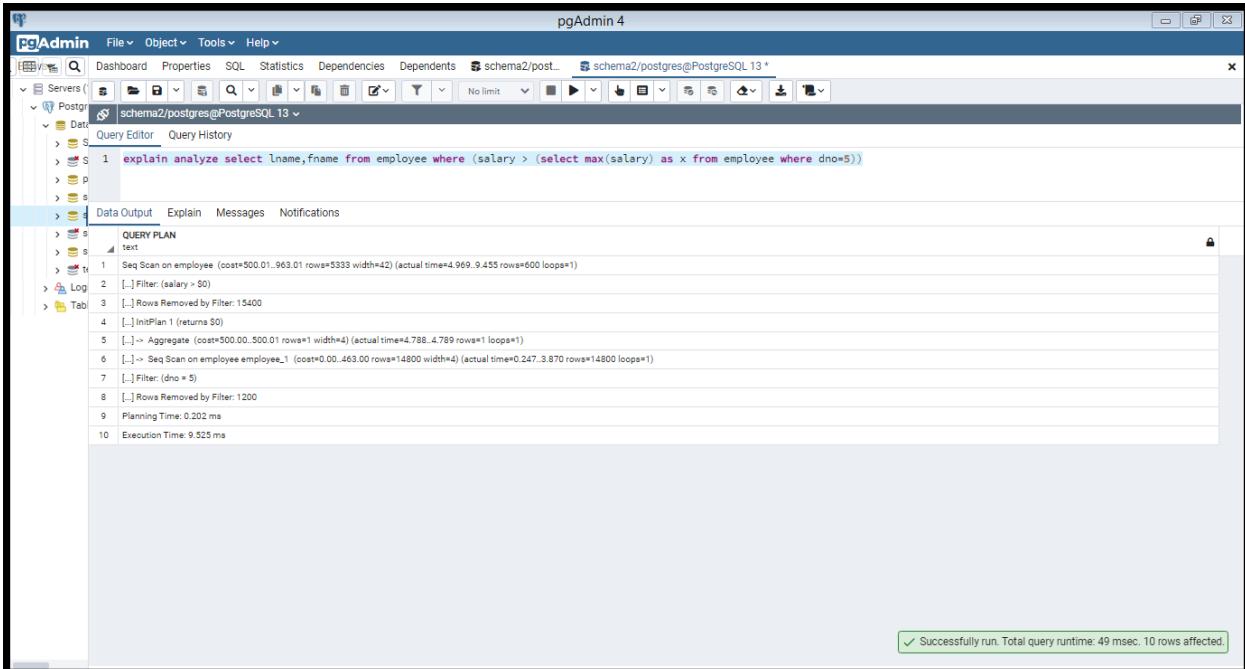
Query3 after optimization:

```
select lname, fname from employee where (salary > (select max(salary) as x from employee where dno=5))
```

we replaced the check done by (all) with max as aggregate functions is better as it can use the indecies made on the column

Query 3 optimized with no index

Planning and execution time:



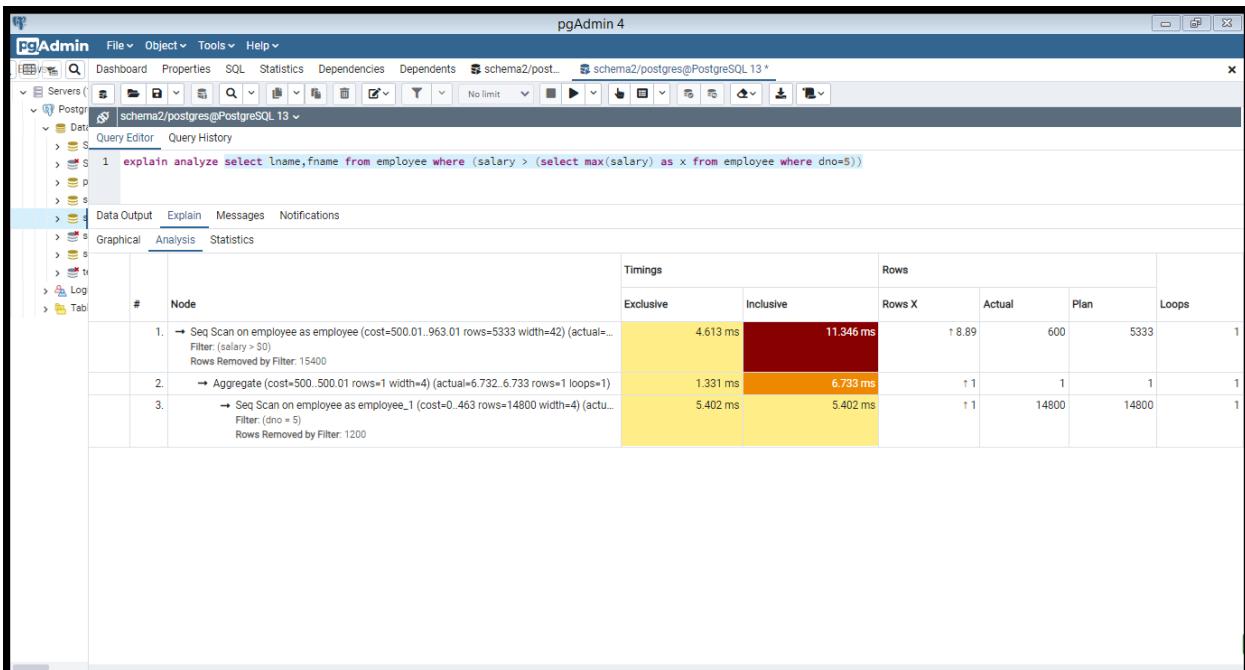
```
explain analyze select lname, fname from employee where (salary > (select max(salary) as x from employee where dno=5))
```

QUERY PLAN

```
1 Seq Scan on employee (cost=500.01..963.01 rows=5333 width=42) (actual time=4.969..9.455 rows=600 loops=1)
2 [...] Filter: (salary > $0)
3 [...] Rows Removed by Filter: 15400
4 [...] InitPlan 1 returns $0
5 [...]-> Aggregate (cost=500.00..500.01 rows=1 width=4) (actual time=4.788..4.789 rows=1 loops=1)
6 [...]-> Seq Scan on employee employee_1 (cost=0.00..463.00 rows=14800 width=4) (actual time=0.247..3.870 rows=14800 loops=1)
7 [...] Filter: (dno = 5)
8 [...] Rows Removed by Filter: 1200
9 Planning Time: 0.202 ms
10 Execution Time: 9.525 ms
```

Successfully run. Total query runtime: 49 msec. 10 rows affected.

Execution plan and costs:



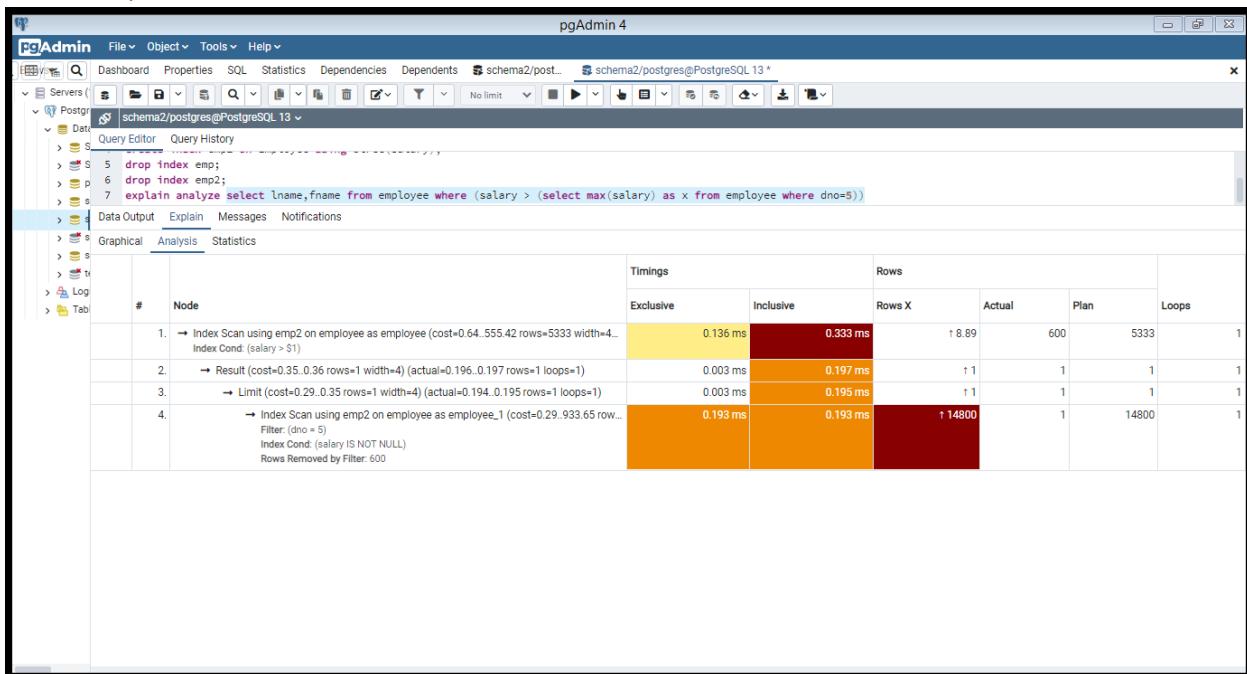
Query3 optimized using B+ tree on : Employee(dno) and Employee(salary)

Disabled flags: seqscan and bitmapscan

Planning and execution time:

```
pgAdmin 4
File ▾ Object ▾ Tools ▾ Help ▾
Dashboard Properties SQL Statistics Dependencies Dependents schema2/post... schema2/postgres@PostgreSQL 13 *
Servers (1) Postgres (1) schema2/postgres@PostgreSQL 13 *
Data (1) S (1) Query Editor (1) Query History
S (1) S (1) 5 drop index emp;
S (1) S (1) 6 drop index emp2;
S (1) S (1) 7 explain analyze select lname, fname from employee where (salary > (select max(salary) as x from employee where dno=5));
Data Output Explain Messages Notifications
QUERY PLAN
text
1 Index Scan using emp2 on employee (cost=0.64..555.42 rows=5333 width=42) (actual time=0.249..0.397 rows=600 loops=1)
[...] Index Cond: (salary > $1)
3 [...] InitPlan 2 returns $1
4 [...] > Result (cost=0.35..0.36 rows=1 width=4) (actual time=0.231..0.232 rows=1 loops=1)
5 [...] InitPlan 1 (returns $0)
6 [...] > Limit (cost=0.29..0.35 rows=1 width=4) (actual time=0.229..0.229 rows=1 loops=1)
7 [...] > Index Scan Backward using emp2 on employee employee_1 (cost=0.29..933.65 rows=14800 width=4) (actual time=0.227..0.227 rows=1 loops=1)
8 [...] Index Cond: (salary IS NOT NULL)
9 [...] Filter: (dno = 5)
10 [...] Rows Removed by Filter: 600
11 Planning Time: 0.278 ms
12 Execution Time: 0.449 ms
```

Execution plan and costs:



Using the B+ tree index improved performance as it helps in aggregate function like max as it search in O(logn) so it found the max number rapidly and used it in the query

Query3 optimized using hash indecies on : Employee(dno) and Employee(salary)

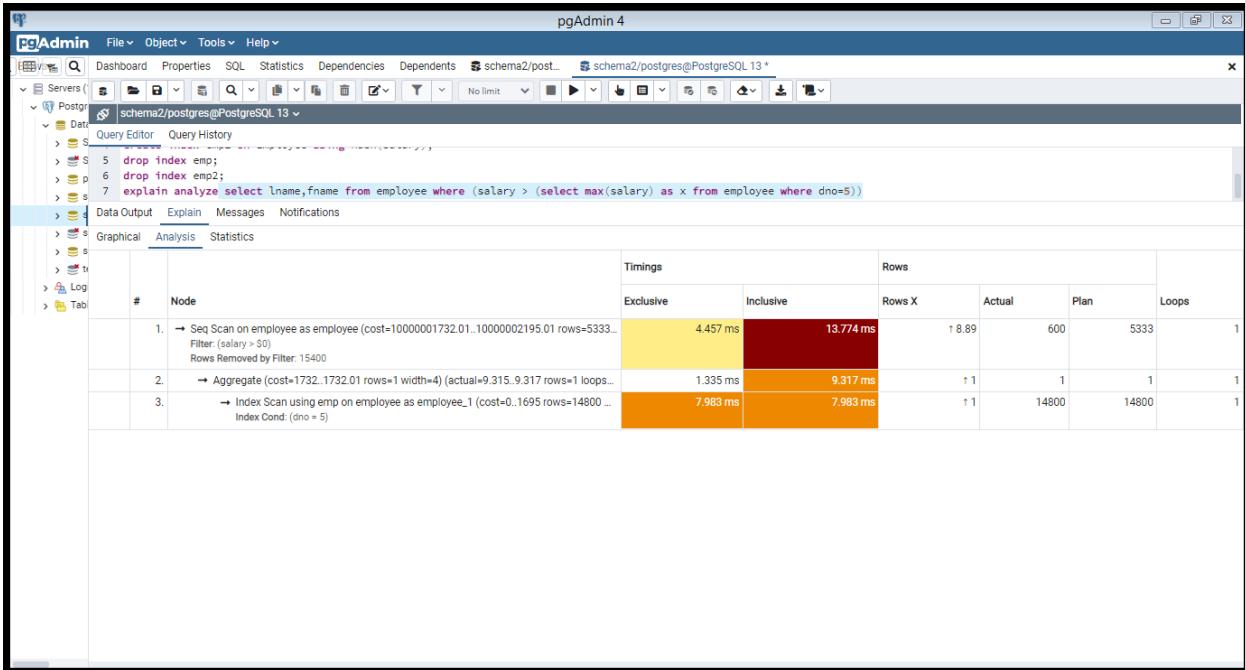
Planning and execution time:

```

pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents schema2/post... schema2/postgres@PostgreSQL_13 *
Servers (1) Postgres schema2/postgres@PostgreSQL_13
  Data (1)
    Query Editor (1)
      1 drop index emp;
      2 drop index emp2;
      3 explain analyze select lname, fname from employee where (salary > (select max(salary) as x from employee where dno=5))
      4 Data Output (1)
      5 Explain (1)
      6 Messages (1)
      7 Notifications (1)

  Explain
  QUERY PLAN
  text
  1 Seq Scan on employee (cost=10000001732.01..10000002195.01 rows=5333 width=42) (actual time=5.868..9.113 rows=600 loops=1)
  2 [...] Filter: (salary > $0)
  3 [...] Rows Removed by Filter: 15400
  4 [...] IntPlan 1 (returns $0)
  5 [...] -> Aggregate (cost=1732.00..1732.01 rows=1 width=4) (actual time=5.731..5.733 rows=1 loops=1)
  6 [...] -> Index Scan using emp on employee employee_1 (cost=0.00..1695.00 rows=14800 width=4) (actual time=0.011..4.822 rows=14800 loops=1)
  7 [...] Index Cond: (dno = 5)
  8 Planning Time: 0.261 ms
  9 Execution Time: 9.163 ms
  
```

Execution plan and costs:



Here using hash did not help in the max function as the hash is used in exact value queries it does not help in searching for the max number and does not help in range queries so it increased the cost

Query3 optimized using BRIN indecies on: Employee(dno) and Employee(salary)

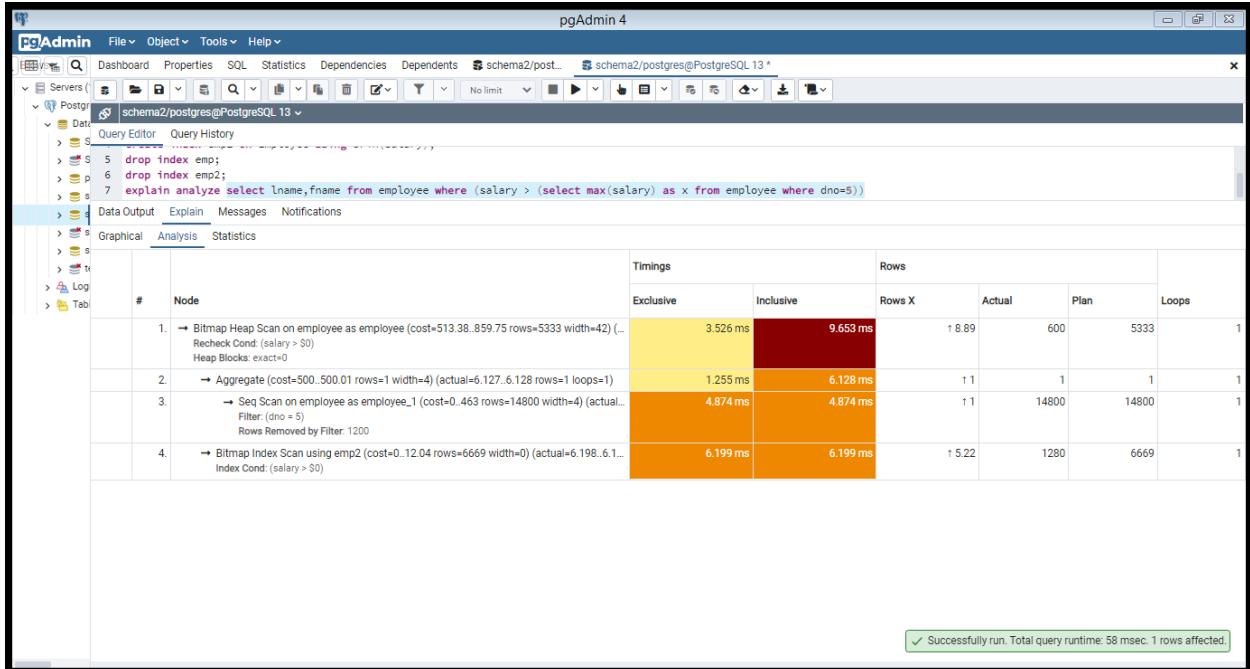
Planning and execution time:

```

pgAdmin 4
File Object Tools Help
Servers (1) PostgreSQL schema2/postgres@PostgreSQL 13 *
Dashboard Properties SQL Statistics Dependencies Dependents schema2/postgres@PostgreSQL 13 *
No limit ▾
Query Editor Query History
> S 5 drop index emp;
> p 6 drop index emp2;
> a 7 explain analyze select lname, fname from employee where (salary > (select max(salary) as x from employee where dno=5))
Data Output Explain Messages Notifications
> S QUERY PLAN
> S text
> t 1 Bitmap Heap Scan on employee (cost=513.38..859.75 rows=5333 width=42) (actual time=5.306..7.415 rows=600 loops=1)
> Log 2 [...] Recheck Cond: (salary > $0)
> Tabl 3 [...] Rows Removed by Index Recheck: 7208
4 [...] Heap Blocks: lossy=128
5 [...] InitPlan 1 (returns $0)
6 [...] -> Aggregate (cost=500.00..500.01 rows=1 width=4) (actual time=5.109..5.111 rows=1 loops=1)
7 [...] -> Seq Scan on employee employee_1 (cost=0.00..463.00 rows=14800 width=4) (actual time=0.392..4.126 rows=14800 loops=1)
8 [...] Filter: (dno = 5)
9 [...] Rows Removed by Filter: 1200
10 [...] -> Bitmap Index Scan on emp2 (cost=0.00..12.04 rows=6669 width=0) (actual time=5.169..5.169 rows=1280 loops=1)
11 [...] Index Cond: (salary > $0)
12 Planning Time: 0.398 ms
13 Execution Time: 7.517 ms

```

Execution plan and costs:



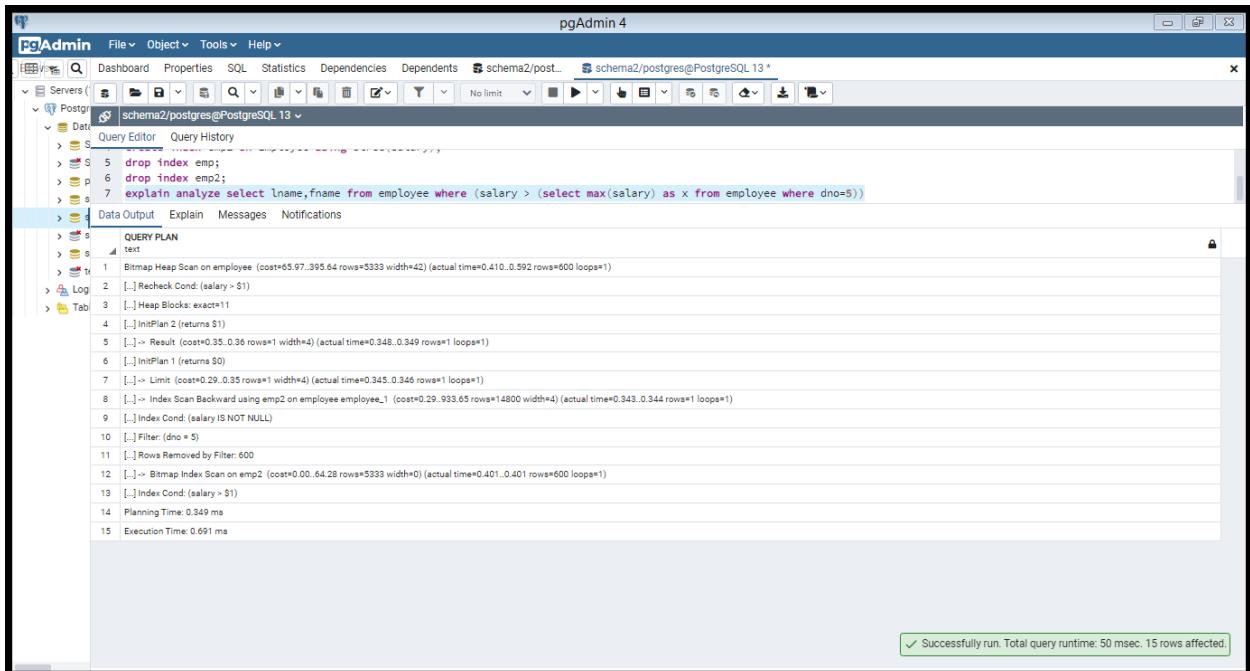
Here the BRIN index increased performance slightly as BRIN works in query that has small range in a big range so it was not that much effective

Query3 optimized using mixed indecies

Disabled flags: seqscan

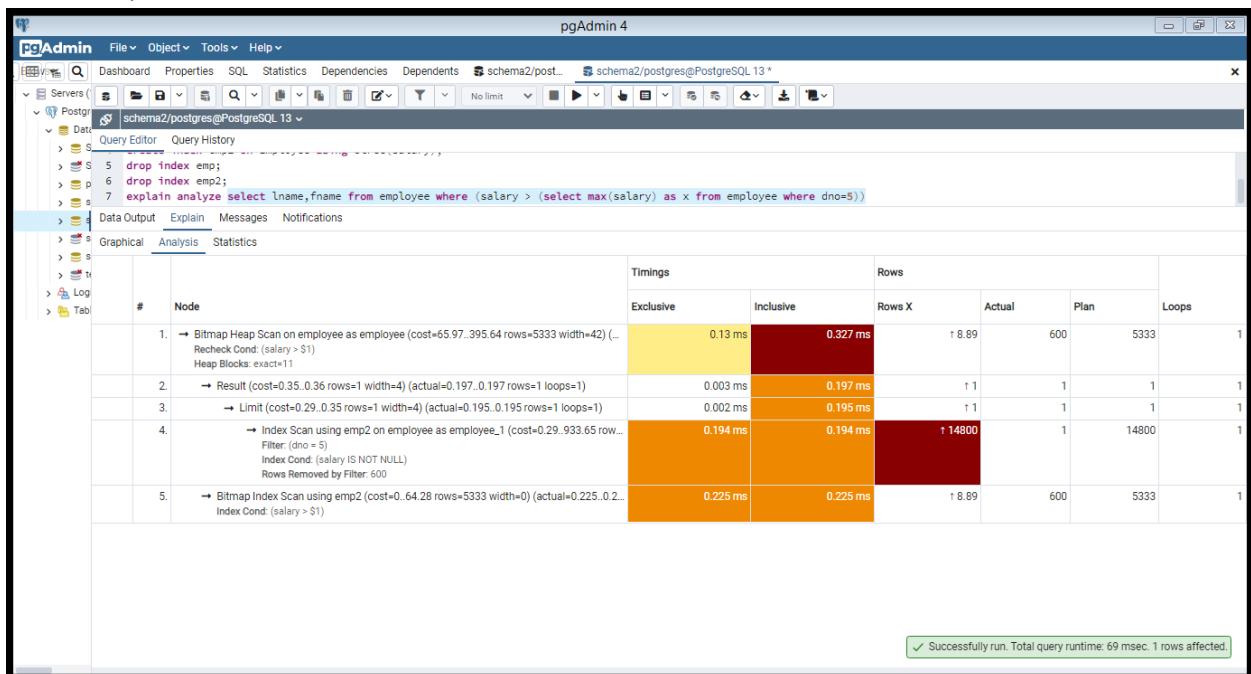
B+ tree on Employee(salary) and BRIN on Employee(dno)

Planning and execution time:



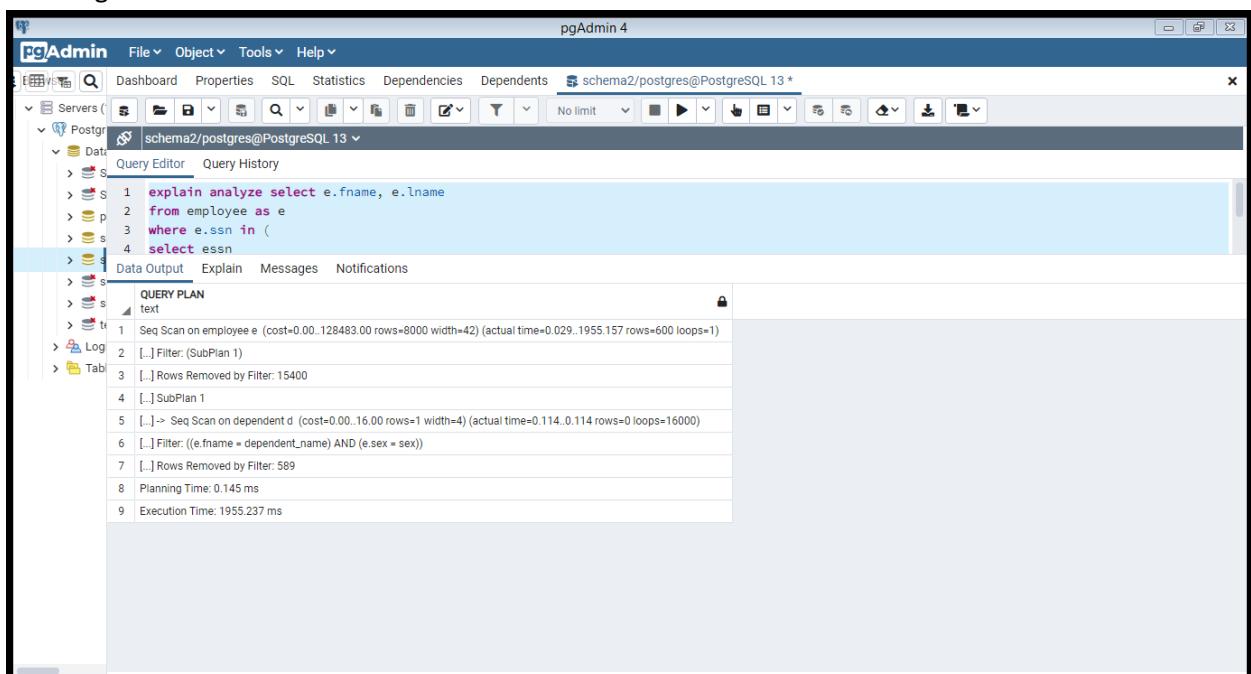
```
pgAdmin 4
File ▾ Object ▾ Tools ▾ Help ▾
PgAdmin File ▾ Object ▾ Tools ▾ Help ▾
Dashboard Properties SQL Statistics Dependencies Dependents schema2/post... schema2/postgres@PostgreSQL 13*
Servers ( 1 ) Data ( 1 ) Query Editor Query History
Postgres ( 1 ) schema2/postgres@PostgreSQL 13*
> S_ 5 drop index emp;
> p_ 6 drop index emp2;
> s_ 7 explain analyze select lname, fname from employee where (salary > (select max(salary) as x from employee where dno=5))
Data Output Explain Messages Notifications
QUERY PLAN
text
1 Bitmap Heap Scan on employee (cost=65.97..395.64 rows=5333 width=42) (actual time=0.410..0.592 rows=600 loops=1)
   1.1 Recheck Cond: (salary > $1)
   1.2 Heap Blocks: exact=11
   1.3 InitPlan 2 (returns $1)
   1.4 ...> Result (cost=0.35..0.36 rows=1 width=4) (actual time=0.348..0.349 rows=1 loops=1)
   1.5 ...> InitPlan 1 (returns $0)
   1.6 ...> Limit (cost=0.29..0.35 rows=1 width=4) (actual time=0.345..0.346 rows=1 loops=1)
   1.7 ...> Index Scan Backward using emp2 on employee_employee_1 (cost=0.29..933.65 rows=14800 width=4) (actual time=0.343..0.344 rows=1 loops=1)
   1.8 ...> Index Cond: (salary IS NOT NULL)
   1.9 ...> Filter: (dno = 5)
   1.10 ...> Rows Removed by Filter: 600
   1.11 ...> Bitmap Index Scan on emp2 (cost=0.00..64.28 rows=5333 width=0) (actual time=0.401..0.401 rows=600 loops=1)
   1.12 ...> Index Cond: (salary > $1)
Planning Time: 0.349 ms
Execution Time: 0.691 ms
Successfully run. Total query runtime: 50 msec. 15 rows affected.
```

Execution plan and costs:

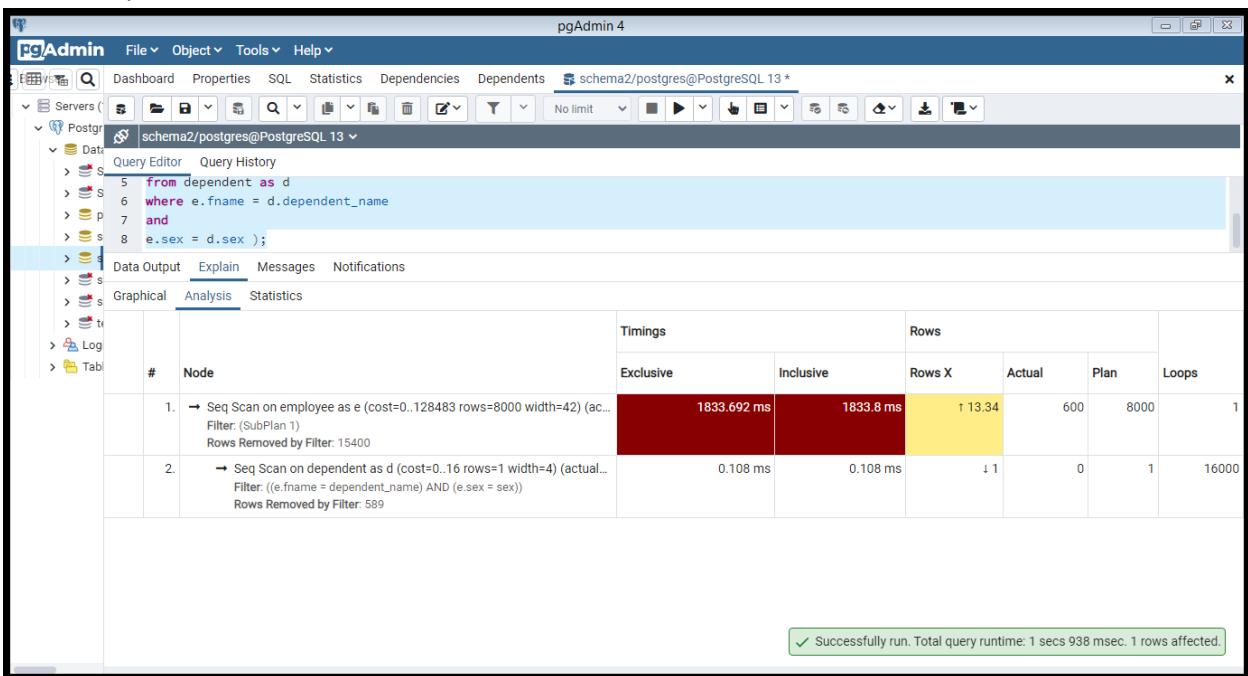


Query4 with no index

Planning and execution time:



Execution plan and costs:



Query 4 using B+ tree indecies on : Employee(sex), Employee(fname) ,
Dependent(dependent_name),Dependent(sex)

Disabled flags: seqscan and bitmap scan

Planning and execution time:

```

11 where e.fname = d.dependent_name
12 and
13 e.sex = d.sex ;

```

QUERY PLAN

text

1 Seq Scan on employee e (cost=0.00..69043.00 rows=8000 width=42) (actual time=0.088..127.462 rows=600 loops=1)

2 [...] Filter: (SubPlan 1)

3 [...] Rows Removed by Filter: 15400

4 [...] SubPlan 1

5 [...] -> Index Scan using dep2 on dependent d (cost=0.28..8.29 rows=1 width=4) (actual time=0.007..0.007 rows=0 loops=..)

6 [...] Index Cond: (dependent_name = e.fname)

7 [...] Filter: (e.sex = sex)

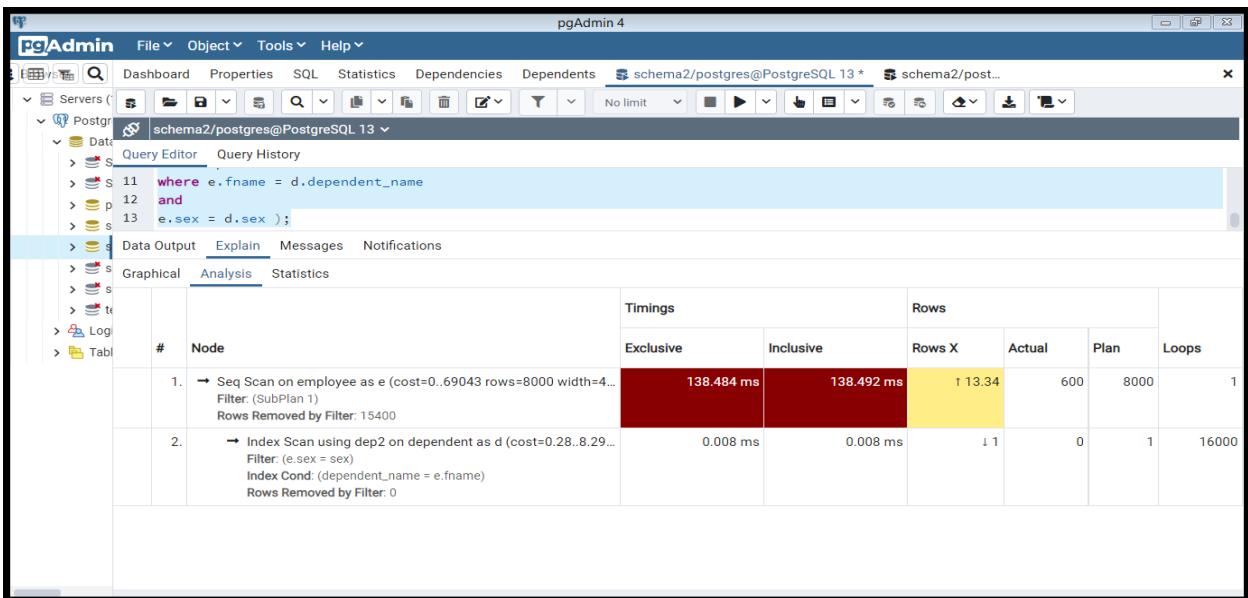
8 Planning Time: 0.216 ms

9 Execution Time: 127.549 ms

✓ Successfully run. Total query runtime: 171 msec. 9 rows affected.

✓ Query returned successfully in 42 msec.

Execution plan and costs:



Here the B+ tree helped in finding the dependent as it search in O(logn) time so it increase performance

Query4 using hash indecies on : Employee(sex), Employee(fname) ,
Dependent(dependent_name),Dependent(sex)

Disabled flags : seqscan and bitmap scan

Planning and execution time:

```

explain analyze select e.fname, e.lname
from employee as e
where e.ssn in (
    select essn
    from dependent as d
    where e.fname = d.dependent_name
    and e.sex = d.sex
);

```

QUERY PLAN

- Seq Scan on employee e (cost=0.00..64643.00 rows=8000 width=42) (actual time=0.047..27.880 rows=600 loops=1)
- [...] Filter: (SubPlan 1)
- [...] Rows Removed by Filter: 15400
- [...] SubPlan 1
- [...] Index Scan using dep2 on dependent d (cost=0.00..8.02 rows=1 width=4) (actual time=0.001..0.001 rows=0 loops=16000)
- [...] Index Cond: (dependent_name = e.fname)
- [...] Filter: (e.sex = sex)
- Planning Time: 0.268 ms
- Execution Time: 27.967 ms

Execution plan and costs:

#	Node	Timings		Rows			
		Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Seq Scan on employee e (cost=0..64643 rows=8000 width=42) (actual=0.034..23.495 r...	23.494 ms	23.495 ms	↑ 13.34	600	8000	1
2.	→ Index Scan using dep2 on dependent d (cost=0..8.02 rows=1 width=4) (actual=0.0...	0.001 ms	0.001 ms	↓ 1	0	1	16000

Here the hash improved the performance as it helps in searching for the dependent who has the same name and sex as it search in O(1) time

Query4 using BRIN indecies on : Employee(sex), Employee(name) ,
Dependent(dependent_name),Dependent(sex)

Planning and execution time:

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

- Servers:** PostgreSQL - schema2/postgres@PostgreSQL 13*
- Query Editor:** Contains the SQL query:


```
14 from dependent as d
15 where e.fname = d.dependent_name
16 and
17 e.sex = d.sex );
```
- Explain Tab:** Shows the query plan with numbered steps:
 - Seq Scan on employee e (cost=1000000000.00..10000264983.00 rows=8000 width=42) (actual time=0.039..680.431 rows=9 loops=1)
 - [...] Filter: (SubPlan 1)
 - [...] Rows Removed by Filter: 15400
 - [...] SubPlan 1
 - [...] Index Scan using dependent_pkey on dependent d (cost=0.28..32.79 rows=1 width=4) (actual time=0.042..0.042 rows=1 loops=1)
 - [...] Index Cond: (dependent_name = e.fname)
 - [...] Filter: (e.sex = sex)
 - Planning Time: 1.243 ms
 - Execution Time: 680.508 ms
- Status Bar:** Successfully run. Total query runtime: 701 msec. 9 rows affected.

Execution plan and costs:

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

- Servers:** PostgreSQL - schema2/postgres@PostgreSQL 13*
- Query Editor:** Contains the same SQL query as the previous screenshot.
- Analysis Tab:** Shows the execution plan in a graphical format with nodes and their properties:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Seq Scan on employee as e (cost=1000000000..10000264983 rows=80... Filter: (SubPlan 1) Rows Removed by Filter: 15400	670.1 ms	670.141 ms	113.34	600	8000	1
2.	→ Index Scan using dependent_pkey on dependent as d (cost=0.28..3... Filter: (e.sex = sex) Index Cond: (dependent_name = e.fname) Rows Removed by Filter: 0	0.041 ms	0.041 ms	1	0	1	16000
- Status Bar:** Successfully run. Total query runtime: 772 msec. 1 rows affected.

Here bin increased the cost and increased execution time as it is used in small range queries only

Query4 using mixed indecies : hash index on Employee(sex),Dependent(sex)
 B+ tree on Employee(fname) , Dependent(dependent_name)

Planning and execution time:

```

14 from dependent as d
15 where e.fname = d.dependent_name
16 and
17 e.sex = d.sex ;

```

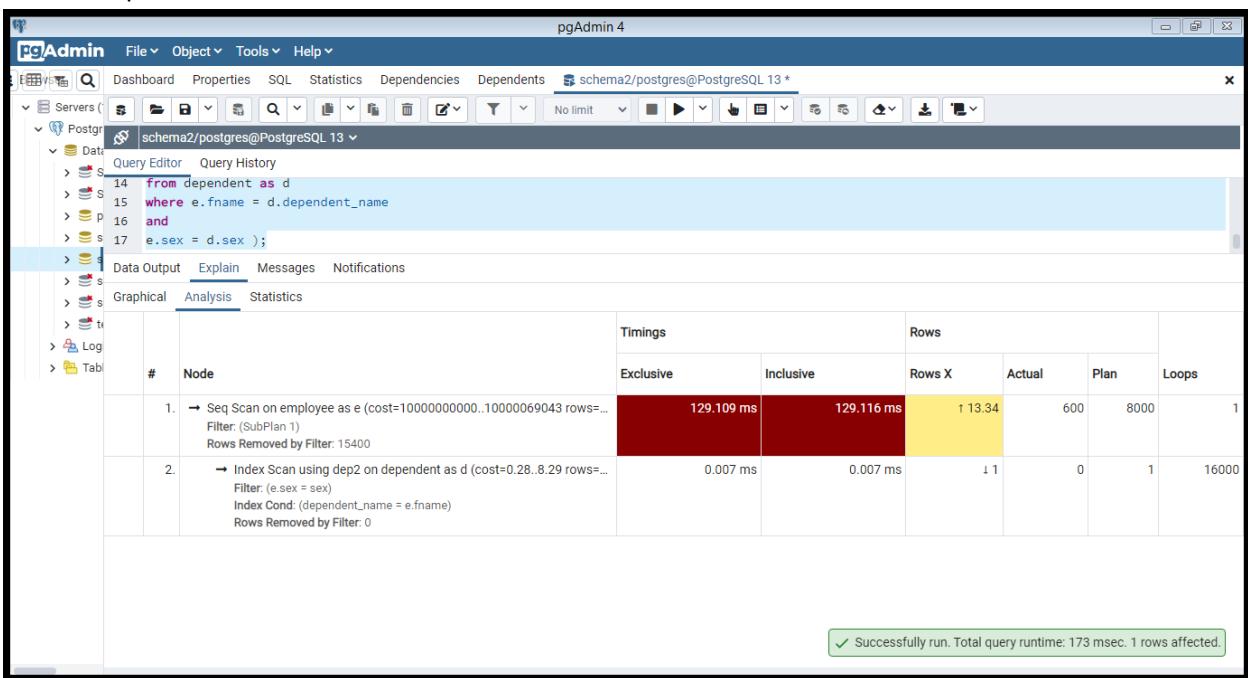
QUERY PLAN

text

1 Seq Scan on employee e (cost=1000000000.00..10000069043.00 rows=8000 width=42) (actual time=0.104..123.013 rows=8000 loops=1)
 2 [...] Filter: (SubPlan 1)
 3 [...] Rows Removed by Filter: 15400
 4 [...] SubPlan 1
 5 [...]-> Index Scan using dep2 on dependent d (cost=0.28..8.29 rows=1 width=4) (actual time=0.007..0.007 rows=0 loops=1)
 6 [...] Index Cond: (dependent_name = e.fname)
 7 [...] Filter: (e.sex = sex)
 8 Planning Time: 1.796 ms
 9 Execution Time: 123.113 ms

Successfully run. Total query runtime: 165 msec. 9 rows affected.

Execution plan and costs:



The indecies in this query increased performance but it also increased the cost too much and when I use seqscans only the execution time increase and cost decrease

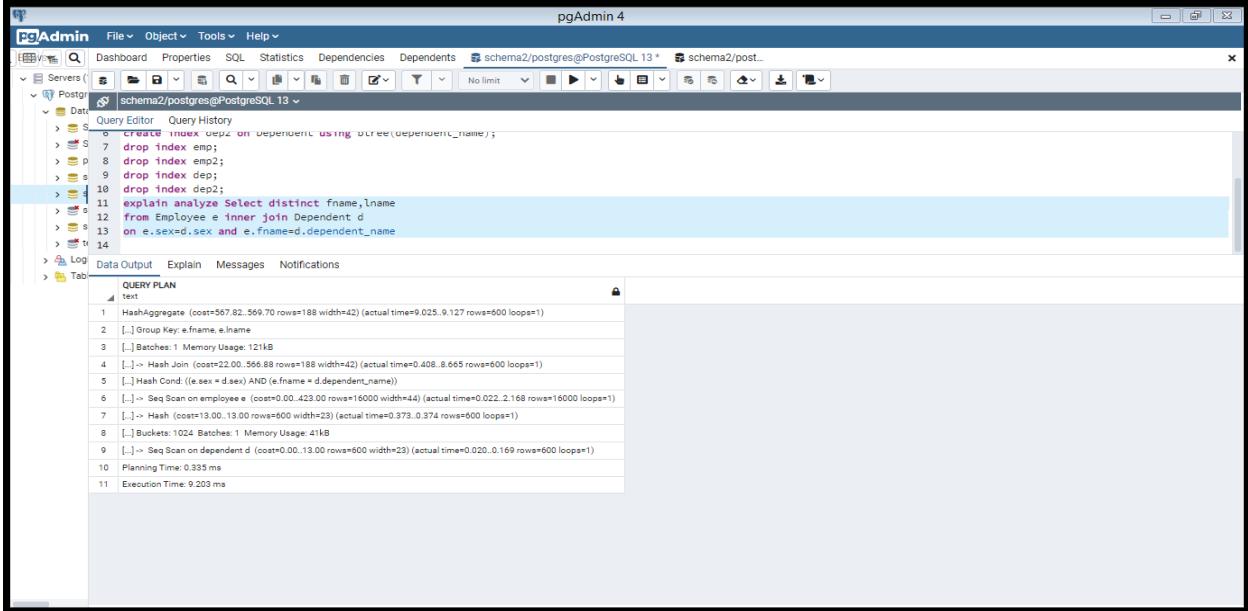
Query4 after optimization :

Select distinct fname,lname

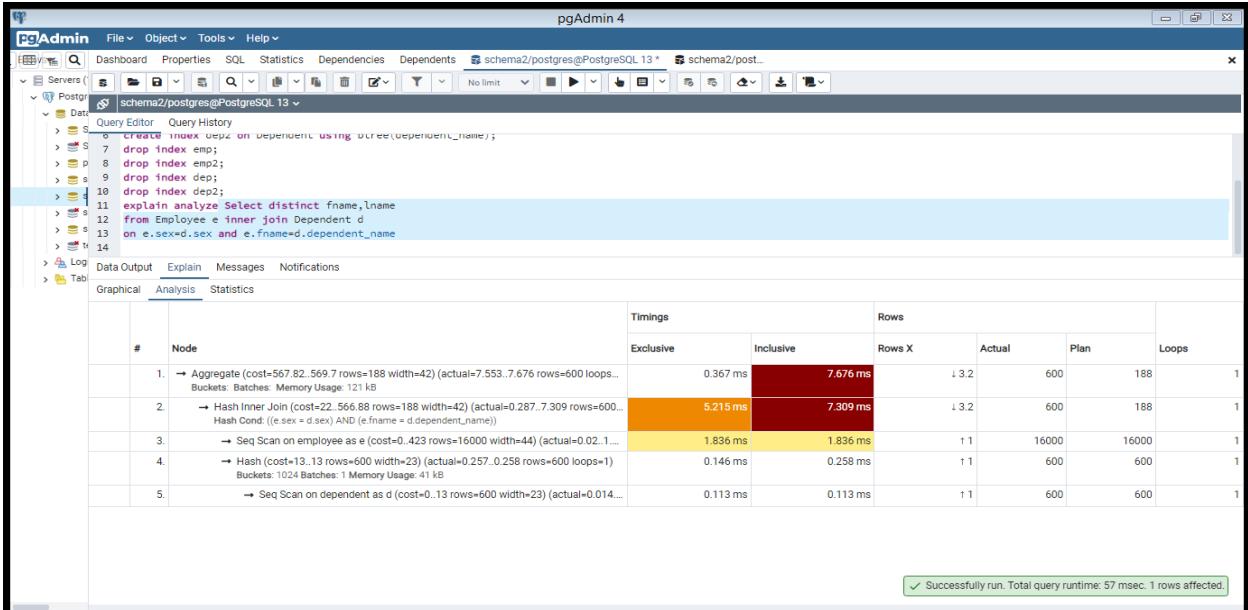
```
from Employee e inner join Dependent d  
on e.sex=d.sex and e.fname=d.dependent_name  
  
we replaced the sub query with inner join as it is faster
```

Query4 optimized with no index

Planning and execution time:



Execution plan and costs:



Query4 optimized using B+ tree on: Dependent(dependent_name),Dependent(sex), Employee(fname), Employee(sex)

Disabled flags: seqscan and bitmapscan

Planning and execution time:

```

pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents schema2/postgres@PostgreSQL 13* schema2/post...
Servers (1) PostgreSQL (1) Data (1) schema2/postgres@PostgreSQL 13 (1)
Query Editor Query History
> S 6 create index empz on dependent using btree(dependent_name);
> S 7 drop index emp;
> P 8 drop index emp2;
> S 9 drop index dep;
> S 10 drop index dep2;
> E 11 explain analyze Select distinct fname,lname
> S 12 from Employee e inner join Dependent d
> S 13 on e.sex=d.sex and e.fname=d.dependent_name
> T 14
Log Data Output Explain Messages Notifications
Tab: QUERY PLAN
text
1 HashAggregate (cost=1449.58..1451.46 rows=188 width=42) (actual time=8.757..8.860 rows=600 loops=1)
[...] Group Key: e.fname, e.name
3 [...] Batches: 1 Memory Usage: 121kB
4 [...] -> Hash Join (cost=36.44..1448.84 rows=188 width=42) (actual time=5.815..8.447 rows=600 loops=1)
5 [...] Hash Cond: ((e.sex = d.sex) AND (e.fname = d.dependent_name))
6 [...] -> Index Scan using emp on employee e (cost=0.29..1290.61 rows=16000 width=44) (actual time=0.061..3.476 rows=1...
7 [...] -> Hash (cost=27.15..27.15 rows=600 width=23) (actual time=0.365..0.365 rows=600 loops=1)
8 [...] Buckets: 1024 Batches: 1 Memory Usage: 41kB
9 [...] -> Index Scan using dep on dependent d (cost=0.15..27.15 rows=600 width=23) (actual time=0.027..0.182 rows=600 lo...
10 Planning Time: 1.248 ms
11 Execution Time: 8.940 ms

```

Execution plan and costs:

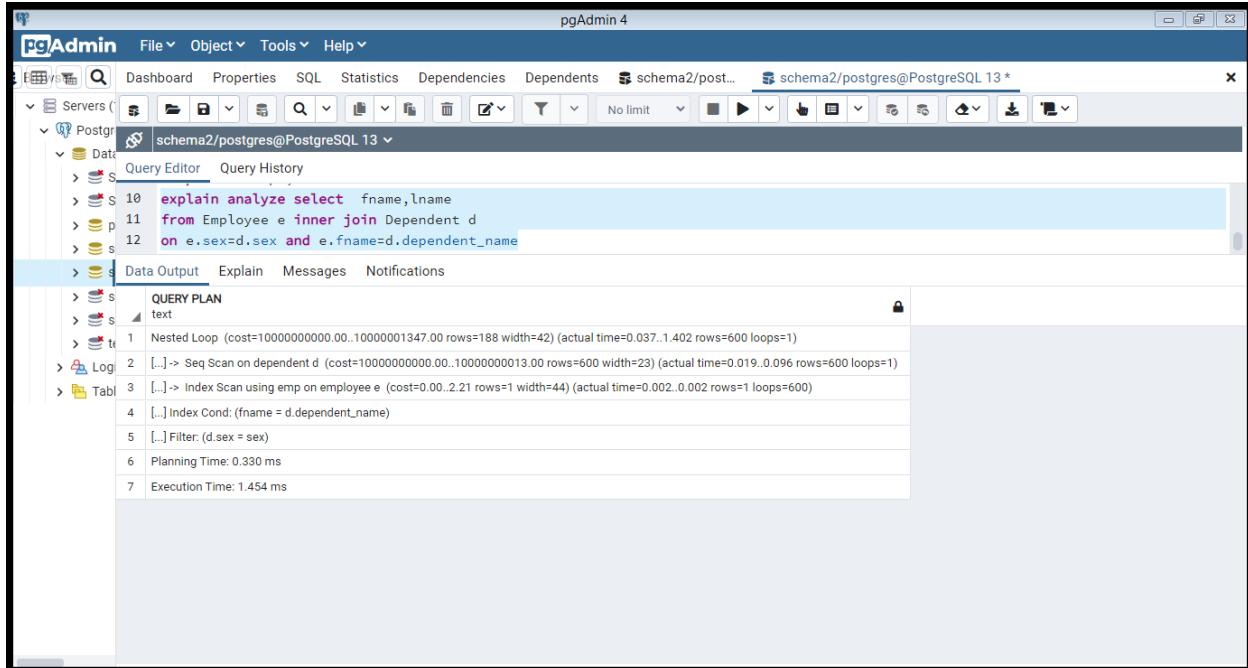
#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Aggregate (cost=1449.58..1451.46 rows=188 width=42) (actual=14.006..14.11 rows=600 ... Buckets: Batches: Memory Usage: 121kB	0.607 ms	14.11 ms	↓ 3.2	600	188	1
2.	→ Hash Inner Join (cost=36.44..1448.64 rows=188 width=42) (actual=9.253..13.503 ro... Hash Cond: ((e.sex = d.sex) AND (e.fname = d.dependent_name))	7.113 ms	13.503 ms	↓ 3.2	600	188	1
3.	→ Index Scan using emp on employee e (cost=0.29..1290.61 rows=16000 width=44) (actual time=0.061..3.476 rows=1...	5.972 ms	5.972 ms	↑ 1	16000	16000	1
4.	→ Hash (cost=27.15..27.15 rows=600 width=23) (actual time=0.365..0.365 rows=600 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 41kB	0.198 ms	0.418 ms	↑ 1	600	600	1
5.	→ Index Scan using dep on dependent d (cost=0.15..27.15 rows=600 width=23) (actual time=0.027..0.182 rows=600 loops=1)	0.22 ms	0.22 ms	↑ 1	600	600	1

here the B+ tree did not improve the execution time that much because in the join it needs to check every row from 1 table with all rows of the other tables and it also checks on 2 conditions sex and name so it did not help that much

Query4 optimized using hash indecies on : Employee(sex), Employee(fname)

Disabled flags: seqscan and bitmap scan

Planning and execution time:



The screenshot shows the pgAdmin 4 interface with the 'Explain' tab selected. The query being analyzed is:

```
explain analyze select fname, lname
from Employee e inner join Dependent d
on e.sex=d.sex and e.fname=d.dependent_name
```

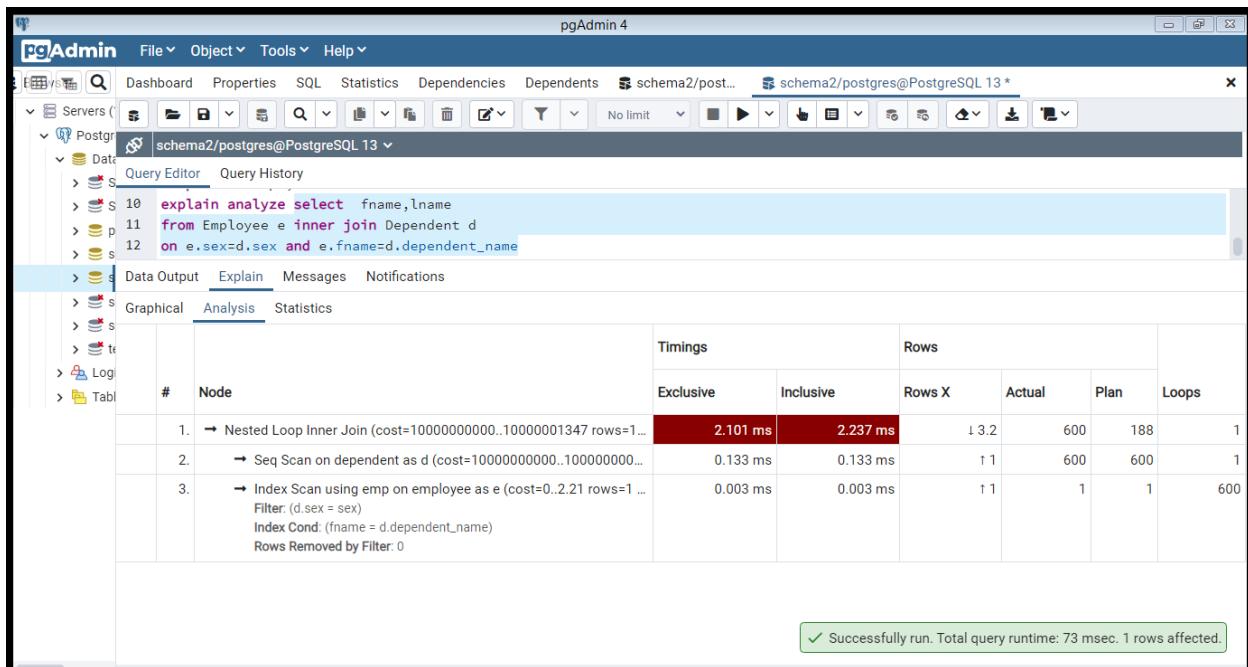
The 'QUERY PLAN' section displays the execution plan:

- Step 1: Nested Loop (cost=1000000000.00..10000001347.00 rows=188 width=42) (actual time=0.037..1.402 rows=600 loops=1)
 - Step 2: Seq Scan on dependent d (cost=1000000000.00..1000000013.00 rows=600 width=23) (actual time=0.019..0.096 rows=600 loops=1)
 - Step 3: Index Scan using emp on employee e (cost=0.00..2.21 rows=1 width=44) (actual time=0.002..0.002 rows=1 loops=600)
 - Step 4: Index Cond: (fname = d.dependent_name)
 - Step 5: Filter: (d.sex = sex)

Timing details:

 - Planning Time: 0.330 ms
 - Execution Time: 1.454 ms

Execution plan and costs:



The screenshot shows the pgAdmin 4 interface with the 'Explain' tab selected. The query is the same as before:

```
explain analyze select fname, lname
from Employee e inner join Dependent d
on e.sex=d.sex and e.fname=d.dependent_name
```

The 'Graphical' tab is selected, showing a tree diagram of the execution plan. The 'Analysis' tab is also visible.

The 'Timings' table provides detailed timing information for each node:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=1000000000..10000001347 rows=1...	2.101 ms	2.237 ms	13.2	600	188	1
2.	→ Seq Scan on dependent d (cost=1000000000..100000000...	0.133 ms	0.133 ms	1 1	600	600	1
3.	→ Index Scan using emp on employee e (cost=0..2.21 rows=1 ... Filter: (d.sex = sex) Index Cond: (fname = d.dependent_name) Rows Removed by Filter: 0	0.003 ms	0.003 ms	1 1	1	1	600

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 73 msec. 1 rows affected."

Here the cost increased too much as I disabled seqscan but the performance increased

With seqscan enabled:

Planning and execution time:

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Servers (1) PostgreSQL schema2/postgres@PostgreSQL 13*

Data (1) S (1) P (1) T (1) Log (1) Tab (1)

Query Editor Explain Messages Notifications

```

> S 4 From Employee e inner join Dependent d
> S 5 on e.sex=d.sex and e.fname=d.dependent_name
> S 6

```

DATA OUTPUT QUERY PLAN text

1 HashAggregate (cost=567.82..569.70 rows=188 width=42) (actual time=8.562..8.749 rows=600 loops=1)
 2 [...] Group Key: e.fname, e.name
 3 [...] Batches: 1 Memory Usage: 121kB
 4 [...] Hash Join (cost=22.00..566.88 rows=188 width=42) (actual time=0.506..8.126 rows=600 loops=1)
 5 [...] Hash Cond: ((e.sex = d.sex) AND (e.fname = d.dependent_name))
 6 [...] > Seq Scan on employee e (cost=0.00..423.00 rows=16000 width=44) (actual time=0.019..1.940 rows=16000 loops=1)
 7 [...] > Hash (cost=13.00..13.00 rows=600 width=23) (actual time=0.471..0.473 rows=600 loops=1)
 8 [...] Buckets: 1024 Batches: 1 Memory Usage: 41kB
 9 [...] > Seq Scan on dependent d (cost=0.00..13.00 rows=600 width=23) (actual time=0.019..0.196 rows=600 loops=1)
 10 Planning Time: 1.255 ms
 11 Execution Time: 8.846 ms

Successfully run. Total query runtime: 4 secs 805 msec. 9 rows affected.

Query returned successfully in 47 msec.

Query returned successfully in 46 msec.

Execution plan and costs:

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Servers (1) PostgreSQL schema2/postgres@PostgreSQL 13*

Data (1) S (1) P (1) T (1) Log (1) Tab (1)

Query Editor Explain Statistics

```

> S 4 From Employee e inner join Dependent d
> S 5 on e.sex=d.sex and e.fname=d.dependent_name
> S 6

```

Graphical Analysis Statistics

#	Node	Timings		Rows			
		Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Aggregate (cost=567.82..569.70 rows=188 width=42) (actual=7.487..7.594 rows=6... Buckets: Batches: Memory Usage: 121 kB	0.342 ms	7.594 ms	1 3.2	600	188	1
2.	→ Hash Inner Join (cost=22.566.88 rows=188 width=42) (actual=0.299..7.253 r... Hash Cond: ((e.sex = d.sex) AND (e.fname = d.dependent_name))	5.11 ms	7.253 ms	1 3.2	600	188	1
3.	→ Seq Scan on employee e (cost=0..423 rows=16000 width=44) (actual... Rows X: 16000	1.875 ms	1.875 ms	1 1	16000	16000	1
4.	→ Hash (cost=13..13 rows=600 width=23) (actual=0.267..0.268 rows=600 l... Buckets: 1024 Batches: 1 Memory Usage: 41 kB	0.148 ms	0.268 ms	1 1	600	600	1
5.	→ Seq Scan on dependent d (cost=0..13 rows=600 width=23) (actu... Rows X: 600	0.121 ms	0.121 ms	1 1	600	600	1

Successfully run. Total query runtime: 56 msec. 1 rows affected.

Successfully run. Total query runtime: 4 secs 805 msec. 9 rows affected.

Query returned successfully in 47 msec.

Query returned successfully in 46 msec.

Query4 optimized using BRIN indecies on : Employee(sex), Employee(fname) ,
Dependent(dependent_name),Dependent(sex)

With seqscan enabled:

Planning and execution time:

The screenshot shows the pgAdmin 4 interface. In the top navigation bar, 'File', 'Object', 'Tools', and 'Help' are visible. Below the navigation bar, the 'Servers' tree shows a connection to 'schema2/postgres@PostgreSQL 13'. The 'Query Editor' tab is active, displaying a series of SQL commands for creating and dropping indexes on tables 'Employee' and 'Dependent'. The 'Explain Plan' tab is also visible, showing the execution plan for the query. The plan details the HashAggregate operation, Hash Join, Hash Cond, and Seq Scan steps, along with their costs, actual times, and row counts.

```
pgAdmin 4
File ▾ Object ▾ Tools ▾ Help ▾
Servers (1) [schema2/postgres@PostgreSQL 13]
  Data [schema2/postgres@PostgreSQL 13]
    Query Editor Explain Plan
      3
      4 create index emp on Employee using brin(sex);
      5 create index emp2 on Employee using brin(fname);
      6 create index dep on Dependent using brin(sex);
      7 create index dep2 on Dependent using brin(dependent_name);
      8 drop index emp;
      9 drop index emp2;
     10 drop index dep;
     11 \d+ index dep2;
      Log Data Output Explain Messages Notifications
      Tab QUERY PLAN
      text
      1 HashAggregate (cost=567.82..569.70 rows=188 width=42) (actual time=7.311..7.534 rows=600 loops=1)
      2 [...] Group Key: e.fname, e.name
      3 [...] Batches: 1 Memory Usage: 121 kB
      4 [...] >> Hash Join (cost=22.00..566.88 rows=188 width=42) (actual time=0.356..7.059 rows=600 loops=1)
      5 [...] Hash Cond: ((e.sex = d.sex) AND (e.fname = d.dependent_name))
      6 [...] >> Seq Scan on employee e (cost=0.00..423.00 rows=16000 width=44) (actual time=0.017..1.752 rows=16000 loops=1)
      7 [...] >> Hash (cost=13.00..13.00 rows=600 width=23) (actual time=0.329..0.332 rows=600 loops=1)
      8 [...] Buckets: 1024 Batches: 1 Memory Usage: 41 kB
      9 [...] >> Seq Scan on dependent d (cost=0.00..13.00 rows=600 width=23) (actual time=0.014..0.135 rows=600 loops=1)
      10 Planning Time: 0.354 ms
      11 Execution Time: 7.620 ms
```

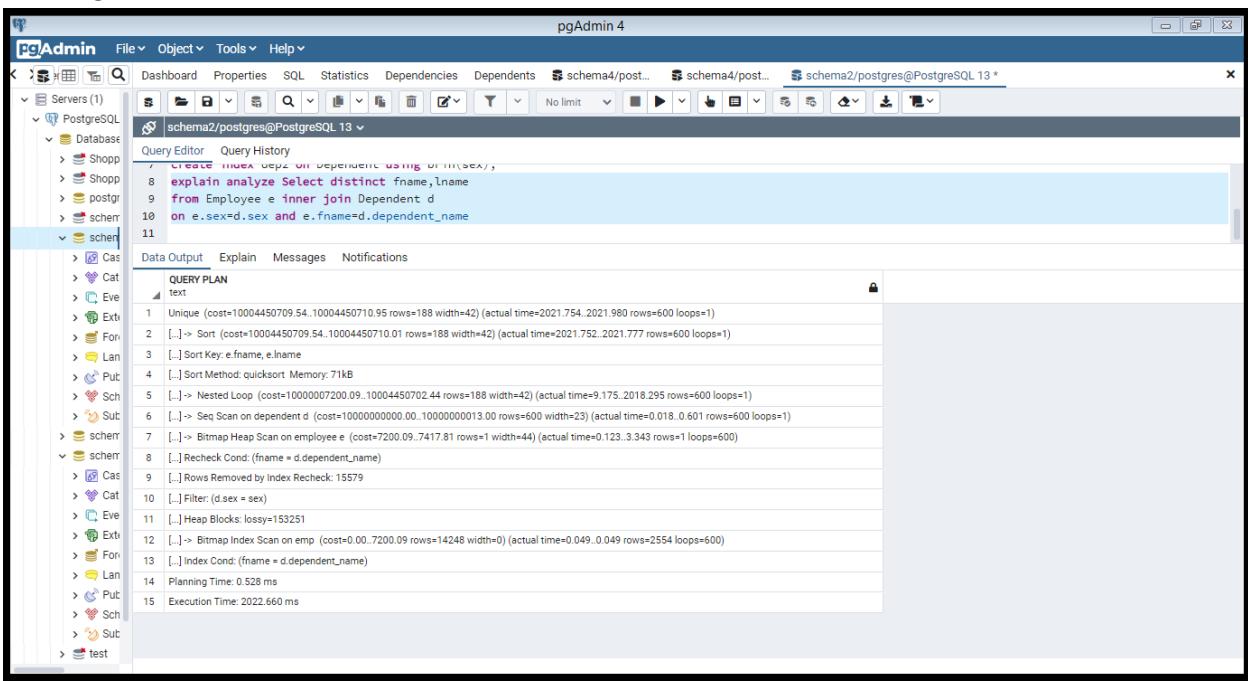
execution plan and costs:

The screenshot shows the pgAdmin 4 interface with the 'Explain Analysis' tab selected. The 'Servers' tree and 'Query Editor' tab are visible. The 'Explain Analysis' tab displays a detailed execution plan with timing information for each node. The plan shows the Hash Inner Join, Hash Cond, Seq Scan on employee, and Seq Scan on dependent operations, with their respective costs, exclusive times, inclusive times, and row counts.

```
pgAdmin 4
File ▾ Object ▾ Tools ▾ Help ▾
Servers (1) [schema2/postgres@PostgreSQL 13]
  Data [schema2/postgres@PostgreSQL 13]
    Query Editor Explain Analysis
      7 create index dep2 on Dependent using brin(dependent_name);
      8 drop index emp;
      9 drop index emp2;
     10 drop index dep;
     11 drop index dep2;
      12 explain analyze Select distinct fname,lname
      13 from Employee e inner join Dependent d
      14 on e.sex=d.sex and e.fname=d.dependent_name
      15
      Log Data Output Explain Messages Notifications
      Tab Graphical Analysis Statistics
      Node Timings Rows
      # Node
      1. → Aggregate (cost=567.82..569.7 rows=188 width=42) (actual=8.566..8.812 rows=600 loops=1)
      Buckets: 1024 Batches: 1 Memory Usage: 121 kB
      Exclusive Inclusive Rows X Actual Plan Loops
      0.481 ms 8.812 ms 1 3.2 600 188 1
      2. → Hash Inner Join (cost=22..566.88 rows=188 width=42) (actual=0.296..8.331 rows=600...)
      Hash Cond: ((e.sex = d.sex) AND (e.fname = d.dependent_name))
      Exclusive Inclusive Rows X Actual Plan Loops
      5.899 ms 8.331 ms 1 3.2 600 188 1
      3. → Seq Scan on employee as e (cost=0..423 rows=16000 width=44) (actual=0.018..2...
      Exclusive Inclusive Rows X Actual Plan Loops
      2.161 ms 2.161 ms 1 1 16000 16000 1
      4. → Hash (cost=13..13 rows=600 width=23) (actual=0.267..0.271 rows=600 loops=1)
      Hash Cond: ((e.sex = d.sex) AND (e.fname = d.dependent_name))
      Buckets: 1024 Batches: 1 Memory Usage: 41 kB
      Exclusive Inclusive Rows X Actual Plan Loops
      0.158 ms 0.271 ms 1 1 600 600 1
      5. → Seq Scan on dependent as d (cost=0..13 rows=600 width=23) (actual=0.015..0.015...
      Exclusive Inclusive Rows X Actual Plan Loops
      0.114 ms 0.114 ms 1 1 600 600 1
```

With seq scan disabled :

Planning and execution time:



The screenshot shows the pgAdmin 4 interface with a query editor window. The query being run is:

```
1 Create index depz on Dependent using brin(sex);
2 explain analyze Select distinct fname, lname
3   from Employee e inner join Dependent d
4     on e.sex=d.sex and e.fname=d.dependent_name
```

The query plan (text) is as follows:

- 1 Unique (cost=10004450709.54..10004450710.95 rows=188 width=42) (actual time=2021.754..2021.980 rows=600 loops=1)
- 2 [...] > Sort (cost=10004450709.54..10004450710.01 rows=188 width=42) (actual time=2021.752..2021.777 rows=600 loops=1)
- 3 [...] Sort Key: e.fname, e.lname
- 4 [...] Sort Method: quicksort Memory: 71kB
- 5 [...] > Nested Loop (cost=10000007200.09..10004450702.44 rows=188 width=42) (actual time=9.175..2018.295 rows=600 loops=1)
- 6 [...] > Seq Scan on dependent d (cost=10000000000.00..10000000013.00 rows=600 width=23) (actual time=0.018..0.601 rows=600 loops=1)
- 7 [...] > Bitmap Heap Scan on employee e (cost=7200.09..7417.81 rows=1 width=44) (actual time=0.123..3.343 rows=1 loops=600)
- 8 [...] Recheck Cond: (fname = d.dependent_name)
- 9 [...] Rows Removed by Index Recheck: 15579
- 10 [...] Filter: (d.sex = sex)
- 11 [...] Heap Blocks: lossy=153251
- 12 [...] > Bitmap Index Scan on emp (cost=0.00..7200.09 rows=14248 width=0) (actual time=0.049..0.049 rows=2554 loops=600)
- 13 [...] Index Cond: (fname = d.dependent_name)
- 14 Planning Time: 0.528 ms
- 15 Execution Time: 2022.660 ms

Both the cost and execution time increased too much as brin is used in small range queries only so it made it worse in this query

Query4 optimized using mixed indecies :

Hash index on Employee(sex), Employee(name) and B+ tree on Dependent(sex), Dependent(dependent_name)

Disabled flags: seqscan and bitmapscan

Planning and execution time:

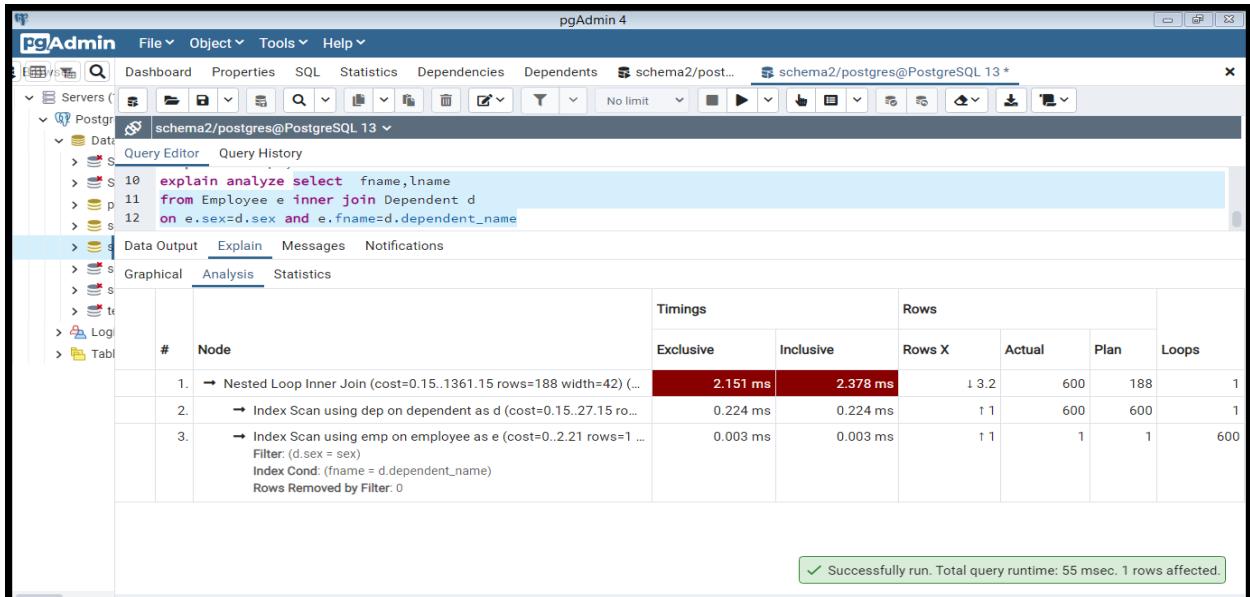
```

explain analyze select fname, lname
from Employee e inner join Dependent d
on e.sex=d.sex and e.fname=d.dependent_name;

```

Step	Operation	Cost	Actual Time
1	Nested Loop	(cost=0.15..1361.15 rows=188 width=42)	(actual time=0.019..1.430 rows=600 loops=1)
2	Index Scan using dep on dependent d	(cost=0.15..27.15 rows=600 width=23)	(actual time=0.007..0.128 rows=600 loops=1)
3	Index Scan using emp on employee e	(cost=0.00..2.21 rows=1 width=44)	(actual time=0.002..0.002 rows=1 loops=600)
4	[...] Index Cond: (fname = d.dependent_name)		
5	[...] Filter: (d.sex = sex)		
6	Planning Time: 0.376 ms		
7	Execution Time: 1.473 ms		

Execution plan and costs:



Here the hash increased performance as it helped in searching for the dependent as it search in O(1) time

Query5 with no index

Planning and execution time:

```

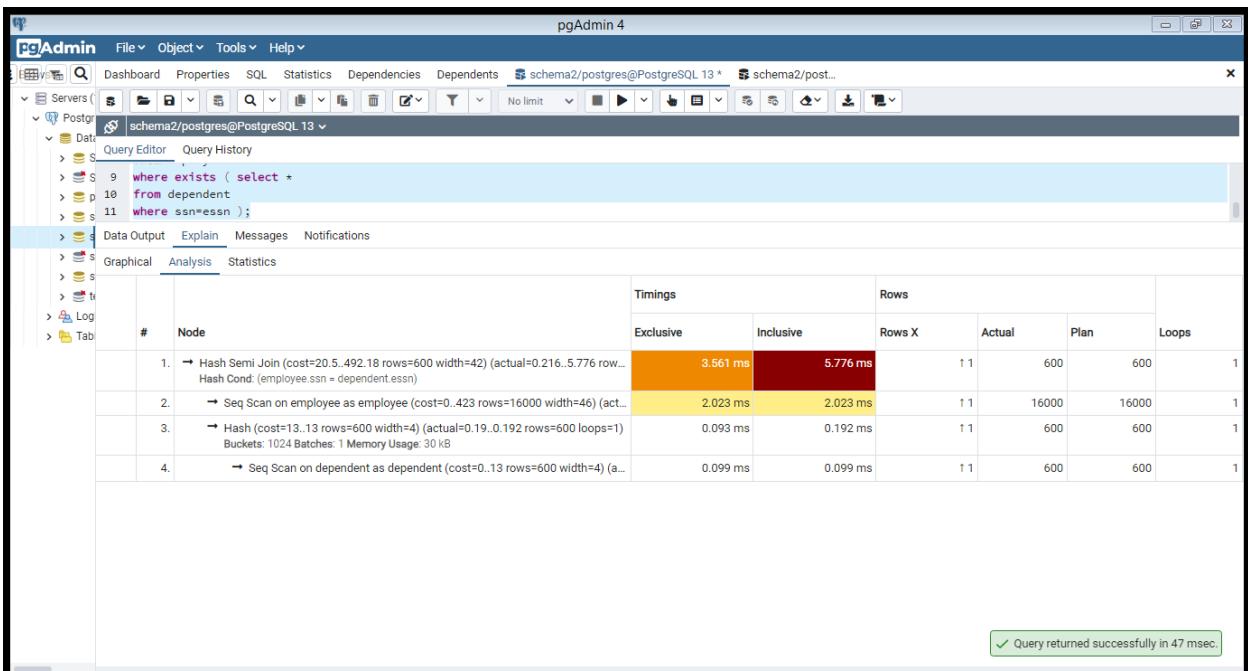
pgAdmin 4

File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents schema2/postgres@PostgreSQL 13 * schema2/post...
Servers PostgreSQL Data Query Editor Query History
> S 9 where exists ( select *
> S p 10 from dependent
> S 11 where ssn=essn );
> S Data Output Explain Messages Notifications
> S QUERY PLAN
> S text
1 Hash Semi Join (cost=20.50..492.18 rows=600 width=42) (actual time=0.323..6.378 rows=600 loops=1)
   2 [...] Hash Cond: (employee.ssn = dependent.essn)
      3 [...]> Seq Scan on employee (cost=0.00..423.00 rows=16000 width=46) (actual time=0.019..2.103 rows=16000 loops=1)
         4 [...]> Hash (cost=13.00..13.00 rows=600 width=4) (actual time=0.294..0.295 rows=600 loops=1)
            5 [...] Buckets: 1024 Batches: 1 Memory Usage: 30kB
            6 [...]> Seq Scan on dependent (cost=0.00..13.00 rows=600 width=4) (actual time=0.016..0.139 rows=600 loops=1)
            7 Planning Time: 0.297 ms
            8 Execution Time: 6.442 ms
> S

```

✓ Query returned successfully in 47 msec.

Execution plan and costs:



Query5 with B+ tree indecies on : Dependent(ssn),Employee(ssn)

Disabled flags: seqscan and bitmap scan

Planning and execution time:

The screenshot shows the pgAdmin 4 interface with the query editor open. The query is:

```
9 where exists ( select *
  10   from dependent
  11   where ssn=essn );
```

The Explain tab displays the execution plan:

- 1 Merge Semi Join (cost=0.56..70.44 rows=600 width=42) (actual time=0.035..0.546 rows=600 loops=1)
 - 2 [...] Merge Cond: (employee.ssn = dependent.ssn)
- 3 [...] Index Scan using emp on employee (cost=0.29..690.28 rows=16000 width=46) (actual time=0.010..0.161 rows=601 loops=1)
 - 4 [...] Index Only Scan using dep on dependent (cost=0.28..35.27 rows=600 width=4) (actual time=0.021..0.193 rows=600 loops=1)
 - 5 [...] Heap Fetches: 600
- 6 Planning Time: 1.138 ms
- 7 Execution Time: 0.586 ms

A green message bar at the bottom right says "Query returned successfully in 47 msec."

Execution plan and costs:

The screenshot shows the pgAdmin 4 interface with the query editor open. The query is the same as above.

The Explain tab shows the graphical execution plan with the following steps:

1. → Merge Semi Join (cost=0.56..70.44 rows=600 width=42) (actual=0.025..0.473 rows=600 width=42)
2. → Index Scan using emp on employee as employee (cost=0.29..690.28 rows=16000 width=46) (actual=0.119..0.119 rows=601 width=46)
3. → Index Only Scan using dep on dependent as dependent (cost=0.28..35.27 rows=600 width=4) (actual=0.163..0.163 rows=600 width=4)

The Analysis tab shows a detailed table of timings and row counts:

#	Node	Timings		Rows			
		Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Merge Semi Join (cost=0.56..70.44 rows=600 width=42) (actual=0.025..0.473 rows=600 width=42)	0.191 ms	0.473 ms	1 1	600	600	1
2.	→ Index Scan using emp on employee as employee (cost=0.29..690.28 rows=16000 width=46) (actual=0.119..0.119 rows=601 width=46)	0.119 ms	0.119 ms	1 26.63	601	16000	1
3.	→ Index Only Scan using dep on dependent as dependent (cost=0.28..35.27 rows=600 width=4) (actual=0.163..0.163 rows=600 width=4)	0.163 ms	0.163 ms	1 1	600	600	1

A green message bar at the bottom right says "Successfully run. Total query runtime: 58 msec. 1 rows affected." and another says "Query returned successfully in 47 msec."

here the B+ tree helped in this query as it search in $O(\log n)$ time

Query5 using hash indecies on : Dependent(ssn) , Employee(ssn)

Disabled flags: seqscan and bitmapscan

Planning and execution time:

The screenshot shows the pgAdmin 4 interface with the 'Explain' tab selected. The query plan details the execution steps, including a Nested Loop, HashAggregate, and various index scans. The 'Messages' tab at the bottom right displays three green success messages: 'Successfully run. Total query runtime: 4 secs 805 msec. 9 rows affected.', 'Query returned successfully in 47 msec.', and 'Query returned successfully in 46 msec.'

Execution plan and costs:

The screenshot shows the pgAdmin 4 interface with the 'Graphical' tab selected. It displays a detailed execution plan with four nodes, each showing its cost, time, and other metrics. The 'Messages' tab at the bottom right shows three green success messages: 'Successfully run. Total query runtime: 4 secs 805 msec. 9 rows affected.', 'Query returned successfully in 47 msec.', and 'Query returned successfully in 46 msec.'

Here the hash improved performance as hash is used in exact value queries as it search in O(1) it helped in the join using the ssn

Query5 using BRIN indecies on : Dependent(ssn) , Employee(ssn)

With seqscan disabled

Planning and execution time:

The screenshot shows the pgAdmin 4 interface. In the top navigation bar, 'File', 'Object', 'Tools', and 'Help' are visible. Below the navigation bar, the 'Servers' tree shows a connection to 'PostgreSQL' and 'schema2/postgres@PostgreSQL 13'. The main area has tabs for 'Query Editor' (selected), 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Query Editor' tab contains the following SQL code:

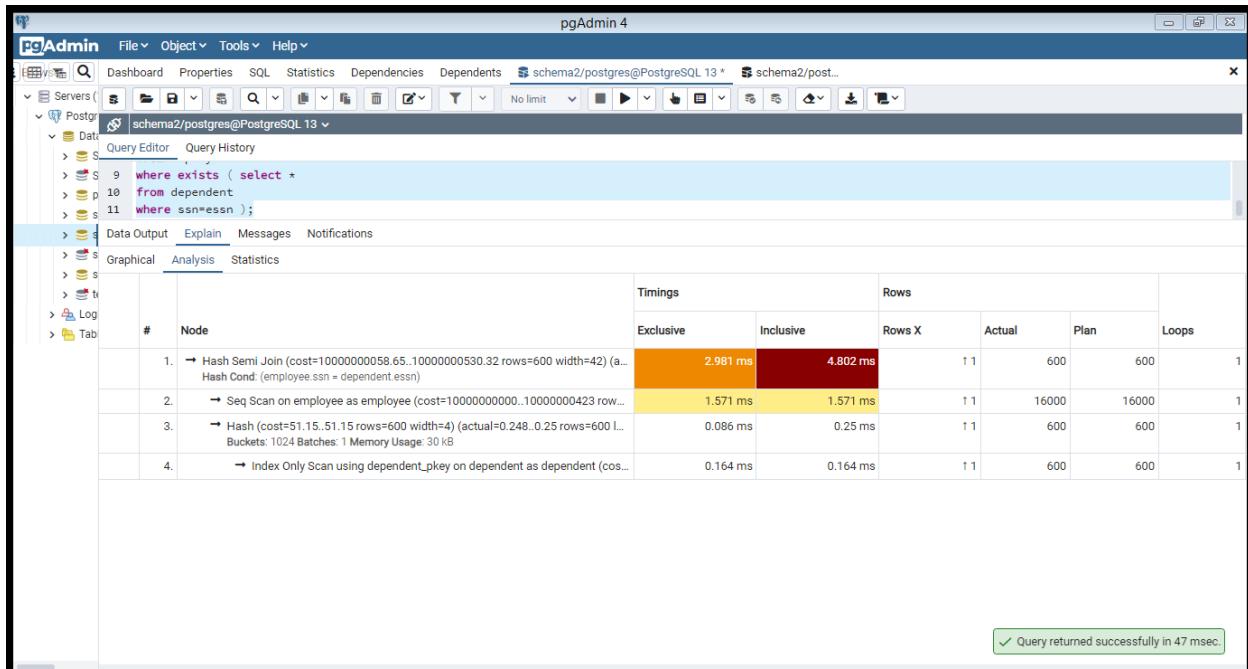
```
> S 9 where exists ( select *  
>   p 10 from dependent  
>   S 11 where ssn=essn );
```

Below the code, the 'Data Output' tab is selected, showing the execution plan:

Step	Operation	Cost	Rows	Width	Time
1	Hash Semi Join (cost=10000000058.65..10000000530.32 rows=600 width=42)	0.267..3.835	600	600	loops=1
2	[...] Hash Cond: (employee.ssn = dependent.essn)				
3	[...] Seq Scan on employee (cost=10000000000.00..10000000423.00 rows=16000 width=46)	0.014..1.223	16000	16000	loops=1
4	[...] > Hash (cost=51.15..51.15 rows=600 width=4)	0.244..0.246	600	600	loops=1
5	[...] Buckets: 1024 Batches: 1 Memory Usage: 30KB				
6	[...] > Index Only Scan using dependent_pkey on dependent (cost=0.28..51.15 rows=600 width=4)	0.009..0.164	600	600	loops=1
7	[...] Heap Fetches: 600				
8	Planning Time: 0.251 ms				
9	Execution Time: 3.880 ms				

A green message at the bottom right says 'Query returned successfully in 47 msec.'

Execution plan and costs:



Here the brin increased the cost to much as it is used in small range queries only

With seqscan enabled:

Planning and execution time:

The screenshot shows the pgAdmin 4 interface with the Explain tab selected. The query being explained is:

```

S 10 where exists ( select *
    > S 11 from dependent
    > S 12 where ssn=essn );
  
```

The explain output shows the following steps:

- 1 Hash Semi Join (cost=20.50..492.18 rows=600 width=42) (actual time=0.177..5.339 rows=600 loops=1)
 - 2 [...] Hash Cond: (employee.ssn = dependent.essn)
- 3 [...] Seq Scan on employee (cost=0.00..423.00 rows=16000 width=46) (actual time=0.013..1.789 rows=16000 loops=1)
 - 4 [...] > Hash (cost=13.00..13.00 rows=600 width=4) (actual time=0.156..0.158 rows=600 loops=1)
 - 5 [...] Buckets: 1024 Batches: 1 Memory Usage: 30kB
 - 6 [...] > Seq Scan on dependent (cost=0.00..13.00 rows=600 width=4) (actual time=0.010..0.101 rows=600 loops=1)
- 7 Planning Time: 0.245 ms
- 8 Execution Time: 5.385 ms

Success messages at the bottom:

- Successfully run. Total query runtime: 73 msec. 8 rows affected.
- Successfully run. Total query runtime: 66 msec. 8 rows affected.
- Query returned successfully in 47 msec.

Execution plan and costs:

The screenshot shows the pgAdmin 4 interface with the Graphical Analysis tab selected. The same query is shown:

```

S 10 where exists ( select *
    > S 11 from dependent
    > S 12 where ssn=essn );
  
```

The graphical analysis shows the following nodes:

1. → Hash Semi Join (cost=20.5..492.18 rows=600 width=42) (actual=0.192..5.055 rows=600 loops=1)
 - Hash Cond: (employee.ssn = dependent.essn)
2. → Seq Scan on employee as employee (cost=0..423 rows=16000 width=46) (actual=0.00..1.767 rows=16000 loops=1)
 - Hash (cost=13..13 rows=600 width=4) (actual=0.17..0.172 rows=600 loops=1)
 - Buckets: 1024 Batches: 1 Memory Usage: 30 kB
3. → Seq Scan on dependent as dependent (cost=0..13 rows=600 width=4) (actual=0.00..0.083 rows=600 loops=1)

A detailed table provides the following data for each node:

#	Node	Timings		Rows				Loops
		Exclusive	Inclusive	Rows X	Actual	Plan		
1.	→ Hash Semi Join (cost=20.5..492.18 rows=600 width=42) (actual=0.192..5.055 rows=600 loops=1)	3.116 ms	5.055 ms	1 1	600	600	1	
2.	→ Seq Scan on employee as employee (cost=0..423 rows=16000 width=46) (actual=0.00..1.767 rows=16000 loops=1)	1.767 ms	1.767 ms	1 1	16000	16000	1	
3.	→ Hash (cost=13..13 rows=600 width=4) (actual=0.17..0.172 rows=600 loops=1)	0.089 ms	0.172 ms	1 1	600	600	1	
4.	→ Seq Scan on dependent as dependent (cost=0..13 rows=600 width=4) (actual=0.00..0.083 rows=600 loops=1)	0.083 ms	0.083 ms	1 1	600	600	1	

Success message at the bottom:

- Query returned successfully in 47 msec.

Query5 using mixed indecies : hash on Employee(ssn) and B+ tree on Dependent(essn)

Planning and execution time :

The screenshot shows the pgAdmin 4 interface with the query editor containing the following SQL:

```
8 where exists ( select *
9   from dependent
10  where ssn=essn );
```

The "Data Output" tab is selected, showing the following execution plan details:

- 1 Merge Semi Join (cost=0.56..70.44 rows=600 width=42) (actual time=0.023..0.452 rows=600 loops=1)
 - [...] Merge Cond: (employee(ssn) = dependent(essn))
- 3 [...] Index Scan using employee_pkey on employee (cost=0.29..690.28 rows=16000 width=46) (actual time=0.009..0.115 rows=601 loops=1)
 - [...] -> Index Only Scan using dep on dependent (cost=0.28..35.27 rows=600 width=4) (actual time=0.009..0.159 rows=600 loops=1)
 - [...] Heap Fetches: 600
- Planning Time: 0.312 ms
- Execution Time: 0.489 ms

Execution plan and costs:

The screenshot shows the pgAdmin 4 interface with the query editor containing the same SQL as before. The "Explain" tab is selected, showing a graphical execution plan with three nodes:

1. → Merge Semi Join (cost=0.56..70.44 rows=600 width=42) (actual=0...)
2. → Index Scan using employee_pkey on employee as employee (...)
3. → Index Only Scan using dep on dependent as dependent (cost=...

A table below the plan provides detailed timing and row counts for each node:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Merge Semi Join (cost=0.56..70.44 rows=600 width=42) (actual=0...)	0.177 ms	0.462 ms	1 1	600	600	1
2.	→ Index Scan using employee_pkey on employee as employee (...)	0.117 ms	0.117 ms	1 26.63	601	16000	1
3.	→ Index Only Scan using dep on dependent as dependent (cost=...	0.169 ms	0.169 ms	1 1	600	600	1

At the bottom right, a message box indicates: "Successfully run. Total query runtime: 50 msec. 1 rows affected."

Query5 after optimization :

It can not be optimized too much as exists statement is optimized but we can use IN statement instead that might be better or equal to it

select fname,lname from employee where ssn in (select essn from dependent)

Query5 optimized with no index

Planning and execution time:

```

1 Hash Semi Join (cost=20.50..492.18 rows=600 width=42) (actual time=0.214..4.462 rows=600 loops=1)
  2 [...] Hash Cond: (employee.ssn = dependent.essn)
  3 [...]-> Seq Scan on employee (cost=0.00..423.00 rows=16000 width=46) (actual time=0.015..1.569 rows=16000 loops=1)
  4 [...]-> Hash (cost=13.00..13.00 rows=600 width=4) (actual time=0.170..0.172 rows=600 loops=1)
  5 [...] Buckets: 1024 Batches: 1 Memory Usage: 30kB
  6 [...]-> Seq Scan on dependent (cost=0.00..13.00 rows=600 width=4) (actual time=0.013..0.077 rows=600 loops=1)
  7 Planning Time: 1.138 ms
  8 Execution Time: 4.517 ms

```

Execution plan and costs:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Hash Semi Join (cost=20.5..492.18 rows=600 width=42) (actual=0.202... Hash Cond: (employee.ssn = dependent.essn)	3.859 ms	6.138 ms	1 1	600	600	1
2.	→ Seq Scan on employee as employee (cost=0..423 rows=16000 width=46)	2.097 ms	2.097 ms	1 1	16000	16000	1
3.	→ Hash (cost=13..13 rows=600 width=4) (actual=0.18..0.183 rows=600... Buckets: 1024 Batches: 1 Memory Usage: 30 kB	0.089 ms	0.183 ms	1 1	600	600	1
4.	→ Seq Scan on dependent as dependent (cost=0..13 rows=600 width=46)	0.094 ms	0.094 ms	1 1	600	600	1

Successfully run. Total query runtime: 55 msec. 1 rows affected.

Query5 optimized using B+ tree indecies on : Dependent(essn) ,Employee(ssn)

Planning and execution time:

```

drop index emp;
drop index dep;
explain analyze select fname, lname from employee where ssn in (select essn from dependent);

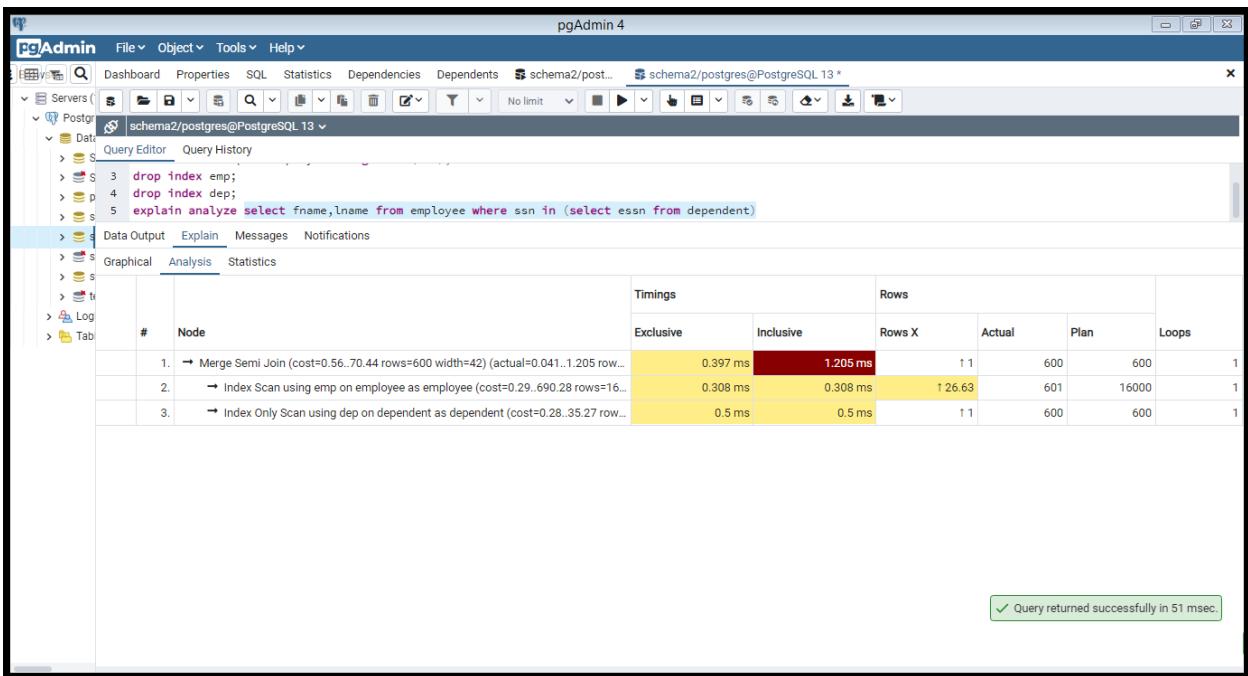
```

QUERY PLAN

1. Merge Semi Join (cost=0.56..70.44 rows=600 width=42) (actual time=0.051..1.145 loops=1)
2. [...] Merge Cond: (employee.ssn = dependent.essn)
3. [...] Index Scan using emp on employee (cost=0.29..690.28 rows=16000 width=46) (actual time=0.016..0.337 loops=1)
4. [...] Index Only Scan using dep on dependent (cost=0.28..35.27 rows=600 width=4) (actual time=0.029..0.415 loops=1)
5. [...] Heap Fetches: 600
6. Planning Time: 2.027 ms
7. Execution Time: 1.225 ms

✓ Query returned successfully in 51 msec.
✓ Query returned successfully in 47 msec.

Execution plan and costs:



Query5 optimized using hash indecies on : Dependent(essn) , Employee(ssn)

Planning and execution time:

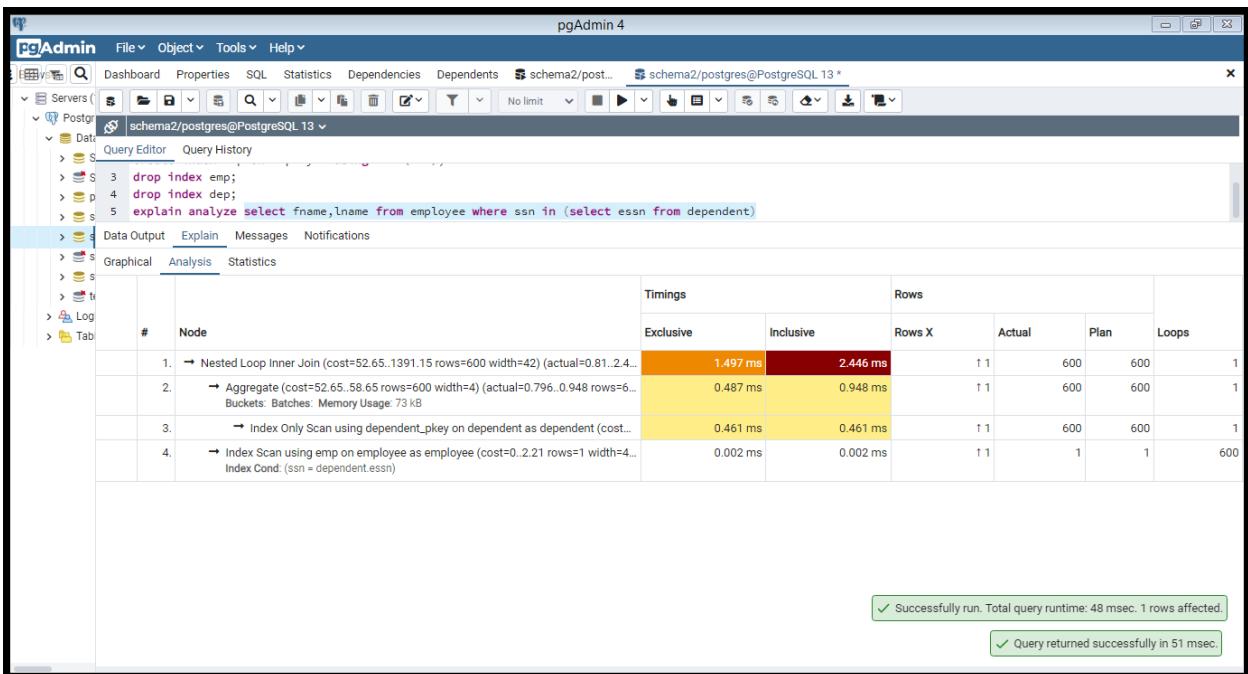
```

pgAdmin 4
File Object Tools Help
Servers (1) PostgreSQL schema2/postgres@PostgreSQL 13*
Data (1) schema2/postgres@PostgreSQL 13*
Query Editor Query History
> S 3 drop index emp;
> P 4 drop index dep;
> S 5 explain analyze select fname, lname from employee where ssn in (select essn from dependent)
> Data Output Explain Messages Notifications
> S QUERY PLAN
> S text
> tx 1 Nested Loop (cost=52.65..1391.15 rows=600 width=42) (actual time=0.301..1.533 rows=600 loops=1)
> Log 2 [...] > HashAggregate (cost=52.65..58.65 rows=600 width=4) (actual time=0.292..0.395 rows=600 loops=1)
> Log 3 [...] Group Key dependent.essn
4 [...] Batches: 1 Memory Usage: 73kB
5 [...] Index Only Scan using dependent_pkey on dependent (cost=0.28..51.15 rows=600 width=4) (actual time=0.009..0.164 rows=600 loops=1)
6 [...] Heap Fetches: 600
7 [...] Index Scan using emp on employee (cost=0.00..2.21 rows=1 width=46) (actual time=0.001..0.002 rows=1 loops=600)
8 [...] Index Cond: (ssn = dependent.essn)
9 Planning Time: 0.251 ms
10 Execution Time: 1.596 ms

```

✓ Query returned successfully in 51 msec.

Execution plan and costs:



Query5 optimized using BRIN indecies on: Dependent(essn) , Employee(ssn)

With seqscan disabled

Planning and execution time :

The screenshot shows the pgAdmin 4 interface. In the Query Editor tab, the following SQL code is run:

```
1 set enable_seqscan=off;
2 set enable_bitmaps=off;
3 create index dep on Dependent using brin(ssn);
```

The results show the query plan and execution statistics:

Step	Operation	Cost	Actual Time	Rows
1	Hash Semi Join	(cost=10000000058.65..10000000530.32 rows=600 width=42)	(actual time=0.293..4.218 rows=600 loops=1)	
2	[...] Hash Cond: (employee.ssn = dependent.essn)			
3	[...] Seq Scan on employee (cost=10000000000.00..10000000423.00 rows=16000 width=46)	(actual time=0.020..1.379 rows=16000 loops=1)		
4	[...] > Hash (cost=51.15..51.15 rows=600 width=4)	(actual time=0.263..0.264 rows=600 loops=1)		
5	[...] Buckets: 1024 Batches: 1 Memory Usage: 30kB			
6	[...] > Index Only Scan using dependent_pkey on dependent (cost=0.28..51.15 rows=600 width=4)	(actual time=0.010..0.166 rows=600 loops=1)		
7	[...] Heap Fetches: 600			
8	Planning Time: 0.391 ms			
9	Execution Time: 4.272 ms			

A green message at the bottom right indicates: "Query returned successfully in 51 msec."

Execution plan and costs:

The screenshot shows the pgAdmin 4 interface. In the Query Editor tab, the following SQL code is run:

```
5 drop index emp;
6 drop index dep;
7 explain analyze select fname, lname from employee where ssn in (select essn from dependent)
```

The results show the execution plan and its costs:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Hash Semi Join (cost=10000000058.65..10000000530.32 rows=600 width=42) (a... Hash Cond: (employee.ssn = dependent.essn))	3.68 ms	6.027 ms	1 1	600	600	1
2.	→ Seq Scan on employee as employee (cost=10000000000..10000000423 row...	2.089 ms	2.089 ms	1 1	16000	16000	1
3.	→ Hash (cost=51.15..51.15 rows=600 width=4) (actual=0.256..0.258 rows=600 ... Buckets: 1024 Batches: 1 Memory Usage: 30 kB)	0.087 ms	0.258 ms	1 1	600	600	1
4.	→ Index Only Scan using dependent_pkey on dependent as dependent (cos...	0.171 ms	0.171 ms	1 1	600	600	1

A green message at the bottom right indicates: "Query returned successfully in 51 msec."

Here the brin increased the cost to much as it is used in small range queries only

With seqscan enabled

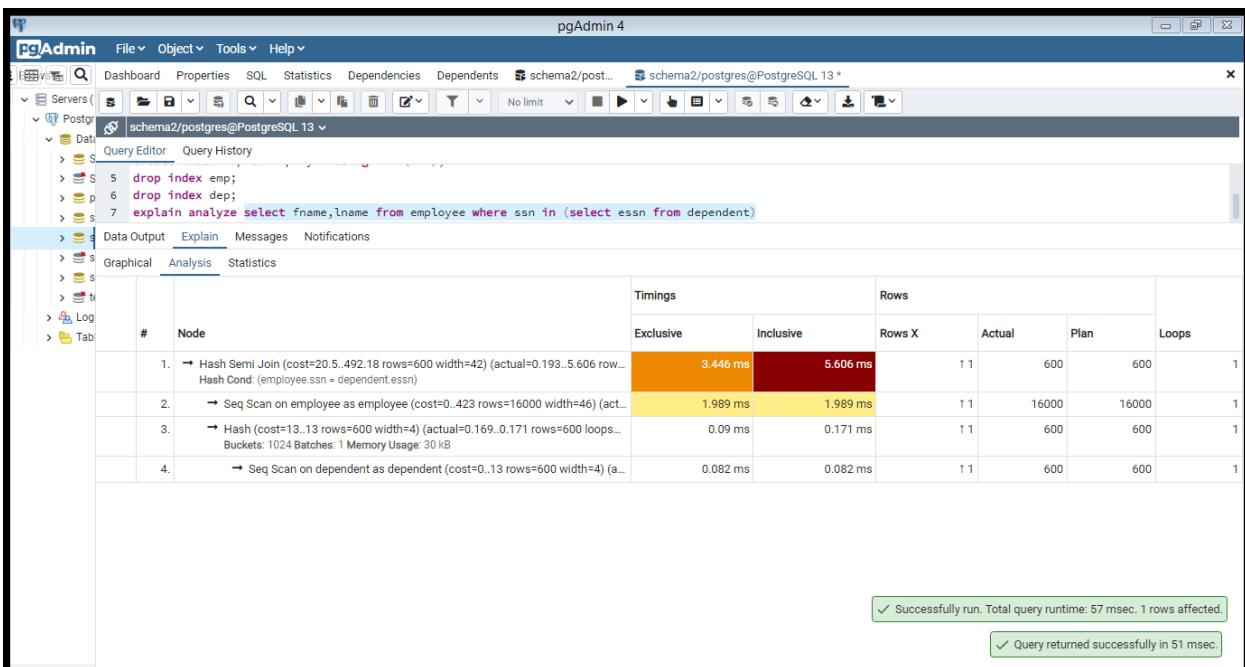
Planning and execution time:

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

- Servers:** PostgreSQL - schema2/postgres@PostgreSQL 13*
- Data:** Query Editor, Query History, Data Output, Explain, Messages, Notifications.
- Query Editor:** Contains the following SQL:

```
> S 5 drop index emp;
> S 6 drop index dep;
> S 7 explain analyze select fname, lname from employee where ssn in (select essn from dependent)
```
- Explain Plan:** Shows the execution plan with the following steps:
 - 1 Hash Semi Join (cost=20.50..492.18 rows=600 width=42) (actual time=0.203..4.310 rows=600 loops=1)
 - 2 [...] Hash Cond: (employee.ssn = dependent.essn)
 - 3 [...] > Seq Scan on employee (cost=0.00..423.00 rows=16000 width=46) (actual time=0.014..1.529 rows=16000 loops=1)
 - 4 [...] > Hash (cost=13.00..13.00 rows=600 width=4) (actual time=0.176..0.177 rows=600 loops=1)
 - 5 [...] Buckets: 1024 Batches: 1 Memory Usage: 30KB
 - 6 [...] > Seq Scan on dependent (cost=0.00..13.00 rows=600 width=4) (actual time=0.017..0.084 rows=600 loops=1)
 - 7 Planning Time: 0.356 ms
 - 8 Execution Time: 4.361 ms
- Messages:** A green message box at the bottom right indicates "Query returned successfully in 51 msec."

Execution plan and costs:



Query5 optimized using mixed indecies : hash on Employee(ssn) and B+ tree on Dependent(essn)

Planning and execution time:

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

- Servers:** PostgreSQL - schema2/postgres@PostgreSQL 13*
- Query Editor:** The query is displayed:

```
5 drop index emp;
6 drop index dep;
7 explain analyze select fname, lname from employee where ssn in (select essn from dependent);
```
- Data Output:** The results show the execution plan:

Step	Operation	Cost	Rows	Width	Time
1	Nested Loop	(cost=36.77..1375.28 rows=600 width=42)	600	42	(actual time=0.327..1.500 rows=600 loops=1)
2	[...] HashAggregate	(cost=36.77..42.77 rows=600 width=4)	600	4	(actual time=0.305..0.408 rows=600 loops=1)
3	[...] Group Key dependent.essn				
4	[...] Batches: 1 Memory Usage: 73kB				
5	[...] > Index Only Scan using dep on dependent	(cost=0.28..35.27 rows=600 width=4)	600	4	(actual time=0.015..0.160 rows=600 loops=1)
6	[...] Heap Fetches: 600				
7	[...] > Index Scan using emp on employee	(cost=0.00..2.21 rows=1 width=46)	1	46	(actual time=0.001..0.001 rows=1 loops=600)
8	[...] Index Cond: (ssn = dependent.essn)				
9	Planning Time: 0.278 ms				
10	Execution Time: 1.559 ms				
- Messages:** Two notifications are shown:
 - Successfully run. Total query runtime: 43 msec. 10 rows affected.
 - Query returned successfully in 51 msec.

Execution plan and costs:

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

- Servers:** PostgreSQL - schema2/postgres@PostgreSQL 13*
- Query Editor:** The query is displayed:

```
5 drop index emp;
6 drop index dep;
7 explain analyze select fname, lname from employee where ssn in (select essn from dependent);
```
- Data Output:** The results show the graphical execution plan and costs:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=36.77..1375.28 rows=600 width=42) (actual=0.306..1...	1.103 ms	1.501 ms	1 1	600	600	1
2.	→ Aggregate (cost=36.77..42.77 rows=600 width=4) (actual=0.295..0.396 rows=6...	0.25 ms	0.396 ms	1 1	600	600	1
3.	→ Index Only Scan using dep on dependent as dependent (cost=0.28..35.27 ...	0.147 ms	0.147 ms	1 1	600	600	1
4.	→ Index Scan using emp on employee as employee (cost=0..2.21 rows=1 width=4...	0.002 ms	0.002 ms	1 1	1	1	600
- Messages:** Two notifications are shown:
 - Successfully run. Total query runtime: 53 msec. 1 rows affected.
 - Query returned successfully in 51 msec.

Query6 with no index

Planning and execution time:

```

pgAdmin 4
File ▾ Object ▾ Tools ▾ Help ▾
Dashboard Properties SQL Statistics Dependencies Dependents schema2/postgres@PostgreSQL 13* schema2/post...
Servers (1) Data (1) Query History (1)
> S 5 salary > 40000
> p 6 and
> s 7 dno in (
>   8 select dno
>   9 from employee
>  10 group by dno
>  11 having count (*) > 5
>  12 group by dnumber;
> Log Data Output Explain Messages Notifications
> Tab QUERY PLAN
text
[...]
8 [...]--> Hash Join (cost=505.04 969.68 rows=200 width=8) (actual time=6.575..9.671 rows=600 loops=1)
9 [...] Hash Cond: (employee.dno = employee_1.dno)
10 [...]--> Seq Scan on employee (cost=0.00..469.00 rows=600 width=4) (actual time=0.194..3.111 rows=600 loops=1)
11 [...] Filter: (salary > 40000)
12 [...] Rows Removed by Filter: 15400
13 [...]--> Hash (cost=504.61..504.61 rows=34 width=4) (actual time=6.361..6.362 rows=102 loops=1)
14 [...] Buckets: 1024 Batches: 1 Memory Usage: 12kB
15 [...]--> HashAggregate (cost=503.00..504.27 rows=34 width=4) (actual time=6.302..6.323 rows=102 loops=1)
16 [...] Group Key: employee_1.dno
17 [...] Filter: (count(*) > 5)
18 [...] Buckets: 1 Memory Usage: 32kB
19 [...]--> Seq Scan on employee employee_1 (cost=0.00..423.00 rows=16000 width=4) (actual time=0.013..1.369 rows=16000 loops=1)
20 [...]--> Hash (cost=3.5..3.50 rows=150 width=4) (actual time=0.130..0.130 rows=150 loops=1)
21 [...] Buckets: 1024 Batches: 1 Memory Usage: 14kB
22 [...]--> Seq Scan on department (cost=0.0..3.50 rows=150 width=4) (actual time=0.041..0.076 rows=150 loops=1)
23 Planning Time: 0.408 ms
24 Execution Time: 28.372 ms

```

Successfully run. Total query runtime: 71 msec. 24 rows affected.

Execution plan and costs:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Aggregate (cost=980.41..982.79 rows=136 width=12) (actual=12.505..12.712 rows=100...)	0.183 ms	12.712 ms	11.36	100	136	1
2.	→ Sort (cost=980.41..980.75 rows=136 width=4) (actual=12.48..12.53 rows=600 loop...	0.177 ms	12.53 ms	14.42	600	136	1
3.	→ Hash Inner Join (cost=510.41..975.59 rows=136 width=4) (actual=8.698..12.3...	0.198 ms	12.354 ms	14.42	600	136	1
4.	→ Hash Inner Join (cost=505.04..969.68 rows=200 width=8) (actual=8.632....)	0.254 ms	12.093 ms	13	600	200	1
5.	→ Seq Scan on employee as employee (cost=0..463 rows=600 width=4) ...	3.335 ms	3.335 ms	1	600	600	1
6.	→ Hash (cost=504.61..504.61 rows=34 width=4) (actual=8.503..8.504 r...	0.02 ms	8.504 ms	13	102	34	1
7.	→ Aggregate (cost=503..504.27 rows=34 width=4) (actual=8.462....)	6.554 ms	8.484 ms	13	102	34	1
8.	→ Seq Scan on employee as employee_1 (cost=0..423 rows=...	1.93 ms	1.93 ms	1	16000	16000	1
9.	→ Hash (cost=3.5..3.5 rows=150 width=4) (actual=0.062..0.063 rows=150 l...	0.025 ms	0.063 ms	1	150	150	1
10.	→ Seq Scan on department as department (cost=0..3.5 rows=150 width...	0.038 ms	0.038 ms	1	150	150	1

Query6 using B+ tree on : Employee(salary), Employee(dno)

Disabled flags: seqscan

Planning and execution time:

```

pgAdmin 4
File Object Tools Help
Servers (1) PostgreSQL schema2/postgres@PostgreSQL 13*
No limit ▾
Data (1) Query Editor Query History
> S 14 group by dno
> p 15 having count (*) > 5
> s 16 group by dnumber;
> Data Output Explain Messages Notifications
> S 14
  ↳ QUERY PLAN
    text
    4 [...] >> Hash Join (cost=412.53..685.21 rows=136 width=4) (actual time=4.185..4.632 rows=600 loops=1)
    5 [...] Hash Cond: (employee.dno = department.dnumber)
    6 [...] >> Hash Join (cost=395.26..667.40 rows=200 width=8) (actual time=4.072..4.380 rows=600 loops=1)
    7 [...] Hash Cond: (employee.dno = employee_1.dno)
    8 [...] >> Bitmap Heap Scan on employee (cost=8.94..279.44 rows=600 width=4) (actual time=0.032..0.105 rows=600 loops=1)
    9 [...] Recheck Cond: (salary > 40000)
   10 [...] Heap Blocks: exact=11
   11 [...] >> Bitmap Index Scan on emp (cost=0.00..8.79 rows=600 width=0) (actual time=0.026..0.027 rows=600 loops=1)
   12 [...] Index Cond: (salary > 40000)
   13 [...] >> Hash (cost=385.90..385.90 rows=34 width=4) (actual time=4.031..4.031 rows=102 loops=1)
   14 [...] Buckets: 1024 Batches: 1 Memory Usage: 12 kB
   15 [...] >> GroupAggregate (cost=0.29..385.56 rows=34 width=4) (actual time=0.120..4.012 rows=102 loops=1)
   16 [...] Group Key: employee_1.dno
   17 [...] Filter: (count(*) > 5)
   18 [...] >> Index Only Scan using emp2 on employee employee_1 (cost=0.29..304.29 rows=16000 width=4) (actual time=0.010..2.194 rows=150)
   19 [...] Heap Fetches: 0
   20 [...] >> Hash (cost=15.39..15.39 rows=150 width=4) (actual time=0.105..0.105 rows=150 loops=1)
   21 [...] Buckets: 1024 Batches: 1 Memory Usage: 14 kB
   22 [...] >> Index Only Scan using department_pkey on department (cost=0.14..15.39 rows=150 width=4) (actual time=0.023..0.072 rows=150)
   23 [...] Heap Fetches: 150
   24 Planning Time: 0.668 ms
   25 Execution Time: 4.921 ms
  
```

Execution plan and costs:

#	Node	Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Aggregate (cost=685.89..687.25 rows=136 width=12) (actual=4.744..4.777 rows=100 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 48 kB	0.286 ms	4.777 ms	↑ 1.36	100	136	1
2.	→ Hash Inner Join (cost=412.53..685.21 rows=136 width=4) (actual=3.751..4.492 rows=600 loops=1) Hash Cond: (employee.dno = department.dnumber)	0.23 ms	4.492 ms	↓ 4.42	600	136	1
3.	→ Hash Inner Join (cost=395.26..667.40 rows=200 width=8) (actual=3.672..4.191 rows=600 loops=1) Hash Cond: (employee.dno = employee_1.dno)	0.37 ms	4.191 ms	↓ 3	600	200	1
4.	→ Bitmap Heap Scan on employee as employee (cost=8.94..279.44 rows=600 width=4) Recheck Cond: (salary > 40000) Heap Blocks: exact=11	0.165 ms	0.193 ms	↑ 1	600	600	1
5.	→ Bitmap Index Scan using emp (cost=0..8.79 rows=600 width=0) (actual time=0.028 ms) Index Cond: (salary > 40000)	0.028 ms	0.028 ms	↑ 1	600	600	1
6.	→ Hash (cost=385.9..385.9 rows=34 width=4) (actual=3.627..3.628 rows=102 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 12 kB	0.039 ms	3.628 ms	↓ 3	102	34	1
7.	→ Aggregate (cost=0.29..385.56 rows=34 width=4) (actual=0.116..3.59 rows=102 loops=1) Filter: (count(*) > 5) Rows Removed by Filter: 0	1.667 ms	3.59 ms	↓ 3	102	34	1
8.	→ Index Only Scan using emp2 on employee as employee_1 (cost=0.29..304.29 rows=16000 width=4) Buckets: 1024 Batches: 1 Memory Usage: 14 kB	1.923 ms	1.923 ms	↑ 1	16000	16000	1
9.	→ Hash (cost=15.39..15.39 rows=150 width=4) (actual=0.072..0.072 rows=150 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 14 kB	0.027 ms	0.072 ms	↑ 1	150	150	1
10.	→ Index Only Scan using department_pkey on department as department (cost=0.14..15.39 rows=150 width=4)	0.045 ms	0.045 ms	↑ 1	150	150	1

Here the B+ tree increased performance as it is used in range queries and it search in O(logn) time

Query 6 using hash indecies on : Employee(salary), Employee(dno) ,Department(Dnumber)

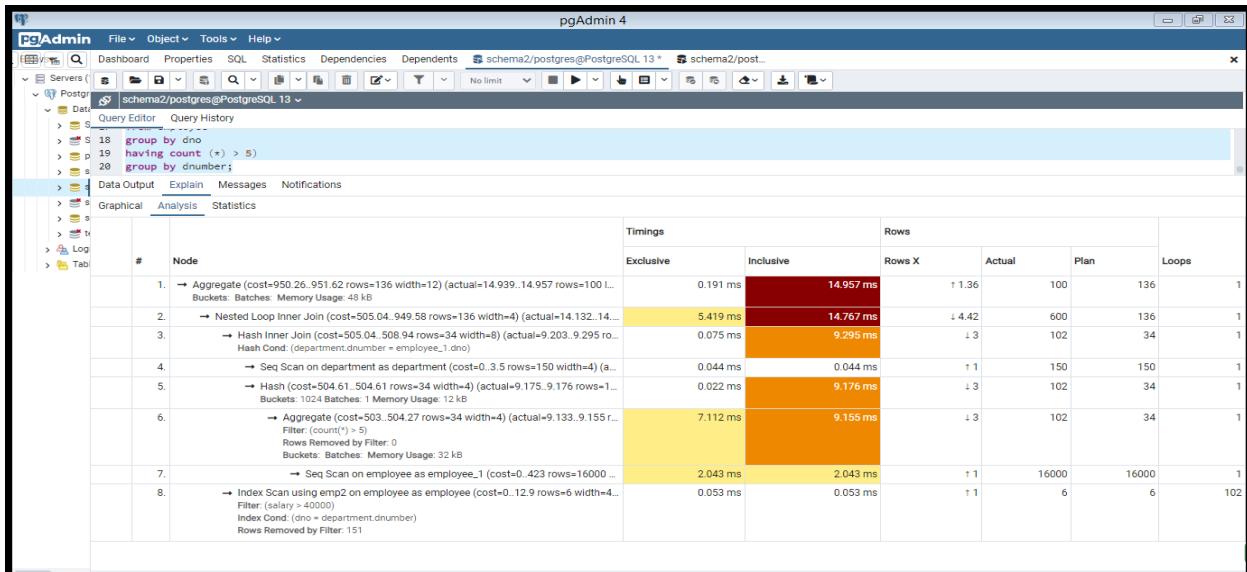
Disabled flags: bitmapscan and index scan

Planning and execution time:

```

pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents schema2/postgres@PostgreSQL 13* schema2/post...
Servers PostgreSQL schema2/postgres@PostgreSQL 13*
Data Query Editor Query History
S 18 group by dno
p 19 having count (*) > 5
s 20 group by dnumber;
Data Output Explain Messages Notifications
S 1 QUERY PLAN
text
1 HashAggregate (cost=950.26..951.62 rows=136 width=12) (actual time=15.076..15.095 rows=100 loops=1)
   Group Key: department.dnumber
   Batches: 1 Memory Usage: 48kB
   1 [...] Group Key: department.dnumber
   2 [...] Batches: 1 Memory Usage: 48kB
   3 [...] Batches: 1 Memory Usage: 48kB
   4 [...] > Nested Loop (cost=505.04..949.58 rows=136 width=4) (actual time=14.357..14.920 rows=500 loops=1)
      5 [...] > Hash Join (cost=505.04..508.94 rows=34 width=8) (actual time=7.697..7.781 rows=102 loops=1)
         6 [...] Hash Cond: (department.dnumber = employee_1.dno)
         7 [...] > Seq Scan on department (cost=0.00..3.50 rows=150 width=4) (actual time=0.031..0.051 rows=150 loops=1)
         8 [...] > Hash (cost=504.61..504.61 rows=34 width=4) (actual time=7.636..7.636 rows=102 loops=1)
         9 [...] Buckets: 1024 Batches: 1 Memory Usage: 12kB
        10 [...] > HashAggregate (cost=503.00..504.27 rows=34 width=4) (actual time=7.538..7.580 rows=102 loops=1)
           11 [...] Group Key: employee_1.dno
           12 [...] Filter: (count(*) > 5)
           13 [...] Batches: 1 Memory Usage: 32kB
           14 [...] > Seq Scan on employee employee_1 (cost=0.00..423.00 rows=16000 width=4) (actual time=0.021..1.701 rows=16000 loops=1)
           15 [...] > Index Scan using emp2 on employee (cost=0.00..12.90 rows=6 width=4) (actual time=0.067..0.069 rows=6 loops=102)
           16 [...] Index Cond: (dno = department.dnumber)
           17 [...] Filter: (salary > 40000)
           18 [...] Rows Removed by Filter: 151
           19 Planning Time: 0.514 ms
           20 Execution Time: 15.191 ms
  
```

Execution plan and costs:



Here the hash did not help that much as it is used in exact value queries it does not help in range queries as it search in O(1) time

Query6 using BRIN indecies on : Employee(salary), Employee(dno) ,Department(Dnumber)

Disabled flags: seqscan

Planning and execution time:

```

pgAdmin 4
File ▾ Object ▾ Tools ▾ Help ▾
Dashboard Properties SQL Statistics Dependencies Dependents schema2/postgres@PostgreSQL 13* schema2/post...
Servers (1) Data (1) Query Editor Query History
> S 4 create index emp on Employee using brin(salary);
> S 5 create index emp2 on Employee using brin(dno);
> p 6 create index dept on Department using brin(Dnumber);
> S 7
> S 8 Data Output Explain Messages Notifications
> S 9 QUERY PLAN
4 [...] -> Hash Join (cost=522.61..871.15 rows=136 width=4) (actual time=7.108..9.807 rows=600 loops=1)
5 [...] Hash Cond: (employee.dno = department.dnumber)
6 [...] -> Hash Join (cost=517.23..865.23 rows=200 width=8) (actual time=7.041..9.501 rows=600 loops=1)
7 [...] Hash Cond: (employee.dno = employee_1.dno)
8 [...] -> Bitmap Heap Scan on employee (cost=12.19..358.55 rows=600 width=4) (actual time=0.160..2.285 rows=600 loops=1)
9 [...] Recheck Cond: (salary > 40000)
10 [...] Rows Removed by Index Recheck: 7208
11 [...] Heap Blocks: lossy=128
12 [...] -> Bitmap Index Scan on emp (cost=0.00..12.04 rows=6669 width=0) (actual time=0.025..0.026 rows=1280 loops=1)
13 [...] Index Cond: (salary > 40000)
14 [...] -> Hash (cost=504.61..504.61 rows=34 width=4) (actual time=6.872..6.874 rows=102 loops=1)
15 [...] Buckets: 1024 Batches: 1 Memory Usage: 12kB
16 [...] -> HashAggregate (cost=503.00..504.27 rows=34 width=4) (actual time=6.787..6.823 rows=102 loops=1)
17 [...] Group Key: employee_1.dno
18 [...] Filter: (count(*) > 5)
19 [...] Buckets: 1 Memory Usage: 32kB
20 [...] -> Seq Scan on employee employee_1 (cost=0.00..423.00 rows=16000 width=4) (actual time=0.009..1.460 rows=16000 loops=1)
21 [...] -> Hash (cost=3..30..3.50 rows=150 width=4) (actual time=0.062..0.062 rows=150 loops=1)
22 [...] Buckets: 1024 Batches: 1 Memory Usage: 14kB
23 [...] -> Seq Scan on department (cost=0..0.35 rows=150 width=4) (actual time=0.016..0.036 rows=150 loops=1)
24 Planning Time: 0.538 ms
25 Execution Time: 10.215 ms
  
```

Execution plan and costs:

#	Node	Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Aggregate (cost=871.83..873.19 rows=136 width=12) (actual=12.66..12.751 rows=100 loops=1)	0.202 ms	12.751 ms	1 1.36	100	136	1
2.	→ Hash Inner Join (cost=522.61..871.15 rows=136 width=4) (actual=9.11..12.549 rows=600 loops=1)	0.223 ms	12.549 ms	1 4.42	600	136	1
3.	→ Hash Inner Join (cost=517.23..865.23 rows=200 width=8) (actual=9.042..12.2.. rows=600 loops=1)	0.218 ms	12.265 ms	1 3	600	200	1
4.	→ Bitmap Heap Scan on employee as employee (cost=12.19..358.55 rows=600 width=4) (actual=6.872..6.874 rows=102 loops=1)	3.151 ms	3.176 ms	1 1	600	600	1
5.	→ Bitmap Index Scan using emp (cost=0..12.04 rows=6669 width=0) (actual=0.025..0.026 rows=1280 loops=1)	0.025 ms	0.025 ms	1 5.22	1280	6669	1
6.	→ Hash (cost=504.61..504.61 rows=34 width=4) (actual=8.87..8.872 rows=102 loops=1)	0.026 ms	8.872 ms	1 3	102	34	1
7.	→ Aggregate (cost=503..504.27 rows=34 width=4) (actual=8.827..8.84.. rows=102 loops=1)	6.782 ms	8.847 ms	1 3	102	34	1
8.	→ Seq Scan on employee as employee_1 (cost=0..423 rows=16000 loops=1)	2.065 ms	2.065 ms	1 1	16000	16000	1
9.	→ Hash (cost=3..30..3.5 rows=150 width=4) (actual=0.059..0.061 rows=150 loops=1)	0.027 ms	0.061 ms	1 1	150	150	1
10.	→ Seq Scan on department as department (cost=0..0.35 rows=150 width=4) (actual=0.034 ms)	0.034 ms	0.034 ms				

Successfully run. Total query runtime: 68 msec. 1 rows affected.

Here the BRIN increased performance as it is used in small range queries

Query6 using mixed indecies : B+ tree Employee(salary) , Employee(dno) and hash on Department(Dnumber)

Planning and execution time:

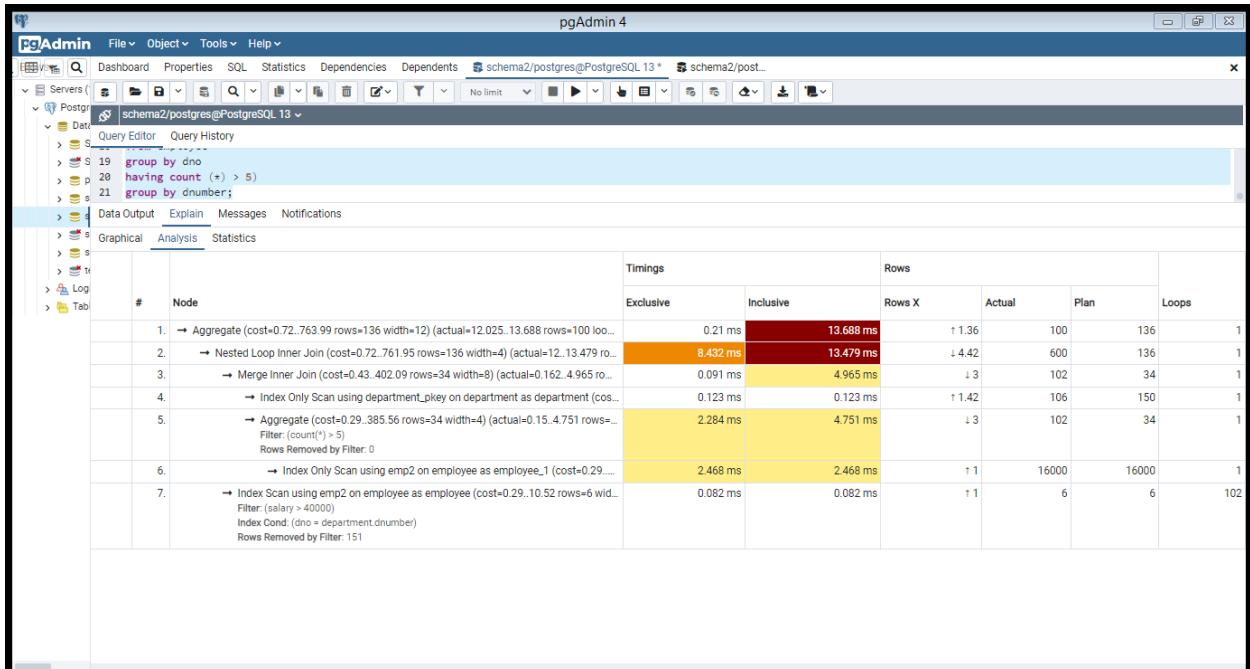
```

pgAdmin 4
File Object Tools Help
Servers (1) PostgreSQL schema2/postgres@PostgreSQL 13*
Query Editor Query History
S 19 group by dno
S 20 having count (*) > 5
P 21 group by dnumber;
Data Output Explain Messages Notifications
QUERY PLAN
text
1 GroupAggregate (cost=0.72..763.99 rows=136 width=12) (actual time=8.369..9.937 rows=100 loops=1)
   2 [...] Group Key: department.dnumber
   3 [...] Nested Loop (cost=0.72..761.95 rows=136 width=4) (actual time=8.345..9.723 rows=600 loops=1)
      4 [...] >> Merge Join (cost=0.43..402.09 rows=34 width=8) (actual time=0.127..3.522 rows=102 loops=1)
         5 [...] Merge Cond: (department.dnumber = employee_1.dno)
         6 [...] >> Index Only Scan using department_pkey on department (cost=0.14..15.39 rows=150 width=4) (actual time=0.008..0.114 rows=106 loops=1)
         7 [...] >> Heap Fetches: 106
         8 [...] >> GroupAggregate (cost=0.29..385.56 rows=34 width=4) (actual time=0.115..3.325 rows=102 loops=1)
         9 [...] Group Key: employee_1.dno
        10 [...] Filter: (count(*) > 5)
        11 [...] >> Index Only Scan using emp2 on employee employee_1 (cost=0.29..304.29 rows=16000 width=4) (actual time=0.006..1.748 rows=16000 loops=1)
        12 [...] >> Heap Fetches: 0
        13 [...] >> Index Scan using emp2 on employee (cost=0.29..10.52 rows=6 width=4) (actual time=0.056..0.059 rows=6 loops=102)
        14 [...] Index Cond: (dno = department.dnumber)
        15 [...] Filter: (salary > 40000)
        16 [...] Rows Removed by Filter: 0
        17 Planning Time: 0.575 ms
        18 Execution Time: 10.014 ms

```

Successfully run. Total query runtime: 51 msec. 18 rows affected.

Execution plan and costs:



Query6 after optimization :

```
select dnumber, count(*)  
from department, employee
```

where dnumber=dno

and

salary > 40000

group by dnumber

having count(*)>5

we removed the in statement as it did not affect the query at all it only make it worse

Query6 optimized with no index:

Planning and execution time:

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1) > PostgreSQL 13 > Databases (8)', the 'schema2' database is selected. In the main area, the 'Query Editor' tab is active, displaying the following SQL code:

```
1 explain analyze select dnumber, count(*)
2 from department, employee
3 where dnumber=dno
```

Below the query editor, the 'Data Output' tab is selected, showing the 'QUERY PLAN' section with the following details:

Step	Operation	Cost	Time
1	HashAggregate	(cost=474.49..476.37 rows=50 width=12)	(actual time=4.758..4.796 rows=100 loops=1)
2	[...] Group Key: department.dnumber		
3	[...] Filter: (count(*) > 5)		
4	[...] Batches: 1 Memory Usage: 48kB		
5	[...] Hash Join (cost=5.38..469.99 rows=600 width=4)	(actual time=0.173..4.573 rows=600 loops=1)	
6	[...] Hash Cond: (employee.dno = department.dnumber)		
7	[...] Seq Scan on employee (cost=0.00..463.00 rows=600 width=4)	(actual time=0.118..4.314 rows=600 loops=1)	
8	[...] Filter: (salary > 40000)		
9	[...] Rows Removed by Filter: 15400		
10	[...] Hash (cost=3.50..3.50 rows=150 width=4)	(actual time=0.048..0.050 rows=150 loops=1)	
11	[...] Buckets: 1024 Batches: 1 Memory Usage: 14kB		
12	[...] Seq Scan on department (cost=0.00..3.50 rows=150 width=4)	(actual time=0.010..0.027 rows=150 loops=1)	
13	Planning Time: 0.209 ms		
14	Execution Time: 4.849 ms		

Execution plan and costs:

The screenshot shows the pgAdmin 4 interface with the same setup as the previous one. The 'schema2' database is selected in the left sidebar. In the main area, the 'Query Editor' tab is active, displaying the following SQL code:

```
6 group by dnumber
7 having count(*)>5
8
```

Below the query editor, the 'Explain' tab is selected, showing the 'Graphical' view of the execution plan. The plan consists of five nodes:

- Node 1: Aggregate (cost=474.49..476.37 rows=50 width=12). This node has a red background, indicating it is the most expensive part of the query.
- Node 2: Hash Inner Join (cost=5.38..469.99 rows=600 width=4).
- Node 3: Seq Scan on employee as employee (cost=0..463 rows=600 width=4).
- Node 4: Hash (cost=3.5..3.5 rows=150 width=4).
- Node 5: Seq Scan on department as department (cost=0..3.5 rows=150 width=4).

Below the graphical plan, the 'Analysis' tab is selected, showing a detailed table of execution times and resource usage:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	Aggregate (cost=474.49..476.37 rows=50 width=12)	0.289 ms	5.245 ms	1 2	100	50	1
2.	Hash Inner Join (cost=5.38..469.99 rows=600 width=4)	0.267 ms	4.957 ms	1 1	600	600	1
3.	Seq Scan on employee as employee (cost=0..463 rows=600 width=4)	4.609 ms	4.609 ms	1 1	600	600	1
4.	Hash (cost=3.5..3.5 rows=150 width=4)	0.042 ms	0.081 ms	1 1	150	150	1
5.	Seq Scan on department as department (cost=0..3.5 rows=150 width=4)	0.039 ms	0.039 ms	1 1	150	150	1

Query6 optimized using B+ tree on : Employee(salary), Department(Dnumber)

Planning and execution time:

```

8 group by dnumber
9 having count(*) > 5
10
11
12
13
14
15 Planning Time: 1.366 ms
16 Execution Time: 0.658 ms
  
```

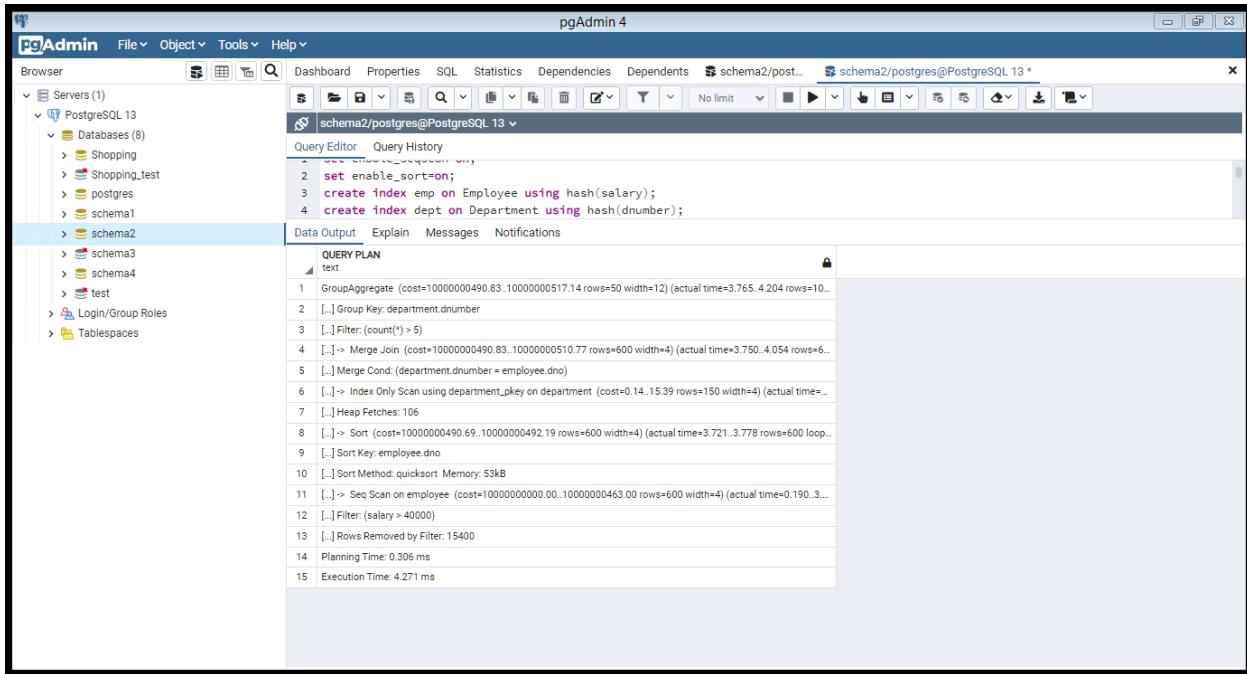
Execution plan and costs:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Aggregate (cost=290.93..292.8 rows=50 width=12) (actual=0.677..0.713 ms) Filter: (count(*) > 5) Rows Removed by Filter: 0 Buckets: Batches: Memory Usage: 48 kB	0.22 ms	0.713 ms	1 2	100	50	1
2.	→ Hash Inner Join (cost=14.31..286.43 rows=600 width=4) (actual=0.274..0.493 ms) Hash Cond: (employee.dno = department.dnumber)	0.274 ms	0.493 ms	1 1	600	600	1
3.	→ Bitmap Heap Scan on employee as employee (cost=8.94..279.44 rows=600 width=4) (actual=0.13..0.16 ms) Recheck Cond: (salary > 40000) Heap Blocks: exact=11	0.13 ms	0.16 ms	1 1	600	600	1
4.	→ Bitmap Index Scan using emp (cost=0..8.79 rows=600 width=4) (actual=0.03..0.03 ms) Index Cond: (salary > 40000)	0.03 ms	0.03 ms	1 1	600	600	1
5.	→ Hash (cost=3.5..3.5 rows=150 width=4) (actual=0.027..0.059 ms) Buckets: 1024 Batches: 1 Memory Usage: 14 kB	0.027 ms	0.059 ms	1 1	150	150	1
6.	→ Seq Scan on department as department (cost=0..3.5 rows=150 width=4) (actual=0.032..0.032 ms)	0.032 ms	0.032 ms	1 1	150	150	1

Query6 optimized using hash indecies on : Employee(salary), Department(dnumber)

With seqscan disabled

Planning and execution time:



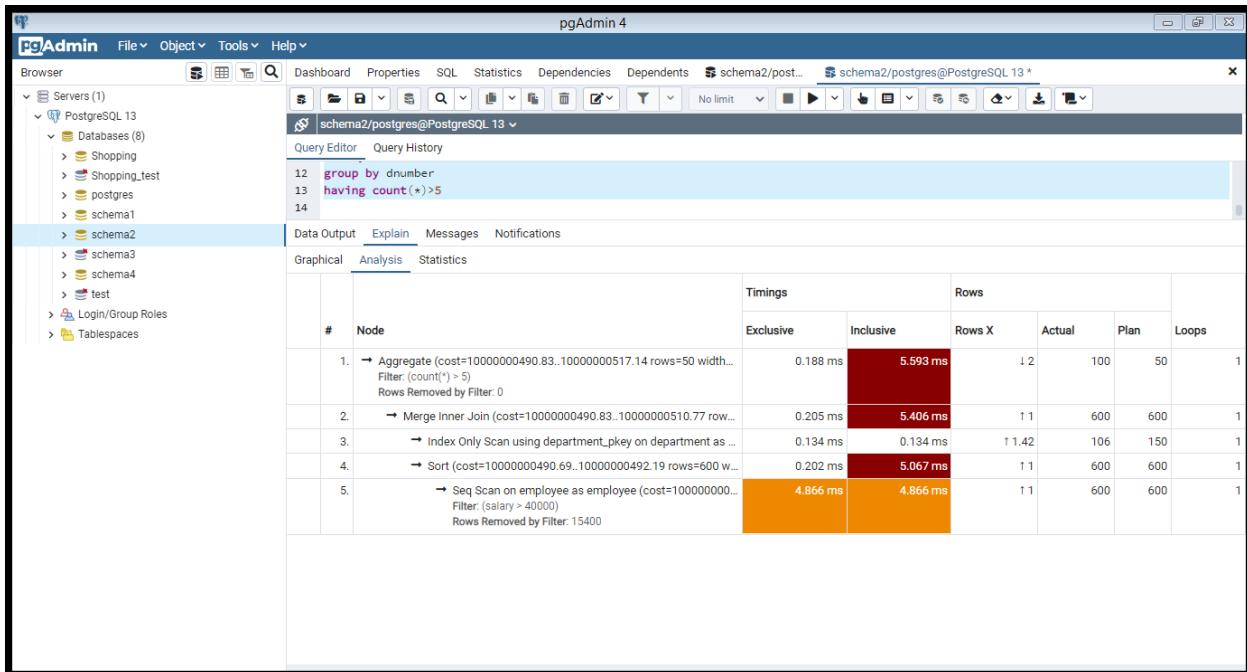
The screenshot shows the pgAdmin 4 interface with the 'Explain' tab selected. The query editor contains the following SQL code:

```
1 set enable_seqscan;
2 set enable_sorton;
3 create index emp on Employee using hash(salary);
4 create index dept on Department using hash(dnumber);
```

The 'Explain' tab displays the query plan in text format:

```
QUERY PLAN
text
1 GroupAggregate (cost=10000000490.83..10000000517.14 rows=50 width=12) (actual time=3.765..4.204 rows=10...)
2 (...) Group Key: department.dnumber
3 (...) Filter: (count(*) > 5)
4 (...) -> Merge Join (cost=10000000490.83..10000000510.77 rows=600 width=4) (actual time=3.750..4.054 rows=6...)
5 (...) Merge Cond: (department.dnumber = employee.dno)
6 (...) -> Index Only Scan using department_pkey on department (cost=0.14..15.39 rows=150 width=4) (actual time=...
7 (...) Heap Fetches: 106
8 (...) -> Sort (cost=10000000490.69..10000000492.19 rows=600 width=4) (actual time=3.721..3.778 rows=600 loop...
9 (...) Sort Key: employee.dno
10 (...) Sort Method: quicksort Memory: 53kB
11 (...) -> Seq Scan on employee (cost=10000000000.00..10000000463.00 rows=600 width=4) (actual time=0.190..3...
12 (...) Filter: (salary > 40000)
13 (...) Rows Removed by Filter: 15400
14 Planning Time: 0.306 ms
15 Execution Time: 4.271 ms
```

Execution plan and costs:



Here the hash did not help and it increased the cost as it is used in exact value queries only in O(1) time so it did not help in the group by and the range query

With seqscan enabled

Planning and execution time:

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1)', there is one entry for 'PostgreSQL 13' which contains several databases: Shopping, Shopping_test, postgres, schema1, schema2, schema3, schema4, test, Login/Group Roles, and Tablespaces. The 'schema2' database is selected. In the main window, the 'Query Editor' tab is active, displaying the following SQL code:

```

1 set enable_seqscan=on;
2 set enable_sort=on;
3 create index emp on Employee using hash(salary);

```

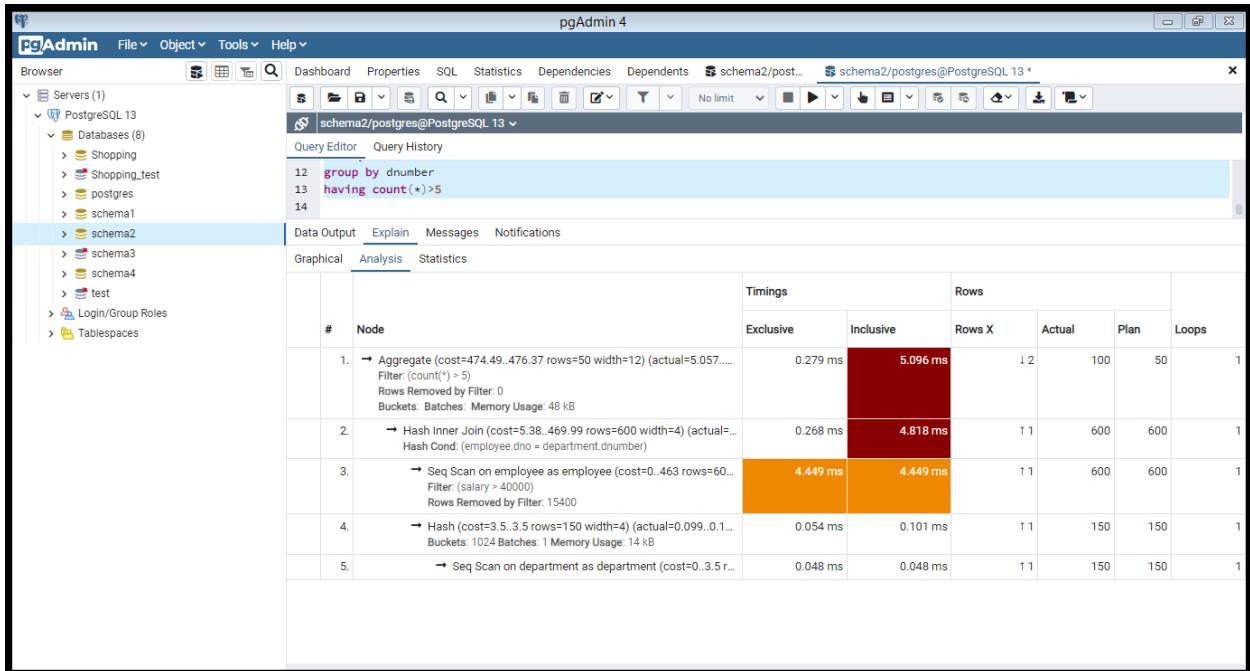
Below the code, the 'Explain' tab is selected, showing the execution plan in text format:

```

QUERY PLAN
text
1 HashAggregate (cost=474.49..476.37 rows=50 width=12) (actual time=3.826..3.852 rows=100 loops=1)
  2 (...) Group Key: department.dnumber
  3 (...) Filter: (count(*) > 5)
  4 (...) Batches: 1 Memory Usage: 48kB
  5 (...) > Hash Join (cost=5.38..469.99 rows=600 width=4) (actual time=0.303..3.589 rows=600 loops=1)
  6 (...) Hash Cond: (employee.dno = department.dnumber)
  7 (...) >> Seq Scan on employee (cost=0.00..463.00 rows=600 width=4) (actual time=0.200..3.232 rows=600 loops=1)
  8 (...) Filter: (salary > 40000)
  9 (...) Rows Removed by Filter: 15400
  10 (...) >> Hash (cost=3.50..3.50 rows=150 width=4) (actual time=0.091..0.092 rows=150 loops=1)
  11 (...) Buckets: 1024 Batches: 1 Memory Usage: 14kB
  12 (...) >> Seq Scan on department (cost=0.00..3.50 rows=150 width=4) (actual time=0.019..0.044 rows=150 loops=1)
  13 Planning Time: 0.351 ms
  14 Execution Time: 3.923 ms

```

Execution plan and costs:



Query6 optimized using BRIN indecies on : Employee(salary), Department(dnumber)

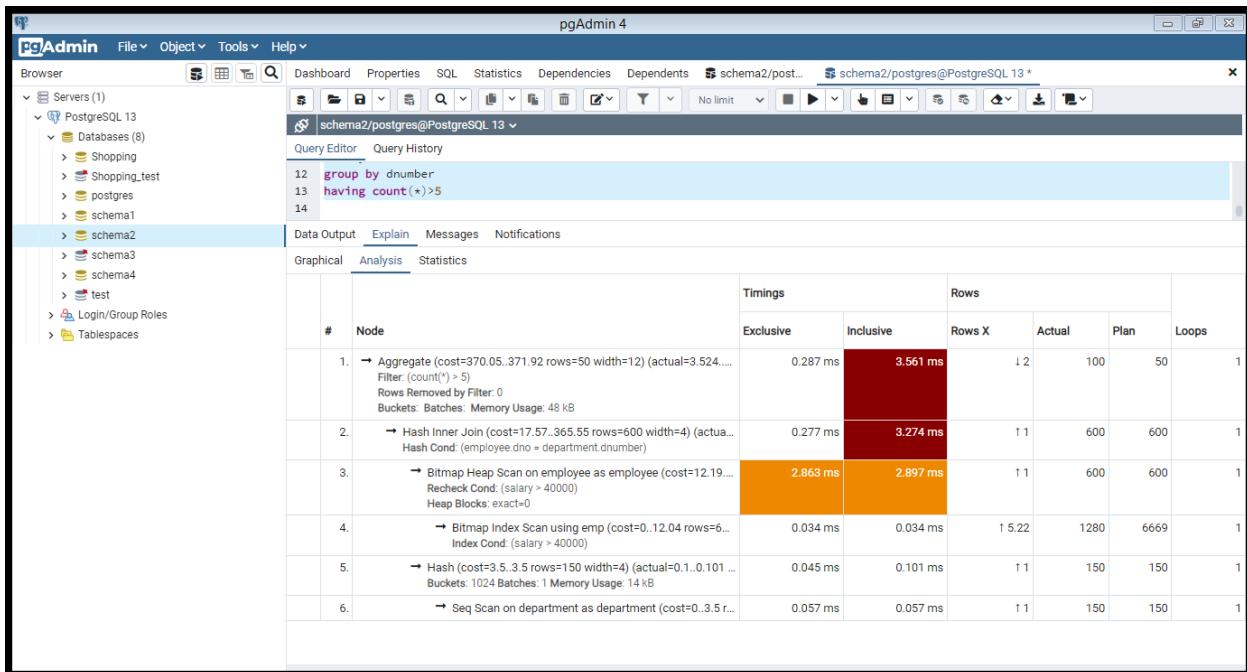
Planning and execution time:

```

pgAdmin 4
File Object Tools Help
Browser
  Servers (1)
    PostgreSQL 13
      Databases (8)
        Shopping
        Shopping_test
        postgres
        schema1
        schema2
        schema3
        schema4
        test
      Login/Group Roles
      Tablespaces
Query Editor Query History
1 create index emp on Employee using brin(salary);
2 create index dept on Department using brin(dnumber);
3 drop index emp;
Data Output Explain Messages Notifications
QUERY PLAN
text
1 HashAggregate (cost=370.05..371.92 rows=50 width=12) (actual time=2.445..2.482 rows=100 loops=1)
2 [...] Group Key: department.dnumber
3 [...] Filter: (count(*) > 5)
4 [...] Batches: 1 Memory Usage: 48kB
5 [...] > Hash Join (cost=17.57..365.55 rows=600 width=4) (actual time=0.230..2.315 rows=600 loops=1)
6 [...] Hash Cond: (employee.dno = department.dnumber)
7 [...] > Bitmap Heap Scan on employee (cost=12.19..358.55 rows=600 width=4) (actual time=0.163..2.097 rows=600 loops=1)
8 [...] Recheck Cond: (salary > 40000)
9 [...] Rows Removed by Index Recheck: 7208
10 [...] Heap Blocks: lossy=128
11 [...] > Bitmap Index Scan on emp (cost=0.00..12.04 rows=6669 width=0) (actual time=0.023..0.024 rows=1280 loops=1)
12 [...] Index Cond: (salary > 40000)
13 [...] > Hash (cost=3.50..3.50 rows=150 width=4) (actual time=0.060..0.061 rows=150 loops=1)
14 [...] Buckets: 1024 Batches: 1 Memory Usage: 14kB
15 [...] > Seq Scan on department (cost=0.00..3.50 rows=150 width=4) (actual time=0.014..0.033 rows=150 loops=1)
16 Planning Time: 1.188 ms
17 Execution Time: 2.567 ms

```

Execution plan and costs:



Query6 with mixed indecies on : B+ tree on Employee(salary) , Department(dnumber)

Planning and execution time:

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1)', there is one entry for 'PostgreSQL 13' which contains 'Databases (8)' including 'Shopping', 'Shopping_test', 'postgres', 'schema1', 'schema2', 'schema3', 'schema4', and 'test'. The 'schema2' database is selected. In the main area, the 'Query Editor' tab is active with the query: 'group by dnumber having count(*)>5'. Below the query is the 'QUERY PLAN' section, which details the execution steps: HashAggregate, Group Key: department.dnumber, Filter: (count(*) > 5), Batches: 1 Memory Usage: 48kB, Hash Join, Hash Cond: (employee.dno = department.dnumber), Bitmap Heap Scan on employee, Recheck Cond: (salary > 40000), Heap Blocks: exact=11, Bitmap Index Scan on emp, Hash, Hash Cond: (salary > 40000), Batches: 1 Memory Usage: 14kB, Seq Scan on department, Planning Time: 1.744 ms, and Execution Time: 0.692 ms. A green message bar at the bottom right indicates 'Successfully run. Total query runtime: 49 msec. 16 rows affected.'

Execution plan and costs:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Aggregate (cost=290.93..292.8 rows=50 width=12) (actual=0.5..0.52... Filter: (count(*) > 5) Rows Removed by Filter: 0 Buckets: Batches: Memory Usage: 48 kB	0.154 ms	0.521 ms	1 2	100	50	1
2.	→ Hash Inner Join (cost=14.31..286.43 rows=600 width=4) (actual=0.198..0.368 ms Hash Cond: (employee.dno = department.dnumber)	0.198 ms	0.368 ms	1 1	600	600	1
3.	→ Bitmap Heap Scan on employee as employee (cost=8.94..0.083 ms Recheck Cond: (salary > 40000) Heap Blocks: Memory Usage: 14 kB	0.083 ms	0.11 ms	1 1	600	600	1
4.	→ Bitmap Index Scan using emp (cost=0..8.79 rows=600 width=4) Index Cond: (salary > 40000)	0.027 ms	0.027 ms	1 1	600	600	1
5.	→ Hash (cost=3..5.5 rows=150 width=4) (actual=0.025..0.06 ms Buckets: 1024 Batches: 1 Memory Usage: 14 kB	0.025 ms	0.06 ms	1 1	150	150	1
6.	→ Seq Scan on department as department (cost=0..3.5 rows=150 width=4)	0.035 ms	0.035 ms	1 1	150	150	1

Query 7 with no indices(Flags[“seqscan = on”, “hashjoin=on”, “mergejoin = on”, all primary key indices are indisvalid = true])

Planning and execution time

```

1 explain analyze select s.sname
2   from sailors s
3   where
4     s.sid in( select r.sid
5       from reserves r
6      where r.bid = 103 );

```

QUERY PLAN

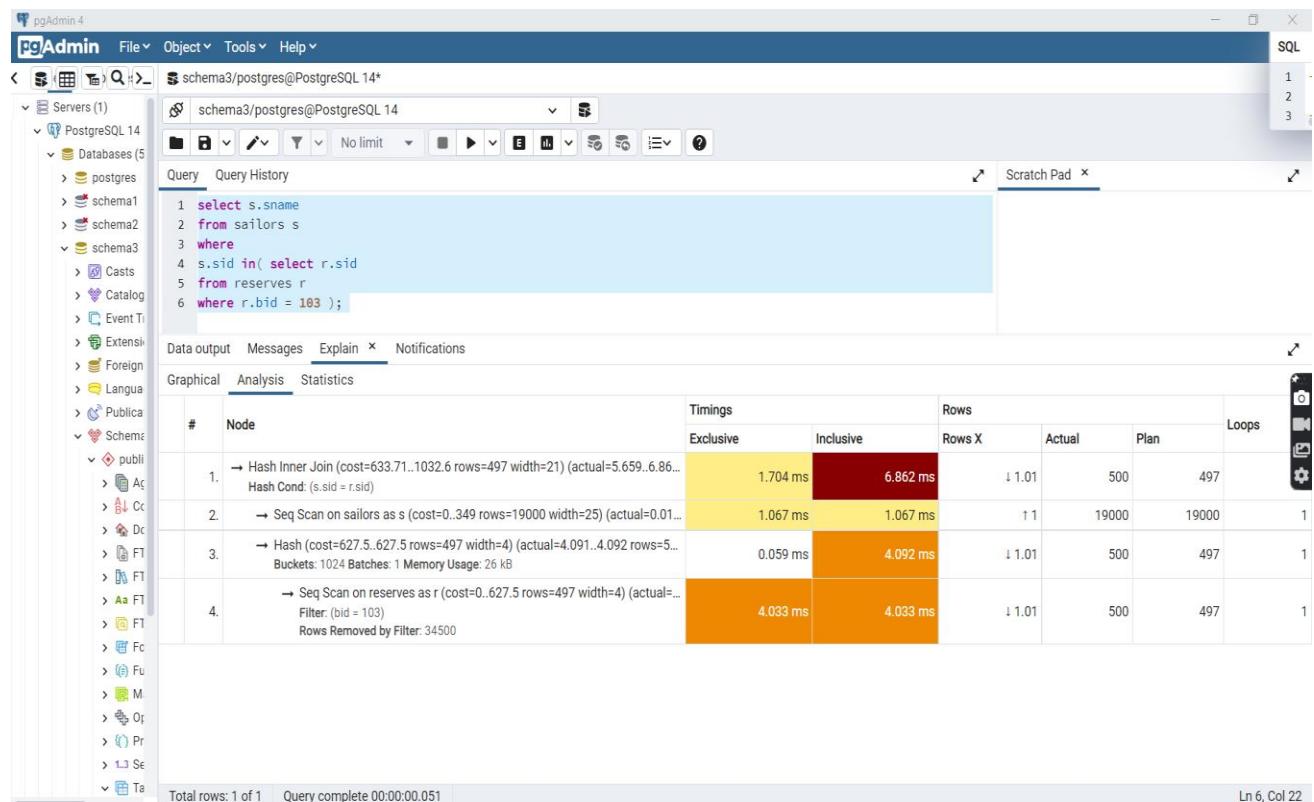
```

1 Hash Join (cost=633.71..1032.60 rows=497 width=21) (actual time=11.459..14.740 rows=500 loops=1)
2 Hash Cond: (s.sid = r.sid)
3   > Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.037..3.260 rows=19000 loops=1)
4     > Hash (cost=627.50..627.50 rows=497 width=4) (actual time=7.084..7.085 rows=500 loops=1)
5       Buckets: 1024 Batches: 1 Memory Usage: 26kB
6     > Seq Scan on reserves r (cost=0.00..627.50 rows=497 width=4) (actual time=6.471..6.918 rows=500 loops=1)
7       Filter: (bid = 103)
8       Rows Removed by Filter: 34500
9 Planning Time: 0.597 ms
10 Execution Time: 14.824 ms

```

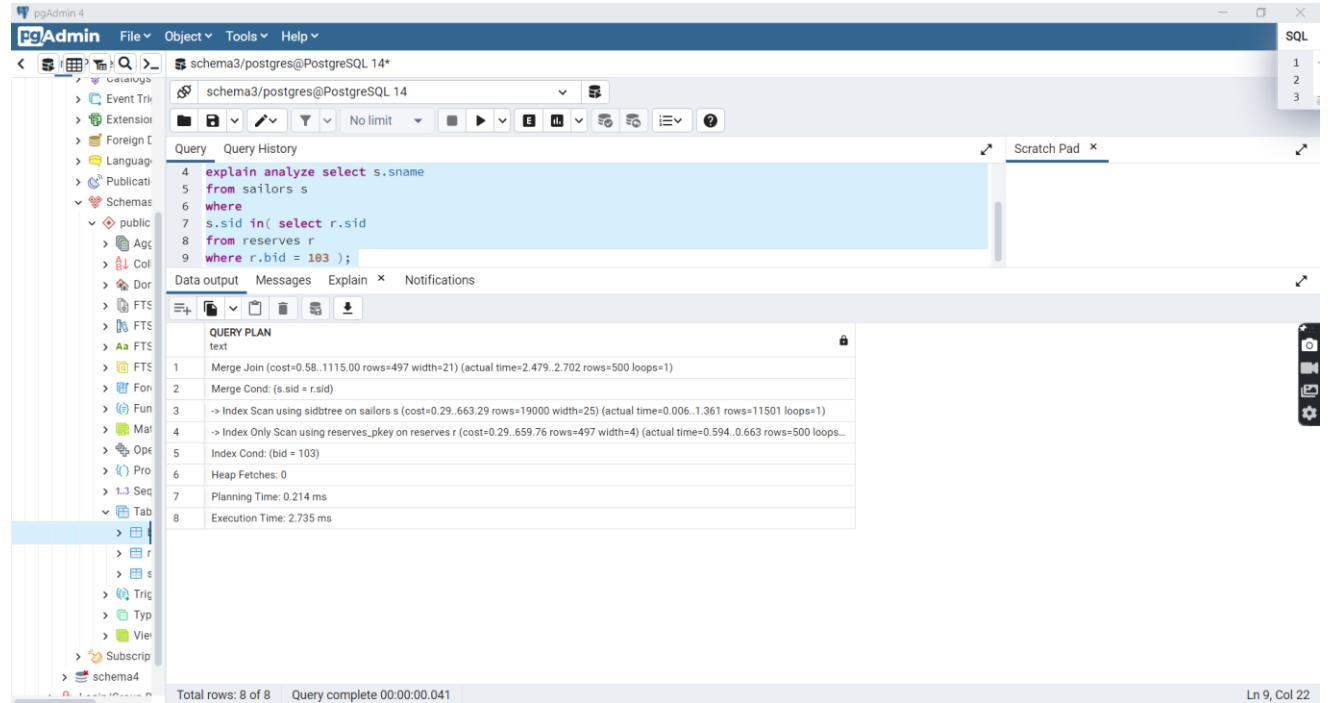
Total rows: 10 of 10 Query complete 00:00:00.087 Ln 6, Col 22

Execution Plan and Costs



Query 7 with B+ Tree on: sailors(sid) Flags[“seqscan = off”, “hashjoin=on”, “mergejoin = on”, sailors_pkey indisvalid = false else are true]

Planning and Execution time



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

- Toolbar:** PgAdmin, File, Object, Tools, Help.
- Object Browser:** Shows databases (schema3/postgres@PostgreSQL 14*), schemas (public, public@schema3), tables (sailors, reserves), and other objects like Event Trig, Foreign L, Language, Publicati, Schemas, FTS, and Tab.
- SQL Editor:** Contains the query:

```
4 explain analyze select s.sname
5 from sailors s
6 where
7 s.sid in( select r.sid
8 from reserves r
9 where r.bid = 103 );
```
- Result Grid:** Displays the query results.
- Query Plan:** Shows the execution plan with the following steps:
 - Merge Join (cost=0.58..1115.00 rows=497 width=21) (actual time=2.479..2.702 rows=500 loops=1)
 - Merge Cond: (s.sid = r.sid)
 - > Index Scan using sidbtree on sailors s (cost=0.29..663.29 rows=19000 width=25) (actual time=0.006..1.361 rows=11501 loops=1)
 - > Index Only Scan using reserves_pkey on reserves r (cost=0.29..659.76 rows=497 width=4) (actual time=0.594..0.663 rows=500 loops=1)
 - Index Cond: (bid = 103)
 - Heap Fetches: 0
 - Planning Time: 0.214 ms
 - Execution Time: 2.735 ms
- Status Bar:** Total rows: 8 of 8 | Query complete 00:00:00.041 | Ln 9, Col 22

Execution Plan and Costs

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

schema3/postgres@PostgreSQL 14*

Query History

```

4 select s.sname
5   from sailors s
6  where
7    s.sid in ( select r.sid
8      from reserves r
9     where r.bid = 103 );

```

Data output Messages Explain Notifications

Graphical Analysis Statistics

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Merge Inner Join (cost=0.58..1115 rows=497 width=21) (actual=3.044..3.319 r...	0.845 ms	3.319 ms	↓ 1.01	500	497	
2.	→ Index Scan using sidbtree on sailors as s (cost=0.29..663.29 rows=19000 ...	1.663 ms	1.663 ms	↑ 1.66	11501	19000	
3.	→ Index Only Scan using reserves_pkey on reserves as r (cost=0.29..659.76 ... Index Cond: (bid = 103)	0.811 ms	0.811 ms	↓ 1.01	500	497	1

Total rows: 1 of 1 Query complete 00:00:00.057 Ln 9, Col 22

B+ tree index optimised here as it was used to search(in O(logn)) in table sailors for every r.sid generated in the subquery

Query 7 with Hash index on: sailors(sid) Flags["seqscan = off", "hashjoin=on", "mergejoin = on", sailors_pkey indisvalid = false else are true]

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

schema3/postgres@PostgreSQL 14*

Query History

```

1 set enable_seqscan = off;
2 set enable_bitmapscan = off;
3 update pg_index set indisvalid = false where indexrelid = 'sailors_pkey':::regclass
4 create index sidhash on sailors using hash(sid);

```

Data output Messages Explain Notifications

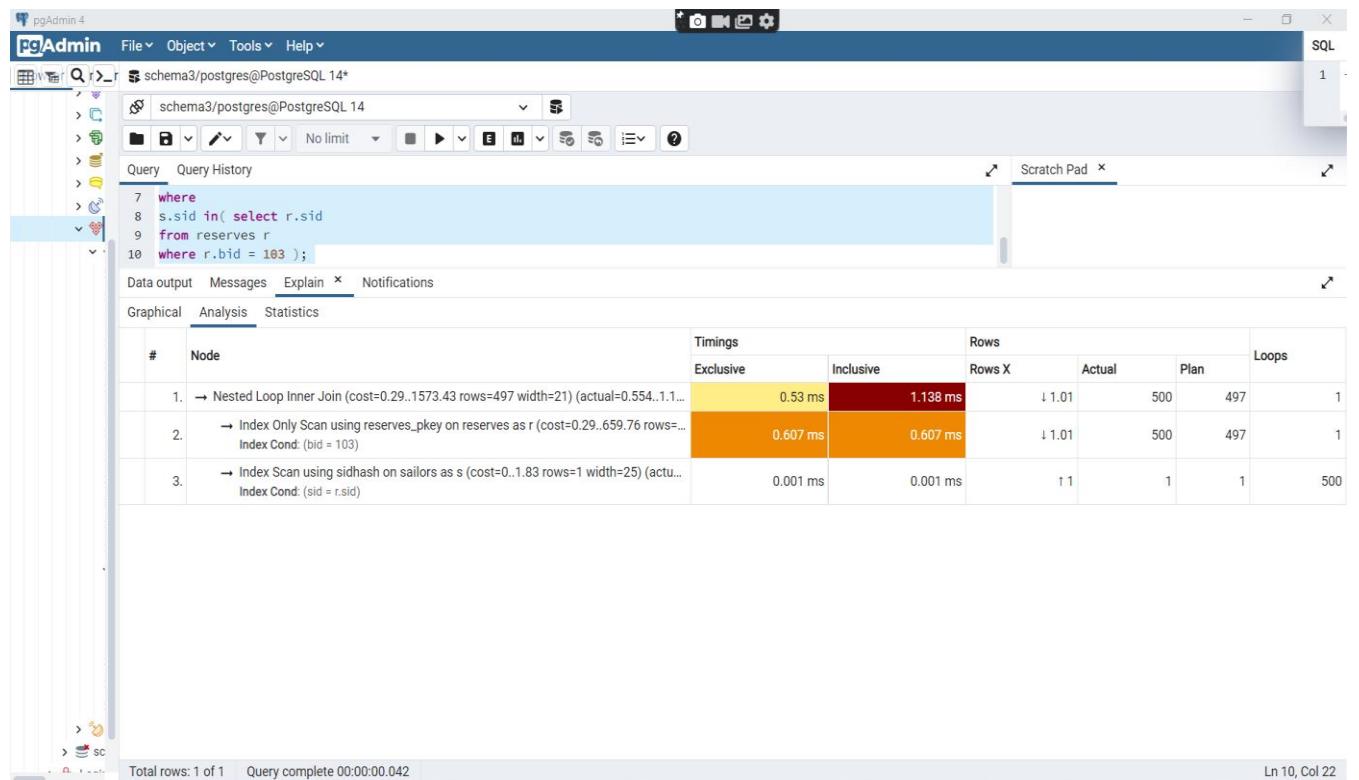
QUERY PLAN

text

1 Nested Loop (cost=0.29..1573.43 rows=497 width=21) (actual time=0.574..1.214 rows=500 loops=1)
2 → Index Only Scan using reserves_pkey on reserves r (cost=0.29..659.76 rows=497 width=4) (actual time=0.562..0.633 rows=500 loops=1)
3 Index Cond: (bid = 103)
4 Heap Fetches: 0
5 → Index Scan using sidhash on sailors s (cost=0.00..1.83 rows=1 width=25) (actual time=0.001..0.001 rows=1 loops=500)
6 Index Cond: (sid = r.sid)
7 Planning Time: 0.250 ms
8 Execution Time: 1.244 ms

Total rows: 8 of 8 Query complete 00:00:00.046 Ln 10, Col 22

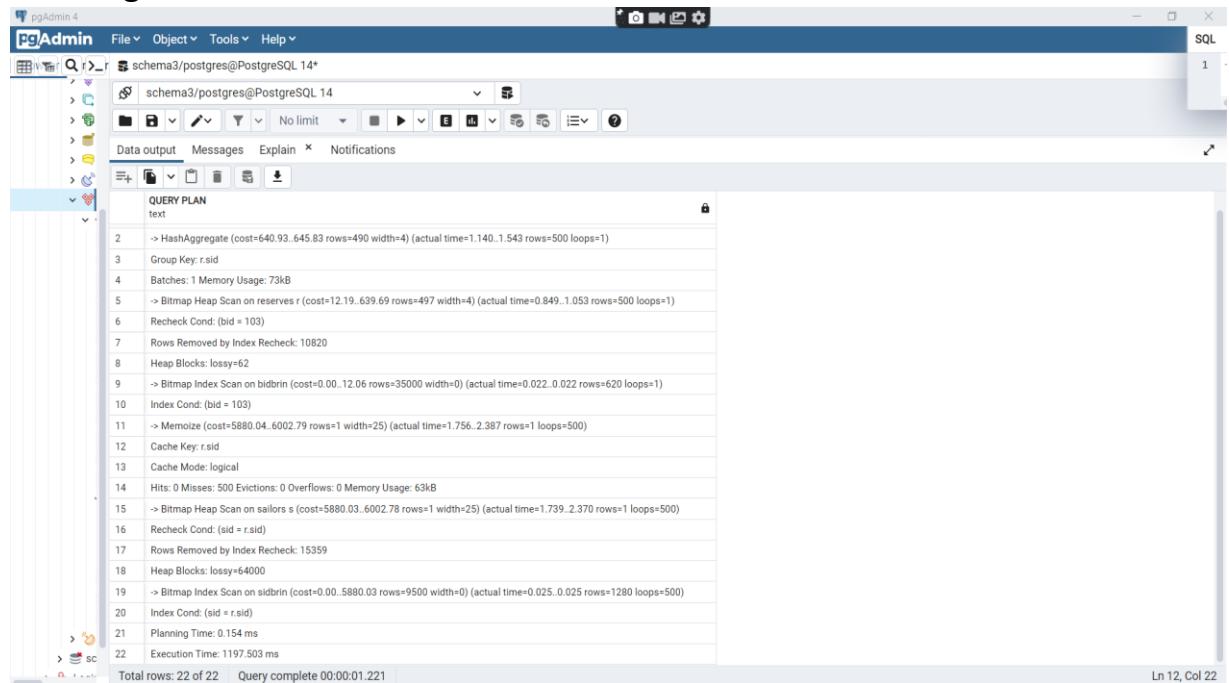
Execution Plan and Costs



Hash index optimised here as it was used to search(in O(1)) in table sailors for every r.sid generated in the subquery

Query 7 with BRIN on: sailors(sid) and Reserves(bid) Flags[“seqscan = off”, “hashjoin=on”, “mergejoin = on”, sailors_pkey indisvalid = false else are true])

Planning and Execution time

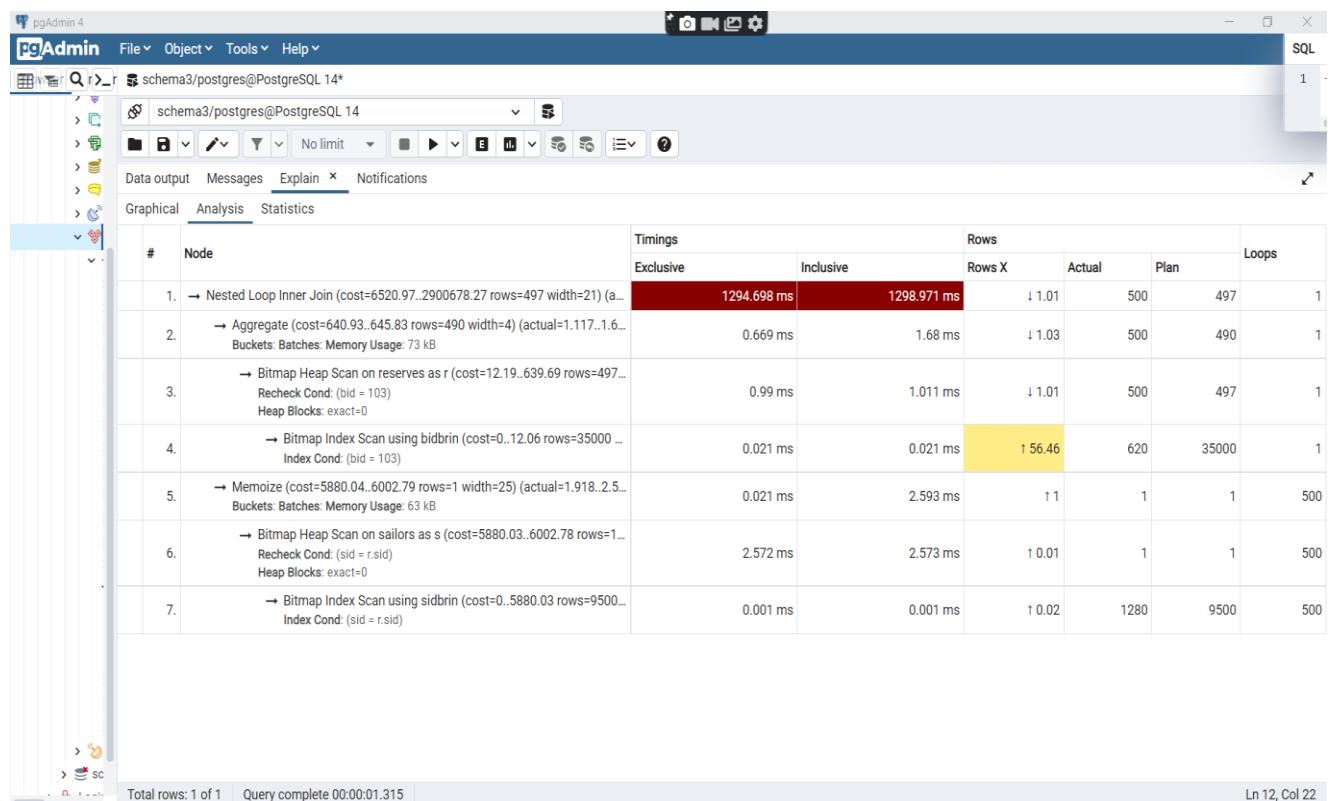


```

pgAdmin 4
File Object Tools Help
schema3/postgres@PostgreSQL 14*
Data output Messages Explain Notifications
QUERY PLAN
text
2. -> HashAggregate (cost=640.93..645.83 rows=490 width=4) (actual time=1.140..1.543 rows=500 loops=1)
   Group Key: r.sid
   Batches: 1 Memory Usage: 73kB
   -> Bitmap Heap Scan on reserves r (cost=12.19..639.69 rows=497 width=4) (actual time=0.849..1.053 rows=500 loops=1)
      Recheck Cond: (bid = 103)
      Rows Removed by Index Recheck: 10820
      Heap Blocks: lossy=2
      -> Bitmap Index Scan on bidbrin (cost=0.00..12.06 rows=35000 width=0) (actual time=0.022..0.022 rows=620 loops=1)
         Index Cond: (bid = 103)
         -> Memoize (cost=5880.04..6002.79 rows=1 width=25) (actual time=1.756..2.387 rows=1 loops=500)
            Cache Key: r.sid
            Cache Mode: logical
            Hits: 0 Misses: 500 Evictions: 0 Overflows: 0 Memory Usage: 63kB
            -> Bitmap Heap Scan on sailors s (cost=5880.03..6002.78 rows=1 width=25) (actual time=1.739..2.370 rows=1 loops=500)
               Recheck Cond: (sid = r.sid)
               Rows Removed by Index Recheck: 15359
               Heap Blocks: lossy=64000
               -> Bitmap Index Scan on sidbrin (cost=0.00..5.880.03 rows=9500 width=0) (actual time=0.025..0.025 rows=1280 loops=500)
                  Index Cond: (sid = r.sid)
                  Planning Time: 0.154 ms
                  Execution Time: 1197.503 ms
Total rows: 22 of 22 Query complete 00:00:01.221
Ln 12, Col 22

```

Execution Plan and Costs



The screenshot shows the pgAdmin 4 interface with the Explain tab selected. The results are displayed in two tabs: Graphical and Analysis. The Graphical tab shows a tree diagram of the query plan, while the Analysis tab provides detailed timing and resource usage data.

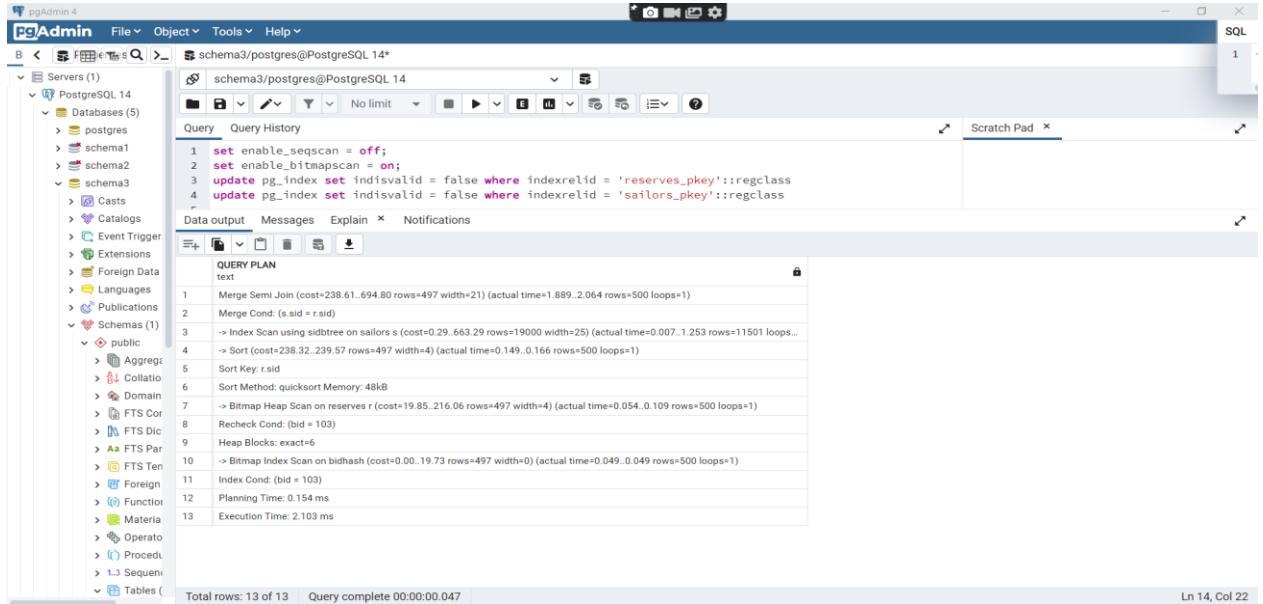
#	Node	Timings		Rows		Loops	
		Exclusive	Inclusive	Rows X	Actual		
1.	-> Nested Loop Inner Join (cost=6520.97..2900678.27 rows=497 width=21) (actual=1.117..1.6...	1294.698 ms	1298.971 ms	↓ 1.01	500	497	1
2.	-> Aggregate (cost=640.93..645.83 rows=490 width=4) (actual=1.117..1.6...	0.669 ms	1.68 ms	↓ 1.03	500	490	1
3.	-> Bitmap Heap Scan on reserves as r (cost=12.19..639.69 rows=497 width=4) (actual=0.849..1.053 rows=500 loops=1)	0.99 ms	1.011 ms	↓ 1.01	500	497	1
4.	-> Bitmap Index Scan using bidbrin (cost=0..12.06 rows=35000 width=0) (actual=0.021 ms)	0.021 ms	0.021 ms	↑ 56.46	620	35000	1
5.	-> Memoize (cost=5880.04..6002.79 rows=1 width=25) (actual=1.918..2.5...	0.021 ms	2.593 ms	↑ 1	1	1	500
6.	-> Bitmap Heap Scan on sailors as s (cost=5880.03..6002.78 rows=1 width=25) (actual=2.572 ms)	2.572 ms	2.573 ms	↑ 1.01	1	1	500
7.	-> Bitmap Index Scan using sidbrin (cost=0..5.880.03 rows=9500 width=0) (actual=0.001 ms)	0.001 ms	0.001 ms	↑ 1.02	1280	9500	500

Total rows: 1 of 1 Query complete 00:00:01.315 Ln 12, Col 22

BRIN index here didn't optimize here as the search on r.bid was an exact value and the search on s.sid was also an exact value and BRIN is suitable for small range queries in a table with a very large range

Query 7 with Hash index on: reserves(bid) and B+ tree index on sailors(sid)
Flags["seqscan = off", "hashjoin=on", "mergejoin = on", sailors_pkey indisvalid = false else are true]

Planning and execution time



```

set enable_seqscan = off;
set enable_bitmapscan = on;
update pg_index set indisvalid = false where indexrelid = 'reserves_pkey'::regclass
update pg_index set indisvalid = false where indexrelid = 'sailors_pkey'::regclass
  
```

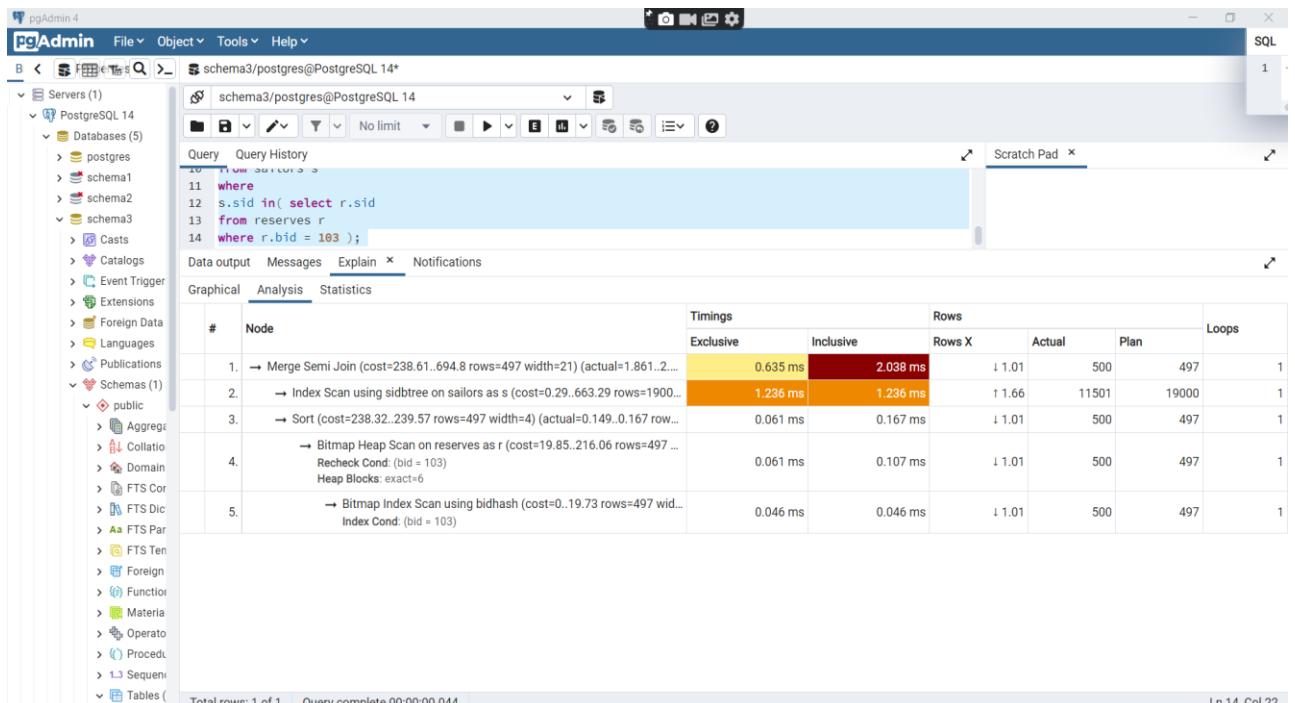
QUERY PLAN

```

1  Merge Semi Join (cost=238.61..694.80 rows=497 width=21) (actual time=1.889..2.064 rows=500 loops=1)
  2  Merge Cond: (s.sid = r.bid)
  3  -> Index Scan using sidtree on sailors s (cost=0.29..663.29 rows=19000 width=25) (actual time=0.007..1.253 rows=11501 loops=1)
  4  -> Sort (cost=238.32..239.57 rows=497 width=4) (actual time=0.149..0.166 rows=500 loops=1)
      Sort Key: r.sid
      Sort Method: quicksort Memory: 48kB
  5  -> Bitmap Heap Scan on reserves r (cost=19.85..216.06 rows=497 width=4) (actual time=0.054..0.109 rows=500 loops=1)
  6  Recheck Cond: (bid = 103)
  7  Heap Blocks: exact=6
  8
  9
  10 -> Bitmap Index Scan on bidhash (cost=0..0.19..19.73 rows=497 width=0) (actual time=0.049..0.049 rows=500 loops=1)
  11 Index Cond: (bid = 103)
  12 Planning Time: 0.154 ms
  13 Execution Time: 2.103 ms
  
```

Total rows: 13 of 13 Query complete 00:00:00.047 Ln 14, Col 22

Execution Plan and Costs



```

where
s.sid in( select r.sid
from reserves r
where r.bid = 103 );
  
```

Graphical

#	Node	Timings	Rows	Loops			
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Merge Semi Join (cost=238.61..694.8 rows=497 width=21) (actual=1.861..2.038 ms)	0.635 ms	2.038 ms	↓ 1.01	500	497	1
2.	→ Index Scan using sidtree on sailors s (cost=0.29..663.29 rows=19000 width=25)	1.236 ms	1.236 ms	↓ 1.66	11501	19000	1
3.	→ Sort (cost=238.32..239.57 rows=497 width=4) (actual=0.149..0.167 ms)	0.061 ms	0.167 ms	↓ 1.01	500	497	1
4.	→ Bitmap Heap Scan on reserves r (cost=19.85..216.06 rows=497 width=4)	0.061 ms	0.107 ms	↓ 1.01	500	497	1
5.	→ Bitmap Index Scan using bidhash (cost=0..0.19..19.73 rows=497 width=0)	0.046 ms	0.046 ms	↓ 1.01	500	497	1

Total rows: 1 of 1 Query complete 00:00:00.044 Ln 14, Col 22

Since the search on r.bid = 103 is an exact value and search on s.sid is an exact value having a hash index(search time = O(1)) on reserves.bid and B+tree index(search time = O(logn)) on sailors.sid improved query performance

Optimised Query 7 SQL:

```
select s.sname  
from sailors s  
where exists ( select r.sid  
    from reserves r  
    where s.sid = r.sid and r.bid = 103 );
```

Reason: This query is optimised as it uses exists instead of in which reduces search time as SQL engine will terminate scanning process once a match is found but in the original query with “in” all records from inner query will be fetched

Optimised Query 7 with no index Flags[“seqscan = on”, “hashjoin=on”, “mergejoin = on”, all primary keyinvalid = true])

Planning and Execution time

pgAdmin 4

File Object Tools Help SQL

schema3/postgres@PostgreSQL 14*

Query History

```
6 from sailors s
7 where exists ( select r.sid
8         from reserves r
9         where s.sid = r.sid and r.bid = 103 );
10
```

Data output Messages Explain Notifications

QUERY PLAN

text

1 Hash Join (cost=633.71..1032.60 rows=497 width=21) (actual time=3.415..4.380 rows=500 loops=1)
2 Hash Cond: (s.sid = r.sid)
3 -> Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.012..0.916 rows=19000 loop...
4 -> Hash (cost=627.50..627.50 rows=497 width=4) (actual time=2.128..2.128 rows=500 loops=1)
5 Buckets: 1024 Batches: 1 Memory Usage: 26kB
6 -> Seq Scan on reserves r (cost=0.00..627.50 rows=497 width=4) (actual time=1.948..2.081 rows=500 loops=1)
7 Filter: (bid = 103)
8 Rows Removed by Filter: 34500
9 Planning Time: 0.184 ms
10 Execution Time: 4.407 ms

Triggers Types Views Subscriptions

schema4

Login/Group Roles

Total rows: 10 of 10 Query complete 00:00:00.074 Ln 12, Col 1

The screenshot shows the pgAdmin 4 interface with a query editor and a query plan window. The query is a subquery that checks if a sailor has a specific bid. The query plan details the execution steps, including a Hash Join, Seq Scan on sailors, Hash, and Seq Scan on reserves, along with planning and execution times.

Execution Plan and Costs

The screenshot shows the pgAdmin 4 interface with a query editor and an explain plan viewer. The query is:

```
from reserves r
where s.sid = r.sid
and r.bid = 103;
```

The explain plan shows the following execution flow:

- Hash Cond: (s.sid = r.sid)
- Seq Scan on sailors as s (cost=0.349 rows=19000 width=25) (actual=0.0...)
- Hash (cost=627.5..627.5 rows=497 width=4) (actual=2.315..2.318 rows=...)
- Seq Scan on reserves as r (cost=0..627.5 rows=497 width=4) (actual...)

Timing details from the explain plan:

Node	Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	1.435 ms	4.81 ms	1.01	500	497	1
2.	1.057 ms	1.057 ms	1.01	19000	19000	1
3.	0.055 ms	2.318 ms	1.01	500	497	1
4.	2.264 ms	2.264 ms	1.01	500	497	1

Message bar: Successfully run. Total query runtime: 66 msec. 1 rows affected.

Optimised Query 7 with B+ tree index on: sailors(sid) Flags[“seqscan = off”, “hashjoin=on”, “mergejoin = on”, sailors_pkey indisvalid = false else are true])

Planning and Execution time

PgAdmin 4

Query

```

1 set enable_seqscan = off;
2 set enable_bitmapscan = off;
3 update pg_index sert indisvalid = false where indexrelid = ''::regclass
4 create index sidbtree on sailors using btree(sid);

6 explain analyze select s.sname
7 from sailors s
8 where exists ( select r.sid
9                 from reserves r
10                  where s.sid = r.sid and r.bid = 103 );

```

QUERY PLAN

```

text
1 Merge Join (cost=0.58..1115.00 rows=497 width=21) (actual time=2.542..2.765 rows=500 loops=1)
2 Merge Cond: (s.sid = r.sid)
3   -> Index Scan using sidbtree on sailors s (cost=0.29..663.29 rows=19000 width=25) (actual time=0.007..1.506 rows=11501 loops=1)
4   -> Index Only Scan using reserves_pkey on reserves r (cost=0.29..659.76 rows=497 width=4) (actual time=0.555..0.622 rows=500 loops=1)
5   Index Cond: (bid = 103)
6   Heap Fetches: 0
7   Planning Time: 0.571 ms
8   Execution Time: 2.797 ms

```

Total rows: 8 of 8 | Query complete 00:00:00.043 | Ln 10, Col 55

Execution Plan and Costs

PgAdmin 4

Query

```

1 set enable_seqscan = off;
2 set enable_bitmapscan = off;
3 update pg_index sert indisvalid = false where indexrelid = ''::regclass
4 create index sidbtree on sailors using btree(sid);

6 select s.sname
7 from sailors s
8 where exists ( select r.sid
9                 from reserves r
10                where s.sid = r.sid and r.bid = 103 );

```

Explain

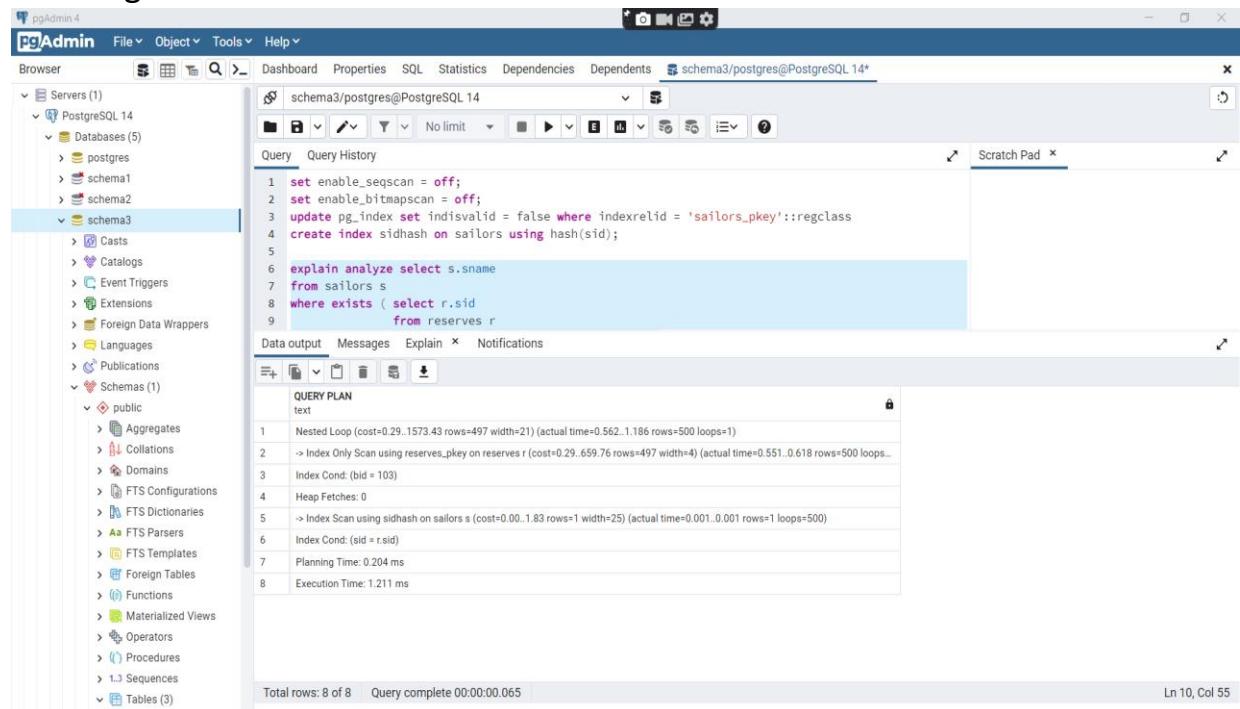
#	Node	Rows	Loops
1.	→ Merge Inner Join (rows=500 loops=1)	500	1
2.	→ Index Scan using sidbtree on sailors as s (rows=11501 loops=1)	11501	1
3.	→ Index Only Scan using reserves_pkey on reserves as r (rows=500 loops=1) Index Cond: (bid = 103)	500	1

Total rows: 1 of 1 | Query complete 00:00:00.055 | Ln 10, Col 55

B+ tree index optimised here as it was used to search(in O(logn)) in table sailors for every r.sid = s.sid generated in the subquery

Optimised Query 7 with hash index on: sailors(sid) Flags[“seqscan = off”, “hashjoin=on”, “mergejoin = on”, sailors_pkey indisvalid = false else are true])

Planning and Execution time



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

- Servers:** PostgreSQL 14 (selected)
- Databases:** postgres, schema1, schema2, schema3 (selected)
- Query Editor:**

```

1 set enable_seqscan = off;
2 set enable_bitmapscan = off;
3 update pg_index set indisvalid = false where indexrelid = 'sailors_pkey'::regclass
4 create index sidhash on sailors using hash(sid);

6 explain analyze select s.sname
7 from sailors s
8 where exists ( select r.sid
                  from reserves r
                )
```
- Query Plan:**

	QUERY PLAN
1	Nested Loop (cost=0.29..1573.43 rows=497 width=21) (actual time=0.562..1.186 rows=500 loops=1) > Index Only Scan using reserves_pkey on reserves r (cost=0.29..659.76 rows=497 width=4) (actual time=0.551..0.618 rows=500 loops=1) 3 Index Cond: (bid = 103) 4 Heap Fetches: 0 5 > Index Scan using sidhash on sailors s (cost=0.00..1.83 rows=1 width=25) (actual time=0.001..0.001 rows=1 loops=500) 6 Index Cond: (sid = r.sid) 7 Planning Time: 0.204 ms 8 Execution Time: 1.211 ms
- Message Bar:** Total rows: 8 of 8 Query complete 00:00:00.065 Ln 10, Col 55

Execution Plan and Costs

```

2 set enable_bitmapscan = off;
3 update pg_index set indisvalid = false where indexrelid = 'sailors_pkey'::regclass
4 create index sidhash on sailors using hash(sid);

5
6 select s.sname
7 from sailors s
8 where exists ( select r.sid
9 from sailors r
10 where r.sid = s.sid )

```

The execution plan shows a Nested Loop Inner Join. The first step (Node 1) is an Index Only Scan using reserves_pkey on reserves as r (rows=500 loops=1). The second step (Node 2) is an Index Scan using sidhash on sailors as s (rows=1 loops=500). The third step (Node 3) is another Index Scan using sidhash on sailors as s (rows=1 loops=500).

#	Node	Rows Actual	Loops
1.	→ Nested Loop Inner Join (rows=500 loops=1)	500	1
2.	→ Index Only Scan using reserves_pkey on reserves as r (rows=500 loops=1) Index Cond: (bid = 103)	500	1
3.	→ Index Scan using sidhash on sailors as s (rows=1 loops=500) Index Cond: (sid = r.sid)	1	500

Hash index optimised here as it was used to search(in O(1)) in table sailors for every r.sid = s.sid generated in the subquery

Optimised Query 7 with BRIN on sailors(sid) and reserves(bid) Flags["seqscan = off","hashjoin=on", "mergejoin = on", sailors_pkey indisvalid = false else are true])

Planning and Execution time

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

- Left Panel (Servers):** Shows a tree view of servers and databases. The current database selected is "schema3@PostgreSQL_14".
- Top Bar:** Includes "File", "Object", "Tools", and "Help" menus.
- Toolbar:** Includes icons for Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a search bar.
- Central Area:**
 - Query Plan:** A detailed breakdown of the query execution plan, starting with a HAVING aggregate and involving multiple tables and operations like Bitmap Heap Scans and Index Scans.
 - Data Output:** Shows the total rows processed (22) and the completion time (00:00:00.694).
- Bottom Right:** Displays the page number "Ln 12, Col 55".

Execution Plan and Costs

PgAdmin 4

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

- Left Sidebar (Browser):**
 - Servers (1) > PostgreSQL 14 > schema3
 - Databases (5) > postgres, schema1, schema2
 - Tables (3)
 - Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (1) > public, Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables (3)
- Main Window:**
 - Tab: schema3/postgres@PostgreSQL 14*
 - Toolbar: Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Explain, Notifications.
 - Panel: Query History, Data output, Messages, Explain (selected), Notifications.
 - Scratch Pad.
 - Graphical, Analysis, Statistics tabs.
 - Table: Rows (Actual vs Loops) for the execution plan steps.
 - Execution Plan (Explain):

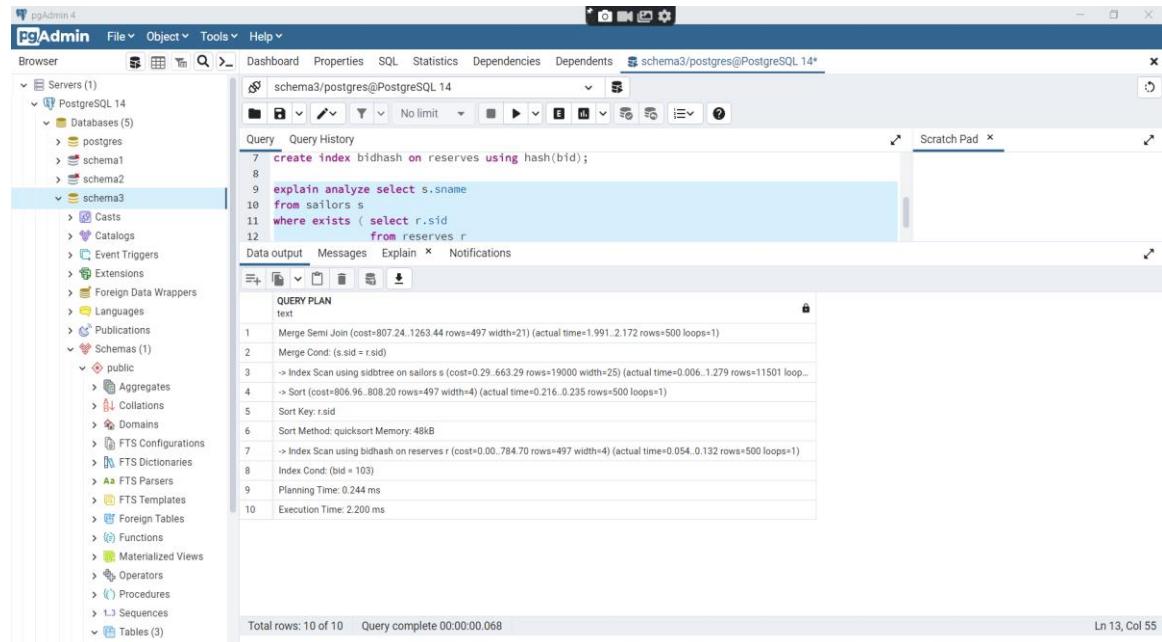
#	Node	Rows	Loops
1.	→ Nested Loop Inner Join (rows=500 loops=1)	500	1
2.	→ Aggregate (rows=500 loops=1) Buckets: Batches: Memory Usage: 73 kB	500	1
3.	→ Bitmap Heap Scan on reserves as r (rows=500 loops=1) Recheck Cond: (bid = 103) Heap Blocks: exact=0	500	1
4.	→ Bitmap Index Scan using bidbrin (rows=620 loops=1) Index Cond: (bid = 103)	620	1
5.	→ Memoize (rows=1 loops=500) Buckets: Batches: Memory Usage: 63 kB	1	500
6.	→ Bitmap Heap Scan on sailors as s (rows=1 loops=500) Recheck Cond: (sid = r.sid) Heap Blocks: exact=0	1	500
7.	→ Bitmap Index Scan using sidbrin (rows=1280 loops=500) Index Cond: (sid = r.sid)	1280	500
 - Total rows: 1 of 1 Query complete 00:00:00.708 Ln 12, Col 55

BRIN index here didn't optimize here as the search on r.bid was an exact value and the

search on s.sid was also an exact value and BRIN is suitable for small range queries in a table with a very large range

Optimised Query 7 with B+ tree index on sailors(sid) and hash index on reserves(bid) Flags[“seqscan = off”, “hashjoin=on”, “mergejoin = on”, sailors_pkey and reserves_pkey indisvalid = false else are true]

Planning and Execution time



The screenshot shows the pgAdmin 4 interface. On the left, the browser pane displays the database structure under 'schema3'. In the main pane, a query is being run:

```
7 create index bidhash on reserves using hash(bid);
8
9 explain analyze select s.sname
10 from sailors s
11 where exists ( select r.sid
12   from reserves r
```

Below the query, the 'QUERY PLAN' section shows the execution plan:

```
text
1 Merge Semi Join (cost=807.24..1263.44 rows=497 width=21) (actual time=1.991..2.172 rows=500 loops=1)
2  Merge Cond: (s.sid = r.sid)
3    -> Index Scan using sidtree on sailors s (cost=0.29..663.29 rows=19000 width=25) (actual time=0.006..1.279 rows=11501 loops=1)
4      -> Sort (cost=806.96..808.20 rows=497 width=4) (actual time=0.216..0.235 rows=500 loops=1)
5    Sort Key: r.sid
6    Sort Method: quicksort Memory: 48kB
7    -> Index Scan using bidhash on reserves r (cost=0.00..784.70 rows=497 width=4) (actual time=0.054..0.132 rows=500 loops=1)
8  Index Cond: (bid = 103)
9  Planning Time: 0.244 ms
10 Execution Time: 2.200 ms
```

Total rows: 10 of 10 Query complete 00:00:00.068 Ln 13, Col 55

Execution Plan and Costs

S

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree view is expanded to show 'PostgreSQL 14' with 'Databases' (including 'postgres', 'schema1', 'schema2', and 'schema3'). 'schema3' is selected. The main area contains a query editor window with the following SQL code:

```
9 select s.sname
10 from sailors s
11 where exists ( select r.sid
12                  from reserves r
13                 where s.sid = r.sid and r.bid = 103 );
```

Below the query is an 'Explain' tab showing the execution plan:

#	Node	Rows	Loops
1.	→ Merge Semi Join (rows=500 loops=1)	500	1
2.	→ Index Scan using sidtree on sailors as s (rows=11501 loops=1)	11501	1
3.	→ Sort (rows=500 loops=1)	500	1
4.	→ Index Scan using bidhash on reserves as r (rows=500 loops=1) Index Cond: (bid = 103)	500	1

At the bottom of the pgAdmin window, status messages indicate: 'Total rows: 1 of 1 | Query complete 00:00:00.044' and 'Ln 13, Col 55'.

Since the search on `r.bid = 103` is an exact value and search on `s.sid = r.sid` is an exact value having a hash index(search time = $O(1)$) on `reserves.bid` and B+tree index(search time = $O(\log n)$) on `sailors.sid` improved query performance

Query 8 with no index Flags[“seqscan = on”, “hashjoin=on”, “mergejoin = on”, all primary key indisvalid = true])

Planning and Execution time

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

- Servers:** PostgreSQL 14 (selected)
- Databases:** postres, schema1, schema2, schema3 (selected)
- Query:** `update pg_index set indisvalid = false where indexrelid = '::regclass'`
- Explain Plan:**

```

1  Hash Semi Join (cost=931.53..1463.74 rows=11985 width=21) (actual time=9.275..10.740 rows=500 loops=1)
  2  Hash Cond: (s.sid = r.rid)
  3    -> Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.018..1.428 rows=19000 loops=1)
  4    -> Hash (cost=713.22..713.22 rows=17465 width=4) (actual time=7.107..7.109 rows=500 loops=1)
  5      Buckets: 32768 Batches: 1 Memory Usage: 274 kB
  6    -> Hash Join (cost=81.21..713.22 rows=17465 width=4) (actual time=6.585..7.003 rows=500 loops=1)
  7    Hash Cond: (r.bid = b.bid)
  8    -> Seq Scan on reserves r (cost=0.00..540.00 rows=35000 width=8) (actual time=0.013..2.597 rows=35000 loops=1)
  9    -> Hash (cost=62.50..62.50 rows=1497 width=4) (actual time=0.701..0.702 rows=1497 loops=1)
 10   Buckets: 2048 Batches: 1 Memory Usage: 69 kB
 11   -> Seq Scan on boat b (cost=0.00..62.50 rows=1497 width=4) (actual time=0.015..0.492 rows=1497 loops=1)
 12   Filter: (color = red::bpchar)
 13   Rows Removed by Filter: 1503
 14   Planning Time: 0.408 ms
 15   Execution Time: 10.863 ms
  
```
- Statistics:** Total rows: 15 of 15 | Query complete 00:00:00.068 | Ln 11, Col 59

Execution Plan and Costs

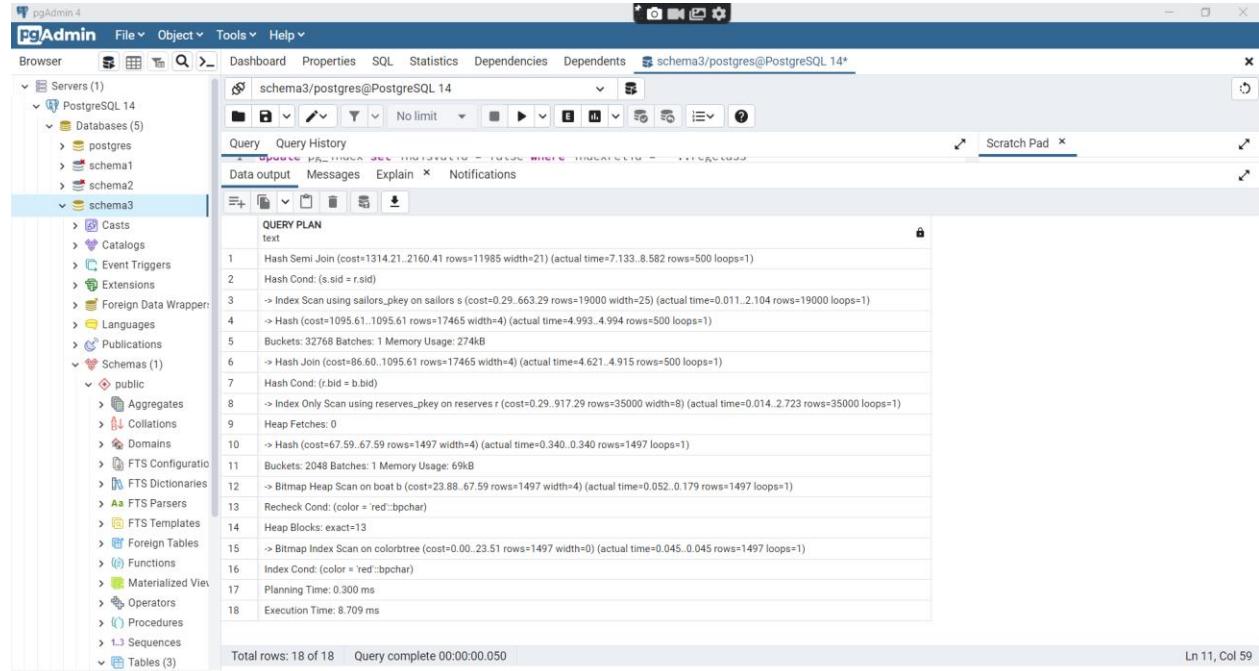
The screenshot shows the pgAdmin 4 interface with the following details:

- Servers:** PostgreSQL 14 (selected)
- Databases:** postres, schema1, schema2, schema3 (selected)
- Query:** `set enable_seqscan = on;`
- Explain Plan:**

#	Node	Rows	Actual	Loops	
1.	→ Hash Semi Join (rows=500 loops=1) Hash Cond: (s.sid = r.rid)			500	1
2.	→ Seq Scan on sailors as s (rows=19000 loops=1)			19000	1
3.	→ Hash (rows=500 loops=1) Buckets: 32768 Batches: 1 Memory Usage: 274 kB			500	1
4.	→ Hash Inner Join (rows=500 loops=1) Hash Cond: (r.bid = b.bid)			500	1
5.	→ Seq Scan on reserves as r (rows=35000 loops=1)			35000	1
6.	→ Hash (rows=1497 loops=1) Buckets: 2048 Batches: 1 Memory Usage: 69 kB			1497	1
7.	→ Seq Scan on boat as b (rows=1497 loops=1) Filter: (color = red::bpchar) Rows Removed by Filter: 1503			1497	1
- Statistics:** Total rows: 1 of 1 | Query complete 00:00:00.065 | Ln 11, Col 59

Query 8 with B+ tree index on: boat(color) Flags["seqscan = off", "hashjoin=on", "mergejoin = on", boat_pkey indisvalid = false else are true]

Planning and Execution time



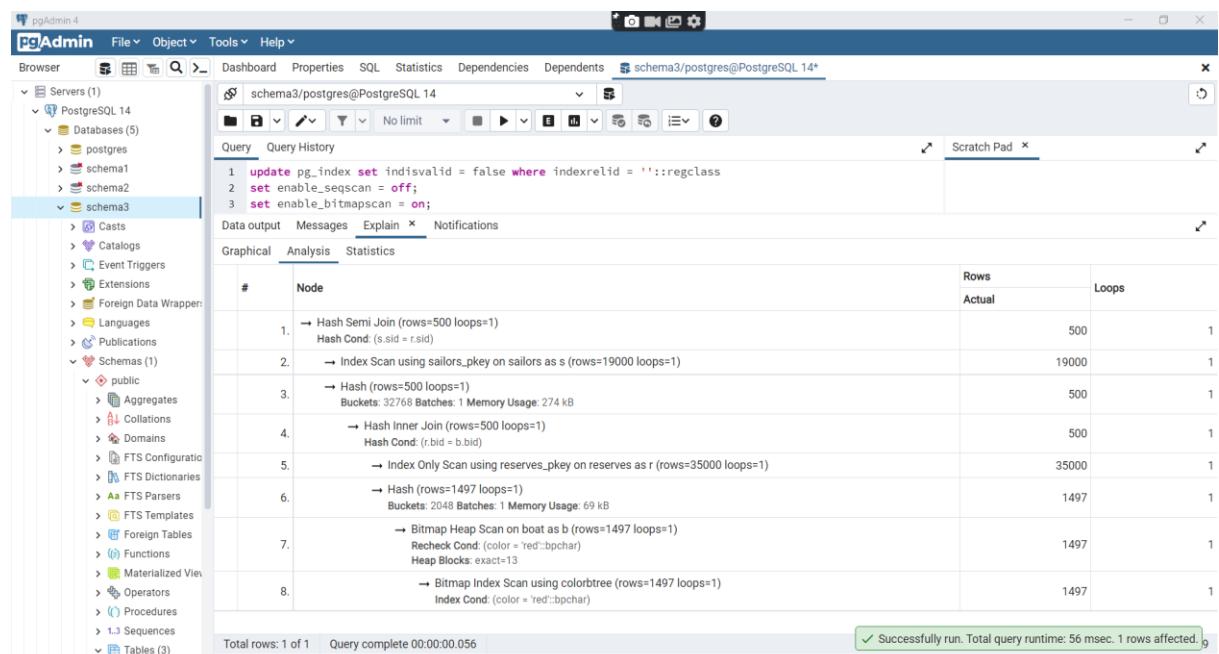
```

pgAdmin 4
PgAdmin File Object Tools Help
Browser Dashboard Properties SQL Statistics Dependencies Dependents schema3/postgres@PostgreSQL 14*
Servers (1)
  PostgreSQL 14
    Databases (5)
      postgres
      schema1
      schema2
      schema3
        Casts
        Catalogs
        Event Triggers
        Extensions
        Foreign Data Wrappers
        Languages
        Publications
        Schemas (1)
          public
          Aggregates
          Collations
          Domains
          FTS Configuration
          FTS Dictionaries
          FTS Parsers
          FTS Templates
          Foreign Tables
          Functions
          Materialized Views
          Operators
          Procedures
          Sequences
          Tables (3)
Query History
Data output Messages Explain Notifications
QUERY PLAN
text
1 Hash Semi Join (cost=1314.21..2160.41 rows=11985 width=21) (actual time=7.133..8.582 rows=500 loops=1)
  2 Hash Cond: (s.sid = r.rid)
  3 -> Index Scan using sailors_pkey on sailors s (cost=0.29..663.29 rows=19000 width=25) (actual time=0.011..2.104 rows=19000 loops=1)
  4 -> Hash (cost=1095.61..1095.61 rows=17465 width=4) (actual time=4.993..4.994 rows=500 loops=1)
  5 Buckets: 32768 Batches: 1 Memory Usage: 274kB
  6 -> Hash Join (cost=86.60..1095.61 rows=17465 width=4) (actual time=4.621..4.915 rows=500 loops=1)
  7 Hash Cond: (r.bid = b.bid)
  8 -> Index Only Scan using reserves_pkey on reserves r (cost=0.29..917.29 rows=35000 width=8) (actual time=0.014..2.723 rows=35000 loops=1)
  9 Heap Fetches: 0
  10 -> Hash (cost=67.59..67.59 rows=1497 width=4) (actual time=0.340..0.340 rows=1497 loops=1)
  11 Buckets: 2048 Batches: 1 Memory Usage: 69kB
  12 -> Bitmap Heap Scan on boat b (cost=23.88..67.59 rows=1497 width=4) (actual time=0.052..0.179 rows=1497 loops=1)
  13 Recheck Cond: (color = 'red'::bpchar)
  14 Heap Blocks: exact=13
  15 -> Bitmap Index Scan on colorbtree (cost=0.00..23.51 rows=1497 width=0) (actual time=0.045..0.045 rows=1497 loops=1)
  16 Index Cond: (color = 'red'::bpchar)
  17 Planning Time: 0.300 ms
  18 Execution Time: 8.709 ms

Total rows: 18 of 18 Query complete 00:00:00.050
Ln 11, Col 59

```

Execution Plan and Costs



```

pgAdmin 4
PgAdmin File Object Tools Help
Browser Dashboard Properties SQL Statistics Dependencies Dependents schema3/postgres@PostgreSQL 14*
Servers (1)
  PostgreSQL 14
    Databases (5)
      postgres
      schema1
      schema2
      schema3
        Casts
        Catalogs
        Event Triggers
        Extensions
        Foreign Data Wrappers
        Languages
        Publications
        Schemas (1)
          public
          Aggregates
          Collations
          Domains
          FTS Configuration
          FTS Dictionaries
          FTS Parsers
          FTS Templates
          Foreign Tables
          Functions
          Materialized Views
          Operators
          Procedures
          Sequences
          Tables (3)
Query History
Data output Messages Explain Notifications
Graphical Analysis Statistics
# Node
Rows
Actual Loops
1. -> Hash Semi Join (rows=500 loops=1)
  Hash Cond: (s.sid = r.rid)
  2. -> Index Scan using sailors_pkey on sailors s (rows=19000 loops=1)
  3. -> Hash (rows=500 loops=1)
    Buckets: 32768 Batches: 1 Memory Usage: 274 kB
  4. -> Hash Inner Join (rows=500 loops=1)
    Hash Cond: (r.bid = b.bid)
  5. -> Index Only Scan using reserves_pkey on reserves r (rows=35000 loops=1)
  6. -> Hash (rows=1497 loops=1)
    Buckets: 2048 Batches: 1 Memory Usage: 69 kB
  7. -> Bitmap Heap Scan on boat b (rows=1497 loops=1)
    Recheck Cond: (color = 'red'::bpchar)
    Heap Blocks: exact=13
  8. -> Bitmap Index Scan using colorbtree (rows=1497 loops=1)
    Index Cond: (color = 'red'::bpchar)

Total rows: 1 of 1 Query complete 00:00:00.056
Successfully run. Total query runtime: 56 msec. 1 rows affected.

```

B+tree index optimised here as it was used to search(in O(logn)) in table boatfor every boat.color = 'red' in the subquery

Query 8 with hash index on boat(color) Flags["seqscan = off", "hashjoin=on", "mergejoin = on", boat_pkey indisvalid = false else are true]

Planning and Execution time

```

pgAdmin 4
File Object Tools Help
Browser File Object Tools Help
Servers (1)
PostgreSQL 14 Databases (5)
  postgres schema1 schema2 schema3
    Casts Catalogs Event Triggers Extensions Foreign Data Wrapper Languages Publications Schemas (1)
      public Aggregates Collations Domains FTS Configuration FTS Dictionaries FTS Parsers FTS Templates Foreign Tables Functions Materialized Views Operators Procedures Sequences Tables (3)
schema3/postgres@PostgreSQL 14*
Query History Data output Messages Explain Notifications
Scratch Pad

QUERY PLAN
text
1 Hash Semi Join (cost=1544.09..2390.30 rows=11985 width=21) (actual time=7.137..8.644 rows=500 loops=1)
  Hash Cond: (s.sid = r.sid)
  3 -> Index Scan using sailors_pkey on sailors s (cost=0.29..663.29 rows=19000 width=25) (actual time=0.009..2.128 rows=19000 loops=1)
  4 -> Hash (cost=1325.49..1325.49 rows=17465 width=4) (actual time=5.058..5.060 rows=500 loops=1)
  5 Buckets: 32768 Batches: 1 Memory Usage: 274kB
  6 -> Hash Semi Join (cost=122.32..1325.49 rows=17465 width=4) (actual time=4.697..4.998 rows=500 loops=1)
  7 Hash Cond: (r.bid = b.bid)
  8 -> Index Only Scan using reserves_pkey on reserves r (cost=0.29..917.29 rows=35000 width=8) (actual time=0.014..2.714 rows=35000 loops=1)
  9 Heap Fetches: 0
  10 -> Hash (cost=103.31..103.31 rows=1497 width=4) (actual time=0.444..0.444 rows=1497 loops=1)
  11 Buckets: 2048 Batches: 1 Memory Usage: 69kB
  12 -> Bitmap Heap Scan on boat b (cost=59.60..103.31 rows=1497 width=4) (actual time=0.080..0.283 rows=1497 loops=1)
  13 Recheck Cond: (color = 'red'::bpchar)
  14 Heap Blocks: exact=13
  15 -> Bitmap Index Scan on colorhash (cost=0.00..59.23 rows=1497 width=0) (actual time=0.071..0.071 rows=1497 loops=1)
  16 Index Cond: (color = 'red'::bpchar)
  17 Planning Time: 0.483 ms
  18 Execution Time: 8.818 ms

Total rows: 18 of 18 Query complete 00:00:00.046
Ln 12, Col 59
  
```

Execution Plan and Costs

#	Node	Rows Actual	Loops
1.	-> Hash Semi Join (rows=500 loops=1) Hash Cond: (s.sid = r.sid)	500	1
2.	-> Index Scan using sailors_pkey on sailors s (rows=19000 loops=1)	19000	1
3.	-> Hash (rows=500 loops=1) Buckets: 32768 Batches: 1 Memory Usage: 274 kB	500	1
4.	-> Hash Semi Join (rows=500 loops=1) Hash Cond: (r.bid = b.bid)	500	1
5.	-> Index Only Scan using reserves_pkey on reserves r (rows=35000 loops=1)	35000	1
6.	-> Hash (rows=1497 loops=1) Buckets: 2048 Batches: 1 Memory Usage: 69 kB	1497	1
7.	-> Bitmap Heap Scan on boat as b (rows=1497 loops=1) Recheck Cond: (color = 'red'::bpchar) Heap Blocks: exact=13	1497	1
8.	-> Bitmap Index Scan using colorhash (rows=1497 loops=1) Index Cond: (color = 'red'::bpchar)	1497	1

Total rows: 1 of 1 Query complete 00:00:00.079
Ln 12, Col 59

Hash index optimised here as it was used to search(in O(1)) in table boat for every boat.color = 'red' in the subquery

Query 8 with BRIN on boat(color) Flags[“seqscan = off”, “hashjoin=on”, “mergejoin = on”, boat_pkey indisvalid = false else are true]

Planning and Execution time

The screenshot shows the pgAdmin 4 interface. On the left, the object browser tree is visible, with the 'schema3' node selected. The main pane displays the execution plan for a query, which is a Hash Semi Join. The plan details the cost and execution time for each step, including index scans, hash joins, and bitmap scans. The status bar at the bottom indicates 'Total rows: 19 of 19' and 'Query complete 00:00:00.061'.

Step	Operation	Cost	Rows	Width	Time
1	Hash Semi Join	1515.68	2361.89	21	(actual time=11.319..13.617 rows=500 loops=1)
2	Hash Cond: (s.sid = r.sid)				
3	-> Index Scan using sailors_pkey on sailors s	0.29	663.29	25	(actual time=0.014..3.290 rows=19000 loops=1)
4	-> Hash	1297.08	1297.08	4	(actual time=7.910..7.912 rows=500 loops=1)
5	Buckets: 32768 Batches: 1 Memory Usage: 274kB				
6	-> Hash Semi Join	93.91..1297.08	17465	4	(actual time=7.361..7.814 rows=500 loops=1)
7	Hash Cond: (r.bid = b.bid)				
8	-> Index Only Scan using reserves_pkey on reserves r	0.29..917.29	35000	8	(actual time=0.023..4.120 rows=35000 loops=1)
9	Heap Fetches: 0				
10	-> Hash	74.91..74.91	1497	4	(actual time=0.850..0.852 rows=1497 loops=1)
11	Buckets: 2048 Batches: 1 Memory Usage: 69kB				
12	-> Bitmap Heap Scan on boat b	12.41..74.91	1497	4	(actual time=0.049..0.597 rows=1497 loops=1)
13	Recheck Cond: (color = red::bpchar)				
14	Rows Removed by Index Recheck: 1503				
15	Heap Blocks: lossy=25				
16	-> Bitmap Index Scan on colorbrin	0.00..12.03	3000	0	(actual time=0.038..0.038 rows=250 loops=1)
17	Index Cond: (color = red::bpchar)				
18	Planning Time: 0.403 ms				
19	Execution Time: 13.808 ms				

Total rows: 19 of 19 Query complete 00:00:00.061 Ln 12, Col 59

Execution Plan and Costs

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Servers (1) PostgreSQL 14 Databases (5) postgres schema1 schema2 schema3

Query History 2 set enable_seqscan = off;

Query Explain Notifications

Graphical Analysis Statistics

#	Node	Rows Actual	Loops
1.	→ Hash Semi Join (rows=500 loops=1) Hash Cond: (s.sid = r.sid)	500	1
2.	→ Index Scan using sailors_pkey on sailors as s (rows=19000 loops=1)	19000	1
3.	→ Index Only Scan using reserves_pkey on reserves as r (rows=35000 loops=1) Buckets: 32768 Batches: 1 Memory Usage: 274 kB	500	1
4.	→ Hash Semi Join (rows=500 loops=1) Hash Cond: (t.bid = b.bid)	500	1
5.	→ Hash (rows=1497 loops=1)	35000	1
6.	→ Bitmap Heap Scan on boat as b (rows=1497 loops=1) Recheck Cond: (color = 'red':bpchar) Heap Blocks: exact=0	1497	1
7.	→ Bitmap Index Scan on boat_colorbrin (rows=250 loops=1) Index Cond: (color = 'red':bpchar)	1497	1
8.	→ Bitmap Index Scan using colorbrin (rows=250 loops=1) Index Cond: (color = 'red':bpchar)	250	1

Total rows: 1 of 1 Query complete 00:00:00.061 Ln 12, Col 59

BRIN index here didn't optimize here as the search on boat.color was an exact value and the search BRIN is suitable for small range queries in a table with a very large range

Query 8 with hash index on boat(color) and B+tree index on sailors(sid)
Flags["seqscan = off", "hashjoin=on", "mergejoin = on", sailors_pkey and reserves_pkey indisvalid = false else are true]

Planning and Execution time

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Servers (1) PostgreSQL 14 Databases (5) postgres schema1 schema2 schema3

Query History

Data output Messages Explain Notifications

QUERY PLAN text

```

1 Hash Semi Join (cost=1544.09..2390.30 rows=11985 width=21) (actual time=7.541..9.043 rows=500 loops=1)
2 Hash Cond: (s.sid = r.sid)
3 → Index Scan using sailortree on sailors s (cost=0.29..663.29 rows=19000 width=25) (actual time=0.010..2.138 rows=19000 loops=1)
4 → Hash (cost=1325.49..1325.49 rows=17465 width=4) (actual time=5.394..5.397 rows=500 loops=1)
5 Buckets: 32768 Batches: 1 Memory Usage: 274kB
6 → Hash Semi Join (cost=122.32..1325.49 rows=17465 width=4) (actual time=5.021..5.319 rows=500 loops=1)
7 Hash Cond: (bid = b.bid)
8 → Index Only Scan using reserves_pkey on reserves r (cost=0.29..917.29 rows=35000 width=8) (actual time=0.016..2.876 rows=35000 loops=1)
9 Heap Fetches: 0
10 → Hash (cost=103.31..103.31 rows=1497 width=4) (actual time=0.485..0.486 rows=1497 loops=1)
11 Buckets: 2048 Batches: 1 Memory Usage: 69kB
12 → Bitmap Heap Scan on boat b (cost=59.60..103.31 rows=1497 width=4) (actual time=0.067..0.295 rows=1497 loops=1)
13 Recheck Cond: (color = 'red':bpchar)
14 Heap Blocks: exact=13
15 → Bitmap Index Scan on colorhash (cost=0.00..59.23 rows=1497 width=0) (actual time=0.058..0.058 rows=1497 loops=1)
16 Index Cond: (color = 'red':bpchar)
17 Planning Time: 0.313 ms
18 Execution Time: 9.176 ms

```

Total rows: 18 of 18 Query complete 00:00:00.050 Ln 13, Col 59

Execution Plan and Costs

The screenshot shows the pgAdmin 4 interface with the Explain tab selected. The query being explained is:

```

1 update pg_index set indisvalid = false where indexrelid = 'boat_pkey'::regclass
2 set enable_seascan = off;
  
```

The execution plan details the following steps:

#	Node	Rows	Loops
1.	→ Hash Semi Join (rows=500 loops=1) Hash Cond: (s.sid = r.sid)	500	1
2.	→ Index Scan using sailortree on sailors as s (rows=19000 loops=1)	19000	1
3.	→ Hash (rows=500 loops=1) Buckets: 32768 Batches: 1 Memory Usage: 274 kB	500	1
4.	→ Hash Semi Join (rows=500 loops=1) Hash Cond: (r.bid = b.bid)	500	1
5.	→ Index Only Scan using reserves_pkey on reserves as r (rows=35000 loops=1)	35000	1
6.	→ Hash (rows=1497 loops=1) Buckets: 2048 Batches: 1 Memory Usage: 69 kB	1497	1
7.	→ Bitmap Heap Scan on boat as b (rows=1497 loops=1) Recheck Cond: (color = 'red'::bpchar) Heap Blocks: exact=13	1497	1
8.	→ Bitmap Index Scan using colorhash (rows=1497 loops=1) Index Cond: (color = 'red'::bpchar)	1497	1

Total rows: 1 of 1 Query complete 00:00:00.093 Ln 13, Col 59

Since the search on b.color= 'red' is an exact value and search on s.sid is an exact value having a hash index(search time = O(1)) on boat.color and B+tree index(search time = O(logn)) on sailors.sid improved query performance

Optimised Query 8 SQL:

```

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

Reason: This query is optimised as it uses exists instead of in which reduces search time as SQL engine will terminate scanning process once a match is found but in the original query with "in" all records from inner query will be fetched

Optimised Query 8 with no index Flags[“seqscan = on”, “hashjoin=on”, “mergejoin = on”, all primary key indisvalid = true]]

Planning and Execution time

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

- Servers:** PostgreSQL 14 (selected)
- Databases:** postgres, schema1, schema2, schema3 (selected)
- Schema:** schema3
- Query:**

```
1 update pg_index set indisvalid = false where indexrelid = 'sailors_pkey'::regclass;
2 update pg_index set indisvalid = false where indexrelid = 'boat_pkey'::regclass;
3 update pg_index set indisvalid = false where indexrelid = 'reserves_pkey'::regclass;
```
- Explain Plan:**

```
text
1 Hash Semi Join (cost=1125.70..1657.91 rows=11985 width=21) (actual time=8.008..9.289 rows=500 loops=1)
  2 Hash Cond: (s.sid = r.sid)
    3 -> Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.027..1.258 rows=19000 loops=1)
    4 -> Hash (cost=907.39..907.39 rows=17465 width=4) (actual time=6.082..6.084 rows=500 loops=1)
      5 Buckets: 32768 Batches: 1 Memory Usage: 274kB
    6 -> Hash Semi Join (cost=81.21..907.39 rows=17465 width=4) (actual time=5.632..5.993 rows=500 loops=1)
    7 Hash Cond: (r.bid = b.bid)
      8 -> Seq Scan on reserves r (cost=0.00..540.00 rows=35000 width=8) (actual time=0.013..2.230 rows=35000 loops=1)
      9 -> Hash (cost=62.50..62.50 rows=1497 width=4) (actual time=0.609..0.610 rows=1497 loops=1)
      10 Buckets: 2048 Batches: 1 Memory Usage: 69kB
      11 -> Seq Scan on boat b (cost=0.00..62.50 rows=1497 width=4) (actual time=0.013..0.416 rows=1497 loops=1)
      12 Filter: (color = 'red'::bpchar)
      13 Rows Removed by Filter: 1503
    14 Planning Time: 0.222 ms
    15 Execution Time: 9.408 ms
```
- Statistics:**

Total rows: 15 of 15 Query complete 00:00:00.106 Ln 14, Col 92

Execution Plan and Costs

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

- Servers:** PostgreSQL 14 (selected)
- Databases:** postgres, schema1, schema2, schema3 (selected)
- Schema:** schema3
- Query:**

```
1 update pg_index set indisvalid = true where indexrelid = 'sailors_pkey'::regclass;
2 update pg_index set indisvalid = true where indexrelid = 'boat_pkey'::regclass;
3 update pg_index set indisvalid = true where indexrelid = 'reserves_pkey'::regclass;
```
- Explain Plan:**

#	Node	Rows	Loops
1.	-> Hash Semi Join (rows=500 loops=1) Hash Cond: (s.sid = r.sid)	500	1
2.	-> Seq Scan on sailors as s (rows=19000 loops=1)	19000	1
3.	-> Hash (rows=500 loops=1) Buckets: 32768 Batches: 1 Memory Usage: 274 kB	500	1
4.	-> Hash Inner Join (rows=500 loops=1) Hash Cond: (r.bid = b.bid)	500	1
5.	-> Seq Scan on reserves as r (rows=35000 loops=1)	35000	1
6.	-> Hash (rows=1497 loops=1) Buckets: 2048 Batches: 1 Memory Usage: 69 kB	1497	1
7.	-> Seq Scan on boat as b (rows=1497 loops=1) Filter: (color = 'red'::bpchar) Rows Removed by Filter: 1503	1497	1
- Statistics:**

Total rows: 1 of 1 Query complete 00:00:00.145 ✓ Successfully run. Total query runtime: 145 msec. 1 rows affected. [2]

Optimised Query 8 with B+tree on boat(color) Flags["seqscan = off", "hashjoin=on", "mergejoin = on", boat_pkey indisvalid = false else are true]

Planning and Execution time

```

pgAdmin 4
PgAdmin File Object Tools Help
Servers (1)
PostgreSQL 14
Databases (5)
postgres
schema1
schema2
schema3
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrapper
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configuration
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized View
Operators
Procedures
Sequences
Tables (3)

Query History
Query Data output Messages Explain Notifications
QUERY PLAN
text
1 Hash Semi Join (cost=1508.37..2354.58 rows=11985 width=21) (actual time=6.920..8.378 rows=500 loops=1)
2 Hash Cond: (s.sid = r.sid)
3 -> Index Scan using sailors_pkey on sailors s (cost=0.29..663.29 rows=19000 width=25) (actual time=0.006..2.126 rows=19000 loops=1)
4 -> Hash (cost=1289.77..1289.77 rows=17465 width=4) (actual time=4.862..4.864 rows=500 loops=1)
5 Buckets: 32768 Batches: 1 Memory Usage: 274kB
6 -> Hash Semi Join (cost=86.60..1289.77 rows=17465 width=8) (actual time=4.513..4.803 rows=500 loops=1)
7 Hash Cond: (r.bid = b.bid)
8 -> Index Only Scan using reserves_pkey on reserves r (cost=0.29..917.29 rows=35000 width=8) (actual time=0.006..2.217 rows=35000 loops=1)
9 Heap Fetches: 0
10 -> Hash (cost=67.59..67.59 rows=1497 width=4) (actual time=0.312..0.314 rows=1497 loops=1)
11 Buckets: 2048 Batches: 1 Memory Usage: 69kB
12 -> Bitmap Heap Scan on boat b (cost=23.88..67.59 rows=1497 width=4) (actual time=0.047..0.175 rows=1497 loops=1)
13 Recheck Cond: (color = 'red'::bpchar)
14 Heap Blocks: exact=13
15 -> Bitmap Index Scan on colortree (cost=0.00..23.51 rows=1497 width=0) (actual time=0.040..0.040 rows=1497 loops=1)
16 Index Cond: (color = 'red'::bpchar)
17 Planning Time: 0.279 ms
18 Execution Time: 8.496 ms

Total rows: 18 of 18 Query complete 00:00:00.046
Ln 14, Col 92

```

Execution Plan and Costs

#	Node	Rows Actual	Loops
1.	-> Hash Semi Join (rows=500 loops=1) Hash Cond: (s.sid = r.sid)	500	1
2.	-> Index Scan using sailors_pkey on sailors s (rows=19000 loops=1)	19000	1
3.	-> Hash (rows=500 loops=1) Buckets: 32768 Batches: 1 Memory Usage: 274 kB	500	1
4.	-> Hash Semi Join (rows=500 loops=1) Hash Cond: (r.bid = b.bid)	500	1
5.	-> Index Only Scan using reserves_pkey on reserves r (rows=35000 loops=1)	35000	1
6.	-> Hash (rows=1497 loops=1) Buckets: 2048 Batches: 1 Memory Usage: 69 kB	1497	1
7.	-> Bitmap Heap Scan on boat b (rows=1497 loops=1) Recheck Cond: (color = 'red'::bpchar) Heap Blocks: exact=13	1497	1
8.	-> Bitmap Index Scan using colortree (rows=1497 loops=1) Index Cond: (color = 'red'::bpchar)	1497	1

```

pgAdmin 4
PgAdmin File Object Tools Help
Servers (1)
PostgreSQL 14
Databases (5)
postgres
schema1
schema2
schema3
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrapper
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configuration
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized View
Operators
Procedures
Sequences
Tables (3)

Query History
Query Data output Messages Explain Notifications
Graphical Analysis Statistics
# Node Rows Actual Loops
1. -> Hash Semi Join (rows=500 loops=1)  
Hash Cond: (s.sid = r.sid) 500 1
2. -> Index Scan using sailors_pkey on sailors s (rows=19000 loops=1) 19000 1
3. -> Hash (rows=500 loops=1)  
Buckets: 32768 Batches: 1 Memory Usage: 274 kB 500 1
4. -> Hash Semi Join (rows=500 loops=1)  
Hash Cond: (r.bid = b.bid) 500 1
5. -> Index Only Scan using reserves_pkey on reserves r (rows=35000 loops=1) 35000 1
6. -> Hash (rows=1497 loops=1)  
Buckets: 2048 Batches: 1 Memory Usage: 69 kB 1497 1
7. -> Bitmap Heap Scan on boat b (rows=1497 loops=1)  
Recheck Cond: (color = 'red'::bpchar)  
Heap Blocks: exact=13 1497 1
8. -> Bitmap Index Scan using colortree (rows=1497 loops=1)  
Index Cond: (color = 'red'::bpchar) 1497 1

Total rows: 1 of 1 Query complete 00:00:00.058
Ln 14, Col 92

```

B+tree index optimised here as it was used to search(in O(logn)) in table boat for every boat.color = 'red' in the subquery

Optimised Query 8 with Hash index on boat(color) Flags["seqscan = off", "hashjoin=on", "mergejoin = on", boat_pkey indisvalid = false else are true]

Planning and Execution time

```

update pg_index set indisvalid = true where indexrelid = 'sailors_pkey'::regclass;
No limit
Data output Messages Explain Notifications
+-----+
| QUERY PLAN |
+-----+
text
1 Hash Semi Join (cost=1544.09..2390.30 rows=11985 width=21) (actual time=6.996..8.438 rows=500 loops=1)
2 Hash Cond: (s.sid = r.sid)
3 -> Index Scan using sailors_pkey on sailors s (cost=0.29..663.29 rows=19000 width=25) (actual time=0.006..2.087 rows=19000 loops=1)
4 -> Hash (cost=1325.49..1325.49 rows=17465 width=4) (actual time=4.994..4.996 rows=500 loops=1)
5 Buckets: 32768 Batches: 1 Memory Usage: 274kB
6 -> Hash Semi Join (cost=122.32..1325.49 rows=17465 width=4) (actual time=4.646..4.938 rows=500 loops=1)
7 Hash Cond: (r.bid = b.bid)
8 -> Index Only Scan using reserves_pkey on reserves r (cost=0.29..917.29 rows=35000 width=8) (actual time=0.006..2.713 rows=35000 loops=1)
9 Heap Fetches: 0
10 -> Hash (cost=103.31..103.31 rows=1497 width=4) (actual time=0.395..0.396 rows=1497 loops=1)
11 Buckets: 2048 Batches: 1 Memory Usage: 69kB
12 -> Bitmap Heap Scan on boat b (cost=59.60..103.31 rows=1497 width=4) (actual time=0.056..0.255 rows=1497 loops=1)
13 Recheck Cond: (color = 'red'::bpchar)
14 Heap Blocks: exact=13
15 -> Bitmap Index Scan on colorhash (cost=0.00..59.23 rows=1497 width=0) (actual time=0.048..0.048 rows=1497 loops=1)
16 Index Cond: (color = 'red'::bpchar)
17 Planning Time: 0.260 ms
18 Execution Time: 8.484 ms
Total rows: 18 of 18 Query complete 00:00:00.070
Ln 14, Col 92

```

Execution Plan and Costs

```

update pg_index set indisvalid = false where indexrelid = 'boat_pkey'::regclass;
No limit
Data output Messages Explain Notifications
+-----+
| GRAPHICAL | ANALYSIS | STATISTICS |
+-----+
# Node
1. -> Hash Semi Join (rows=500 loops=1)
   Hash Cond: (s.sid = r.sid)
2. -> Index Scan using sailors_pkey on sailors s (rows=19000 loops=1)
3. -> Hash (rows=500 loops=1)
   Buckets: 32768 Batches: 1 Memory Usage: 274 kB
4. -> Hash Semi Join (rows=500 loops=1)
   Hash Cond: (r.bid = b.bid)
5. -> Index Only Scan using reserves_pkey on reserves r (rows=35000 loops=1)
6. -> Hash (rows=1497 loops=1)
   Buckets: 2048 Batches: 1 Memory Usage: 69 kB
7. -> Bitmap Heap Scan on boat b (rows=1497 loops=1)
   Recheck Cond: (color = 'red'::bpchar)
   Heap Blocks: exact=13
8. -> Bitmap Index Scan using colorhash (rows=1497 loops=1)
   Index Cond: (color = 'red'::bpchar)
Total rows: 1 of 1 Query complete 00:00:00.067
Ln 14, Col 92

```

Hash index optimised here as it was used to search(in O(1)) in table boat for every boat.color = 'red' in the subquery

Optimised Query 8 with BRIN on boat(color) Flags[“seqscan = off”, “hashjoin=on”, “mergejoin = on”, boat_pkey indisvalid = false else are true])

Planning and Execution time

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

- Left Panel (Servers):** Shows a tree view of the database structure:
 - Servers (1)
 - PostgreSQL 14
 - Databases (5)
 - postgres
 - schema1
 - schema2
 - schema3
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrapper
 - Languages
 - Publications
 - Schemas (1)
 - public
 - Aggregates
 - Collations
 - Domains
 - FTS Configuration
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (3)
- Central Panel (Query Plan):** A detailed list of the execution plan steps. The first step is a Hash Semi Join. Subsequent steps include Hash Cond, Index Scan, Hash, Hash Semi Join, Hash Cond, Index Only Scan, Hash, Hash, Hash, Heap Fetches, Hash, Hash Cond, Bitmap Heap Scan, Recheck Cond, Rows Removed by Index Recheck, Heap Blocks, Bitmap Index Scan, Index Cond, Planning Time, and Execution Time.
- Bottom Status Bar:** Displays "Total rows: 19 of 19" and "Query complete 00:00:00.117".
- Bottom Right Corner:** Shows "Ln 14, Col 92".

Execution plan and Costs

pgAdmin 4

Servers (1) schema3/postgres@PostgreSQL 14*

File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents

Databases (5)

- postgres
- schema1
- schema2
- schema3**
- Casts
- Catalogs
- Event Triggers
- Extensions
- Foreign Data Wrapper
- Languages
- Publications
- Schemas (1)
 - public**
 - Aggregates
 - Collations
 - Domains
 - FTS Configuratio
 - FTS Dictionaries
 - A FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized View
 - Operators
 - Procedures
 - Sequences
 - Tables (3)

Query History

```
1 update pg_index set indisvalid = true where indexrelid = 'sailors_pkey':>regclass;
2 update pg_index set indisvalid = false where indexrelid = 'boat_pkey':>regclass;
3 update pg_index set indisvalid = true where indexrelid = 'reserves_pkey':>regclass;
```

Scratch Pad

Query Explain Notifications

#	Node	Rows	Loops
		Actual	
1.	→ Hash Semi Join (rows=500 loops=1) Hash Cond: (s.sid = r.sid)	500	1
2.	→ Index Scan using sailors_pkey on sailors as s (rows=19000 loops=1)	19000	1
3.	→ Hash (rows=500 loops=1) Buckets: 32768 Batches: 1 Memory Usage: 274 kB	500	1
4.	→ Hash Semi Join (rows=500 loops=1) Hash Cond: (r.bid = b.bid)	500	1
5.	→ Index Only Scan using reserves_pkey on reserves as r (rows=35000 loops=1)	35000	1
6.	→ Hash (rows=1497 loops=1) Buckets: 2048 Batches: 1 Memory Usage: 69 kB	1497	1
7.	→ Bitmap Heap Scan on boat as b (rows=1497 loops=1) Recheck Cond: (color = 'red'::bpchar) Heap Blocks: exact=0	1497	1
8.	→ Bitmap Index Scan using colorbrin on boat (rows=250 loops=1) Index Cond: (color = 'red'::bpchar)	250	1

Total rows: 1 of 1 Query complete 00:00:00.053 Ln 14, Col 92

BRIN index here didn't optimize here as the search on boat.color was an exact value and the search BRIN is suitable for small range queries in a table with a very large range

Optimised Query 8 with B+ tree index on sailor(sid) and hash index on boat(color) Flags[“seqscan = off”, “hashjoin=on”, “mergejoin = on”, sailors_pkey and boat_pkey indisvalid = false else are true]

Planning and Execution time

The screenshot shows the pgAdmin 4 interface with the 'Explain' tab selected. The query plan text is as follows:

```

QUERY PLAN
text
3  group key: r.sid
4  Batches: 1 Memory Usage: 433kB
5  -> Hash Semi Join (cost=122.32..1325.49 rows=17465 width=4) (actual time=5.877..6.173 rows=500 loops=1)
6  Hash Cond: (r.bid = b.bid)
7  -> Index Only Scan using reserves_pkey on reserves r (cost=0.29..917.29 rows=35000 width=8) (actual time=0.017..3.357 rows=35000 loops=1)
8  Heap Fetches: 0
9  -> Hash (cost=103.31..103.31 rows=1497 width=4) (actual time=0.441..0.442 rows=1497 loops=1)
10 Bucket: 2048 Batches: 1 Memory Usage: 69kB
11 -> Bitmap Heap Scan on boat b (cost=59.60..103.31 rows=1497 width=4) (actual time=0.065..0.280 rows=1497 loops=1)
12 Recheck Cond: (color = 'red':bpchar)
13 Heap Blocks: exact=13
14 -> Bitmap Index Scan on colorhash (cost=0.00..59.23 rows=1497 width=0) (actual time=0.055..0.055 rows=1497 loops=1)
15 Index Cond: (color = 'red':bpchar)
16 -> Memoize (cost=0.01..0.10 rows=1 width=25) (actual time=0.001..0.001 rows=1 loops=500)
17 Cache Key: r.sid
18 Cache Mode: logical
19 Hits: 0 Misses: 500 Evictions: 0 Overflows: 0 Memory Usage: 63kB
20 -> Index Scan using sidtree on sailors s (cost=0.00..0.09 rows=1 width=25) (actual time=0.001..0.001 rows=1 loops=500)
21 Index Cond: (sid = r.sid)
22 Planning Time: 0.276 ms
23 Execution Time: 7.627 ms
Total rows: 23 of 23  Query complete 00:00:00.045

```

Execution Plan and Costs

#	Node	Rows Actual	Loops
1.	-> Nested Loop Inner Join (rows=500 loops=1)	500	1
2.	-> Aggregate (rows=500 loops=1) Buckets: Batches: Memory Usage: 433 kB	500	1
3.	-> Hash Semi Join (rows=500 loops=1) Hash Cond: (r.bid = b.bid)	500	1
4.	-> Index Only Scan using reserves_pkey on reserves r (rows=35000 loops=1)	35000	1
5.	-> Hash (rows=1497 loops=1) Buckets: 2048 Batches: 1 Memory Usage: 69 kB	1497	1
6.	-> Bitmap Heap Scan on boat b (rows=1497 loops=1) Recheck Cond: (color = 'red':bpchar) Heap Blocks: exact=13	1497	1
7.	-> Bitmap Index Scan using colorhash (rows=1497 loops=1) Index Cond: (color = 'red':bpchar)	1497	1
8.	-> Memoize (rows=1 loops=500) Buckets: Batches: Memory Usage: 63 kB	1	500
9.	-> Index Scan using sidtree on sailors s (rows=1 loops=500) Index Cond: (sid = r.sid)	1	500

Total rows: 1 of 1 Query complete 00:00:00.087 Ln 15, Col 92

Since the search on b.color= 'red' is an exact value and search on s.sid is an exact value having a hash index(search time = O(1)) on boat.color and B+tree index(search time = O(logn)) on sailors.sid improved query performance

Query 9 with no index Flags[“seqscan = on”, “hashjoin=on”, “mergejoin = on”, all primary key indisvalid = false else are true]

Planning and Execution time

The screenshot shows the pgAdmin 4 interface with the Explain tab selected. The query plan details the execution steps:

- Rows removed by filter: 1003
- Hash (cost=349.00..349.00 rows=19000 width=25) (actual time=11.771..11.772 rows=19000 loops=1)
- Buckets: 32768 Batches: 1 Memory Usage: 1370kB
- Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.031..5.065 rows=19000 loops=1)
- Hash (cost=1345.71..1345.71 rows=17500 width=8) (actual time=16.321..16.323 rows=500 loops=1)
- Buckets: 32768 Batches: 1 Memory Usage: 276kB
- Hash Join (cost=667.75..1345.71 rows=17500 width=8) (actual time=15.847..16.229 rows=500 loops=1)
- Hash Cond: (r2.sid = s2.sid)
- Hash Join (cost=81.25..713.26 rows=17500 width=4) (actual time=11.818..12.119 rows=500 loops=1)
- Hash Cond: (r2.bid = b2.bid)
- Seq Scan on reserves r2 (cost=0.00..540.00 rows=35000 width=8) (actual time=0.031..4.243 rows=35000 loops=1)
- Hash (cost=62.50..62.50 rows=1500 width=4) (actual time=1.450..1.450 rows=1500 loops=1)
- Buckets: 2048 Batches: 1 Memory Usage: 69kB
- Seq Scan on boat b2 (cost=0.00..62.50 rows=1500 width=4) (actual time=0.469..1.041 rows=1500 loops=1)
- Filter: (color = 'green':bpchar)
- Rows Removed by Filter: 1500
- Hash (cost=349.00..349.00 rows=19000 width=4) (actual time=3.895..3.896 rows=19000 loops=1)
- Buckets: 32768 Batches: 1 Memory Usage: 924kB
- Seq Scan on sailors s2 (cost=0.00..349.00 rows=19000 width=4) (actual time=0.023..1.458 rows=19000 loops=1)
- Planning Time: 1.453 ms
- Execution Time: 35.974 ms

Total rows: 32 of 32 Query complete 00:00:00.092

Successfully run. Total query runtime: 92 msec. 32 rows affected.

Execution Plan and Costs

The screenshot shows the pgAdmin 4 interface with the Analysis tab selected. The execution plan details the cost and resource usage for each step:

Step	Operation	Rows	Loops
1.	→ Hash Semi Join (rows=500 loops=1) Hash Cond: (s.sid = s2.sid)	500	1
2.	→ Hash Inner Join (rows=500 loops=1) Hash Cond: (r.sid = s.sid)	500	1
3.	→ Hash Inner Join (rows=500 loops=1) Hash Cond: (r.bid = b.bid)	500	1
4.	→ Seq Scan on reserves as r (rows=35000 loops=1)	35000	1
5.	→ Hash (rows=1497 loops=1) Buckets: 2048 Batches: 1 Memory Usage: 69 kB	1497	1
6.	→ Seq Scan on boat as b (rows=1497 loops=1) Filter: (color = 'red':bpchar) Rows Removed by Filter: 1503	1497	1
7.	→ Hash (rows=19000 loops=1) Buckets: 32768 Batches: 1 Memory Usage: 1370 kB	19000	1
8.	→ Seq Scan on sailors as s (rows=19000 loops=1)	19000	1
9.	→ Hash (rows=500 loops=1) Buckets: 32768 Batches: 1 Memory Usage: 276 kB	500	1
10.	→ Hash Inner Join (rows=500 loops=1) Hash Cond: (r2.sid = s2.sid)	500	1
11.	→ Hash Inner Join (rows=500 loops=1) Hash Cond: (r2.bid = b2.bid)	500	1
12.	→ Seq Scan on reserves as r2 (rows=35000 loops=1)	35000	1

Total rows: 1 of 1 Query complete 00:00:00.086

Ln 13, Col 140

Query 9 with B+tree index on sailors(sid) and boat(color) Flags[“seqscan = off”, “hashjoin=on”, “mergejoin = on”, sailors_pkey and boat_pkey indisvalid = false else are true]

Planning and Execution time

pgAdmin 4

File Object Tools Help

Browser schema3/postgres@PostgreSQL 14*

Extensions Foreign Data Wrapper Languages Publications Schemas (1) public Aggregates Collations Domains FTS Configuration FTS Dictionaries FTS Parsers FTS Templates Foreign Tables Functions Materialized Views Operators Procedures Sequences Tables (3) boat reserves sailors Trigger Function Types Views Subscriptions schema4 Login/Group Roles Tablespaces

Dashboard Properties SQL Statistics Dependencies Dependents

Data output Messages Notifications

QUERY PLAN

text

15 -> Bitmap Index Scan on colorbtree (cost=0.00..2.00 rows=1497 width=100) (actual time=0.00..0.00 rows=1497 loops=1)

16 Index Cond: (color = 'red'-bpchar)

17 -> Hash (cost=2094.15..2094.15 rows=17500 width=8) (actual time=10.06..10.067 rows=500 loops=1)

18 Buckets: 32768 Batches: 1 Memory Usage: 276 kB

19 -> Hash Join (cost=86.98..2094.15 rows=17500 width=8) (actual time=9.340..9.977 rows=500 loops=1)

20 Hash Cond: (r2.bid = b2.bid)

21 -> Merge Join (cost=0.58..1701.50 rows=35000 width=12) (actual time=0.037..7.796 rows=35000 loops=1)

22 Merge Cond: (s2.sid = r2.sid)

23 -> Index Only Scan using sidbtree on sailors s2 (cost=0.29..501.29 rows=19000 width=4) (actual time=0.022..0.945 rows=12001 loops=1)

24 Heap Fetches: 0

25 -> Index Only Scan using reserves_pkey on reserves r2 (cost=0.29..917.29 rows=35000 width=8) (actual time=0.014..2.947 rows=35000 loops=1)

26 Heap Fetches: 0

27 -> Hash (cost=67.66..67.66 rows=1500 width=4) (actual time=0.342..0.343 rows=1500 loops=1)

28 Buckets: 2048 Batches: 1 Memory Usage: 69 kB

29 -> Bitmap Heap Scan on boat b2 (cost=23.91..67.66 rows=1500 width=4) (actual time=0.053..0.184 rows=1500 loops=1)

30 Recheck Cond: (color = 'green'-bpchar)

31 Heap Blocks: exact+13

32 -> Bitmap Index Scan on colorbtree (cost=0.00..23.53 rows=1500 width=0) (actual time=0.046..0.046 rows=1500 loops=1)

33 Index Cond: (color = 'green'-bpchar)

34 Planning Time: 1.582 ms

35 Execution Time: 25.510 ms

Total rows: 35 of 35 Query complete 00:00:00.060

Successfully run. Total query runtime: 60 msec. 35 rows affected.

Execution Plan and Costs

pgAdmin 4

File Object Tools Help

Browser schema3/postgres@PostgreSQL 14*

Extensions Foreign Data Wrapper Languages Publications Schemas (1) public Aggregates Collations Domains FTS Configuration FTS Dictionaries FTS Parsers FTS Templates Foreign Tables Functions Materialized Views Operators Procedures Sequences Tables (3) boat reserves sailors Trigger Function Types Views Subscriptions schema4 Login/Group Roles Tablespaces

Dashboard Properties SQL Statistics Dependencies Dependents

Data output Messages Explain Notifications

Graphical Analysis Statistics

1. -> Hash Semi Join (rows=500 loops=1) Hash Cond: (s.sid = s2.sid)	500	1
2. -> Hash Inner Join (rows=500 loops=1) Hash Cond: (r.bid = b.bid)	500	1
3. -> Merge Inner Join (rows=35000 loops=1)	35000	1
4. -> Index Scan using sidbtree on sailors as s (rows=12001 loops=1)	12001	1
5. -> Index Only Scan using reserves_pkey on reserves as r (rows=35000 loops=1)	35000	1
6. -> Hash (rows=1497 loops=1) Buckets: 2048 Batches: 1 Memory Usage: 69 kB	1497	1
7. -> Bitmap Heap Scan on boat as b (rows=1497 loops=1) Recheck Cond: (color = 'red'-bpchar) Heap Blocks: exact+13	1497	1
8. -> Bitmap Index Scan using colorbtree (rows=1497 loops=1) Index Cond: (color = 'red'-bpchar)	1497	1
9. -> Hash (rows=500 loops=1) Buckets: 32768 Batches: 1 Memory Usage: 276 kB	500	1
10. -> Hash Inner Join (rows=500 loops=1) Hash Cond: (r2.bid = b2.bid)	500	1
11. -> Merge Inner Join (rows=35000 loops=1)	35000	1
12. -> Index Only Scan using sidbtree on sailors as s2 (rows=12001 loops=1)	12001	1
13. -> Index Only Scan using reserves_pkey on reserves as r2 (rows=35000 loops=1)	35000	1

Total rows: 1 of 1 Query complete 00:00:00.098

Ln 14, Col 140

B+tree index optimised here as it was used to search(in O(logn)) in table sailors for every reserves.sid=sailors.sid and was used for boat.color = ‘red’ and boat.color = ‘green’

Query 9 with hash index on boat(color) Flags["seqscan = off", "hashjoin=on", "mergejoin = on", boat_pkey indisvalid = false else are true]

Planning and Execution time

The screenshot shows the pgAdmin 4 interface with the Explain tab selected. The query plan is as follows:

```

QUERY PLAN
text
r -> Memosize (cost=0.29..0.31 rows=1 width=4) (actual time=0.000..0.000 rows=1 loops=1)
  14 Cache Key: r.bid
  15 Cache Mode: logical
  16 Hits: 34995 Misses: 5 Evictions: 0 Overflows: 0 Memory Usage: 1kB
  17 -> Index Scan using boat_pkey on boat b2 (cost=0.28..0.30 rows=1 width=4) (actual time=0.006..0.006 rows=0 loops=5)
  18 Index Cond: (bid = r.bid)
  19 Filter: (color = 'green':bpchar)
  20 Rows Removed by Filter: 1
  21 -> Index Only Scan using sailors_pkey on sailors s2 (cost=0.29..501.29 rows=19000 width=4) (actual time=0.024..1.174 rows=11500 loops=1)
  22 Heap Fetches: 0
  23 -> Index Only Scan using reserves_pkey on reserves r (cost=0.29..917.29 rows=35000 width=8) (actual time=0.027..3.589 rows=34001 loops=1)
  24 Heap Fetches: 0
  25 -> Hash (cost=103.31..103.31 rows=1497 width=4) (actual time=0.671..0.672 rows=1497 loops=1)
  26 Buckets: 2048 Batches: 1 Memory Usage: 69kB
  27 -> Bitmap Heap Scan on boat b (cost=59.60..103.31 rows=1497 width=4) (actual time=0.083..0.428 rows=1497 loops=1)
  28 Recheck Cond: (color = 'red':bpchar)
  29 Heap Blocks: exact=13
  30 -> Bitmap Index Scan on colorhash (cost=0.00..59.23 rows=1497 width=0) (actual time=0.069..0.070 rows=1497 loops=1)
  31 Index Cond: (color = 'red':bpchar)
  32 Planning Time: 1.356 ms
  33 Execution Time: 25.285 ms
Total rows: 33 of 33  Query complete 00:00:00.069
Ln 14, Col 140

```

Execution Plan and Costs

The screenshot shows the pgAdmin 4 interface with the Explain tab selected. The graphical execution plan is as follows:

Step	Operation	Cost	Rows
1.	→ Hash Inner Join (rows=500 loops=1) Hash Cond: (r.bid = b.bid)	500	1
2.	→ Merge Inner Join (rows=1000 loops=1)	1000	1
3.	→ Merge Semi Join (rows=500 loops=1)	500	1
4.	→ Index Scan using sailors_pkey on sailors as s (rows=11501 loops=1)	11501	1
5.	→ Merge Inner Join (rows=500 loops=1)	500	1
6.	→ Nested Loop Inner Join (rows=500 loops=1)	500	1
7.	→ Index Only Scan using reserves_pkey on reserves as r (rows=35000 loops=1)	35000	1
8.	→ Memoize (rows=0 loops=35000) Buckets: Batches: Memory Usage: 1 kB	0	35000
9.	→ Index Scan using boat_pkey on boat as b2 (rows=0 loops=5) Filter: (color = 'green':bpchar) Index Cond: (bid = r.bid) Rows Removed by Filter: 1	0	5
10.	→ Index Only Scan using sailors_pkey on sailors as s2 (rows=11500 loops=1)	11500	1
11.	→ Index Only Scan using reserves_pkey on reserves as r (rows=34001 loops=1)	34001	1
12.	→ Hash (rows=1497 loops=1) Buckets: 2048 Batches: 1 Memory Usage: 69 kB → Bitmap Heap Scan on boat as b (rows=1497 loops=1) Recheck Cond: (color = 'red':bpchar) Heap Blocks: exact=13	1497	1
13.	→ Bitmap Heap Scan on boat as b (rows=1497 loops=1) Recheck Cond: (color = 'red':bpchar) Heap Blocks: exact=13	1497	1

Total rows: 1 of 1 Query complete 00:00:00.059

Hash index optimised here as it was used to search(in O(1)) for boat.color = 'red' and boat.color = 'green'

Query 9 with BRIN index on boat(Color) Flags["seqscan = off", "hashjoin=on", "mergejoin = on", boat_pkey indisvalid = false else are true]

Planning and Execution time

The screenshot shows the pgAdmin 4 interface with the Explain tab selected. The query being explained is:

```

text
  Values key: 1..1000
14 Cache Mode: logical
15 Hits: 34995 Misses: 5 Evictions: 0 Overflows: 0 Memory Usage: 1kB
16 > Index Scan using boat_pkey on boat b2 (cost=0.28..0.30 rows=1 width=4) (actual time=0.012..0.012 rows=0 loops=5)
17   Index Cond: (bid = r2.bid)
18   Filter: (color = 'green':bpchar)
19   Rows Removed by Filter: 1
20 > Index Only Scan using sailors_pkey on sailors s2 (cost=0.29..501.29 rows=19000 width=4) (actual time=0.022..1.183 rows=11500 loops=1)
21   Index Cond: (color = 'green':bpchar)
22   Heap Fetches: 0
23 > Index Only Scan using reserves_pkey on reserves r (cost=0.29..917.29 rows=35000 width=8) (actual time=0.038..4.224 rows=34001 loops=1)
24   Index Cond: (color = 'red':bpchar)
25   Heap Fetches: 0
26 > Hash (cost=74.91..74.91 rows=1497 width=4) (actual time=1.273..1.274 rows=1497 loops=1)
27   Buckets: 2048 Batches: 1 Memory Usage: 69kB
28   > Bitmap Heap Scan on boat b (cost=12.41..74.91 rows=1497 width=4) (actual time=0.093..0.908 rows=1497 loops=1)
29   Recheck Cond: (color = 'red':bpchar)
30   Rows Removed by Index Recheck: 1503
31   Heap Blocks: lossy+25
32   > Bitmap Index Scan on colorbin (cost=0.00..12.03 rows=3000 width=0) (actual time=0.079..0.079 rows=250 loops=1)
33   Index Cond: (color = 'red':bpchar)
34   Planning Time: 2.292 ms
35   Execution Time: 42.114 ms
  
```

Total rows: 34 of 34 Query complete 00:00:00.085 Successfully run. Total query runtime: 85 msec. 34 rows affected.

Execution Plan and Costs

The screenshot shows the pgAdmin 4 interface with the Analysis tab selected. The graphical execution plan for the query is as follows:

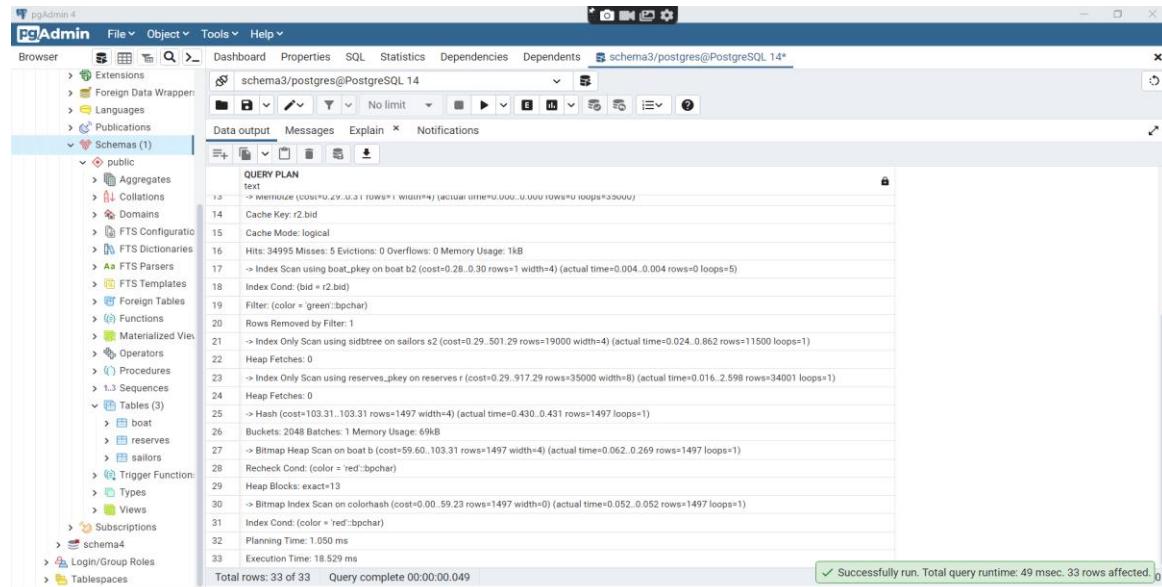
1. → Hash Inner Join (rows=500 loops=1)
Hash Cond: (r.bid = b.bid)
2. → Merge Inner Join (rows=1000 loops=1)
3. → Merge Semi Join (rows=500 loops=1)
4. → Index Scan using sailors_pkey on sailors as s (rows=11501 loops=1)
5. → Merge Inner Join (rows=500 loops=1)
6. → Nested Loop Inner Join (rows=500 loops=1)
7. → Index Only Scan using reserves_pkey on reserves as r2 (rows=35000 loops=1)
8. → Memoize (rows=0 loops=35000)
Buckets: Batches Memory Usage: 1 kB
9. → Index Scan using boat_pkey on boat as b2 (rows=0 loops=5)
Filter: (color = 'green':bpchar)
Index Cond: (bid = r2.bid)
Rows Removed by Filter: 1
10. → Index Only Scan using sailors_pkey on sailors as s2 (rows=11500 loops=1)
11. → Index Only Scan using reserves_pkey on reserves as r (rows=34001 loops=1)
12. → Hash (rows=1497 loops=1)
Buckets: 2048 Batches: 1 Memory Usage: 69 kB
13. → Bitmap Heap Scan on boat as b (rows=1497 loops=1)
Recheck Cond: (color = 'red':bpchar)
Heap Blocks: exact=0

Total rows: 1 of 1 Query complete 00:00:00.085 Ln 15, Col 140

BRIN index here didn't optimize here as the search on boat.color was an exact value and the search BRIN is suitable for small range queries in a table with a very large range

Query 9 with B+tree index on sailors(sid) and hash index on boat(color)
Flags[“seqscan = off”, “hashjoin=on”, “mergejoin = on”, sailors_pkey and boat_pkey indisvalid = false else are true]

Planning and Execution time

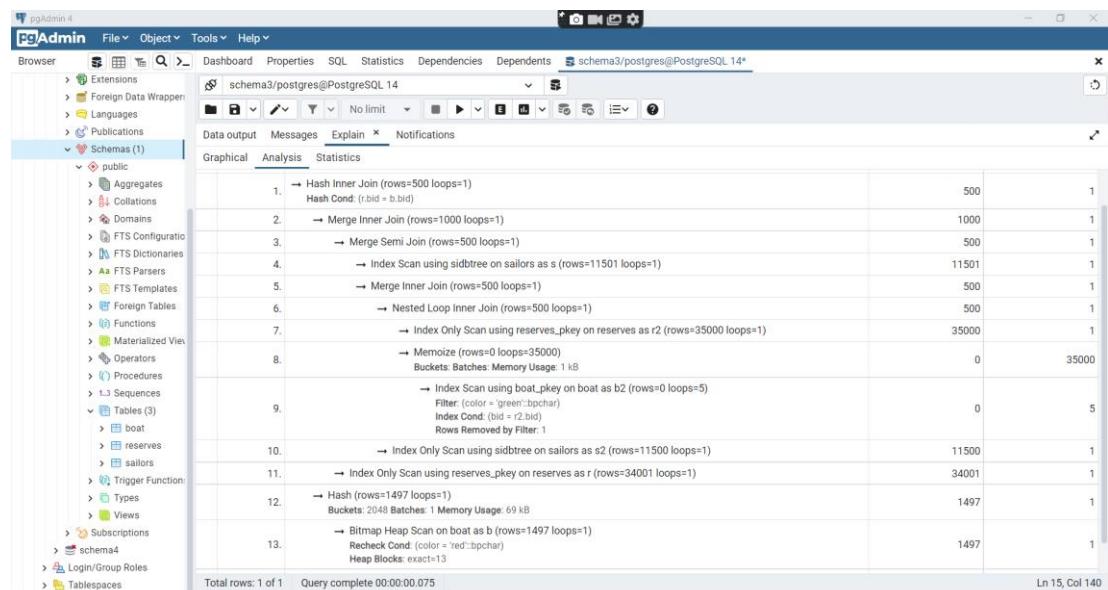


```

pgAdmin 4
File Object Tools Help
Browser Extensions Foreign Data Wrapper Languages Publications Schemas (1)
  public Aggregates Collations Domains FTS Configuration FTS Dictionaries FTS Parsers FTS Templates Foreign Tables Functions Materialized Views Operators Procedures Sequences Tables (3)
  boat reserves sailors Trigger Function Types Views Subscriptions schema4 Login/Group Roles Tablespaces
schema3/postgres@PostgreSQL 14*
Data output Messages Explain Notifications
QUERY PLAN
text
1. → Memtable (cost=0.00..0.01 rows=1 width=4) (actual time=0.000..0.000 rows=1 loops=1)
  Cache Key: r2.bid
  Cache Mode: logical
  Hits: 34995 Misses: 5 Executions: 0 Overflows: 0 Memory Usage: 1kB
  17. → Index Scan using boat_pkey on boat b2 (cost=0.28..0.30 rows=1 width=4) (actual time=0.004..0.004 rows=1 loops=5)
    Index Cond: (bid = r2.bid)
    Filter: (color = 'green'::bpchar)
  20. Rows Removed by Filter: 1
  21. → Index Only Scan using sidtree on sailors s2 (cost=0.29..501.29 rows=19000 width=4) (actual time=0.024..0.862 rows=11500 loops=1)
  22. Heap Fetches: 0
  23. → Index Only Scan using reserves_pkey on reserves r (cost=0.29..917.29 rows=35000 width=8) (actual time=0.016..2.598 rows=34001 loops=1)
  24. Heap Fetches: 0
  25. → Hash (cost=103.31..103.31 rows=1497 width=4) (actual time=0.430..0.431 rows=1497 loops=1)
  Buckets: 2048 Batches: 1 Memory Usage: 69kB
  27. → Bitmap Heap Scan on boat b (cost=59.60..103.31 rows=1497 width=4) (actual time=0.062..0.269 rows=1497 loops=1)
  28. Recheck Cond: (color = 'red'::bpchar)
  29. Heap Blocks: exact=13
  30. → Bitmap Index Scan on colorhash (cost=0.00..59.23 rows=1497 width=0) (actual time=0.052..0.052 rows=1497 loops=1)
  31. Index Cond: (color = 'red'::bpchar)
  32. Planning Time: 1.050 ms
  33. Execution Time: 18.529 ms
Total rows: 33 of 33  Query complete 00:00:00.049
Successfully run. Total query runtime: 49 msec. 33 rows affected.

```

Execution Plan and Costs



Since the search on b.color= ‘red’ and b.color = ‘green’ is an exact value and search on sailors.sid = reserves.sid is an exact value having a hash index(search time = O(1)) on boat.color and B+tree index(search time = O(logn)) on sailors.sid improved query performance

Optimised Query 9 SQL:

```
select s.sname  
from sailors s inner join reserves r on s.sid = r.sid inner join boat b on b.bid =  
r.bid  
where b.color = 'red' and exists ( select s2.sid  
from sailors s2 inner join reserves r2 on s2.sid = r2.sid  
inner join boat b2 on b2.bid=r2.bid  
where s.sid = s2.sid and b2.color = 'green');
```

Reason: This query is optimised as it uses inner join instead of cartesian product so less number of rows is propagated upwards in inner nodes of relational algebra tree also it uses exists instead of in which reduces search time as SQL engine will terminate scanning process once a match is found but in the original query with “in” all records from inner query will be fetched

Optimised Query 9 with no index Flags[“seqscan = on”, “hashjoin=on”, “mergejoin = on”, all primary key indisvalid = true])

Planning and Execution time

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Dashboard Properties SQL Statistics Dependencies Dependents schema3/postgres@PostgreSQL 14*

Data output Messages Explain Notifications

QUERY PLAN text

```

12  Rows Removed by Filter: 1503
13  -> Hash (cost=349.00..349.00 rows=19000 width=25) (actual time=5.040..5.040 rows=19000 loops=1)
14    Buckets: 32768 Batches: 1 Memory Usage: 1370kB
15    -> Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.022..1.862 rows=19000 loops=1)
16    -> Hash (cost=1345.71..1345.71 rows=17500 width=8) (actual time=15.826..15.829 rows=500 loops=1)
17      Buckets: 32768 Batches: 1 Memory Usage: 276kB
18      -> Hash Join (cost=667.75..1345.71 rows=17500 width=8) (actual time=14.519..15.640 rows=500 loops=1)
19      Hash Cond: (r2.sid = s2.sid)
20      -> Hash Join (cost=81.25..713.26 rows=17500 width=4) (actual time=5.303..6.104 rows=500 loops=1)
21      Hash Cond: (r2.bid = b2.bid)
22      -> Seq Scan on reserves r2 (cost=0.00..540.00 rows=35000 width=8) (actual time=0.020..2.291 rows=35000 loops=1)
23      -> Hash (cost=62.50..62.50 rows=1500 width=4) (actual time=0.486..0.487 rows=1500 loops=1)
24      Buckets: 2048 Batches: 1 Memory Usage: 69kB
25      -> Seq Scan on boat b2 (cost=0.00..62.50 rows=1500 width=4) (actual time=0.141..0.317 rows=1500 loops=1)
26      Filter: (color = 'green'::bpchar)
27      Rows Removed by Filter: 1500
28      -> Hash (cost=349.00..349.00 rows=19000 width=4) (actual time=9.008..9.009 rows=19000 loops=1)
29      Buckets: 32768 Batches: 1 Memory Usage: 924kB
30      -> Seq Scan on sailors s2 (cost=0.00..349.00 rows=19000 width=4) (actual time=0.035..3.383 rows=19000 loops=1)
31  Planning Time: 1.636 ms
32  Execution Time: 29.140 ms

```

Total rows: 32 of 32 Query complete 00:00:00.100 Ln 16, Col 37

Execution Plan and Costs

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Dashboard Properties SQL Statistics Dependencies Dependents schema3/postgres@PostgreSQL 14*

Data output Messages Explain Notifications

Graphical Analysis Statistics

1.	→ Hash Semi Join (rows=500 loops=1) Hash Cond: (s.sid = s2.sid)	500	1
2.	→ Hash Inner Join (rows=500 loops=1) Hash Cond: (r.sid = s.sid)	500	1
3.	→ Hash Inner Join (rows=500 loops=1) Hash Cond: (r.bid = b.bid)	500	1
4.	→ Seq Scan on reserves r (rows=35000 loops=1)	35000	1
5.	→ Hash (rows=1497 loops=1) Buckets: 2048 Batches: 1 Memory Usage: 69 kB	1497	1
6.	→ Seq Scan on boat as b (rows=1497 loops=1) Filter: (color = 'red'::bpchar) Row Removed by Filter: 1503	1497	1
7.	→ Hash (rows=19000 loops=1) Buckets: 32768 Batches: 1 Memory Usage: 1370 kB	19000	1
8.	→ Seq Scan on sailors as s (rows=19000 loops=1)	19000	1
9.	→ Hash (rows=500 loops=1) Buckets: 32768 Batches: 1 Memory Usage: 276 kB	500	1
10.	→ Hash Inner Join (rows=500 loops=1) Hash Cond: (r2.sid = s2.sid)	500	1
11.	→ Hash Inner Join (rows=500 loops=1) Hash Cond: (r2.bid = b2.bid)	500	1
12.	→ Seq Scan on reserves as r2 (rows=35000 loops=1) → Hash (rows=1500 loops=1)	35000	1

Total rows: 1 of 1 Query complete 00:00:00.056 Ln 16, Col 37

Optimised Query 9 with B+tree index on boat(color) Flags["seqscan = off", "hashjoin=on", "mergejoin = on", boat_pkey indisvalid = false else are true]

Planning and Execution time

```

pgAdmin 4
File Object Tools Help
schema3/postgres@PostgreSQL 14*
Dashboard Properties SQL Statistics Dependencies Dependents schema3/postgres@PostgreSQL 14*
Data output Messages Explain Notifications
QUERY PLAN
text
13  -> Memoize (cost=0.29..0.31 rows=1 width=4) (actual time=0.000..0.000 rows=0 loops=35000)
14    Cache Key: r2.bid
15    Cache Mode: logical
16    Hits: 34995 Misses: 5 Evictions: 0 Overflows: 0 Memory Usage: 1kB
17    -> Index Scan using boat_pkey on boat b2 (cost=0.28..0.30 rows=1 width=4) (actual time=0.006..0.006 rows=0 loops=5)
18      Index Cond: (bid = r2.bid)
19      Filter: (color = 'green'::bpchar)
20      Rows Removed by Filter: 1
21    -> Index Only Scan using sailors_pkey on sailors s2 (cost=0.29..501.29 rows=19000 width=4) (actual time=0.022..0.870 rows=11500 loops=1)
22      Heap Fetches: 0
23    -> Index Only Scan using reserves_pkey on reserves r (cost=0.29..917.29 rows=35000 width=8) (actual time=0.018..2.848 rows=34001 loops=1)
24      Heap Fetches: 0
25    -> Hash (cost=67.59..67.59 rows=1497 width=4) (actual time=0.439..0.440 rows=1497 loops=1)
26      Buckets: 2048 Batches: 1 Memory Usage: 69kB
27      -> Bitmap Heap Scan on boat b (cost=23.88..67.59 rows=1497 width=4) (actual time=0.065..0.242 rows=1497 loops=1)
28      Recheck Cond: (color = 'red'::bpchar)
29      Heap Blocks: exact=13
30      -> Bitmap Index Scan on colorbtree (cost=0.00..23.51 rows=1497 width=0) (actual time=0.055..0.055 rows=1497 loops=1)
31      Index Cond: (color = 'red'::bpchar)
32      Planning Time: 2.066 ms
33      Execution Time: 22.677 ms
Total rows: 33 of 33  Query complete 00:00:00.066

```

Successfully run. Total query runtime: 66 msec. 33 rows affected.

Execution Plan and Costs

Step	Operation	Cost	Rows
1.	Hash Join (rows=35000 loops=1)	500	1
2.	Merge Inner Join (rows=1000 loops=1)	1000	1
3.	Merge Semi Join (rows=500 loops=1)	500	1
4.	Index Scan using sailors_pkey on sailors as s (rows=11501 loops=1)	11501	1
5.	Merge Inner Join (rows=500 loops=1)	500	1
6.	Nested Loop Inner Join (rows=500 loops=1)	500	1
7.	Index Only Scan using reserves_pkey on reserves as r2 (rows=35000 loops=1)	35000	1
8.	Memoize (rows=0 loops=35000)	0	35000
	Buckets: Batches: Memory Usage: 1 kB		
9.	Index Scan using boat_pkey on boat as b2 (rows=0 loops=5)	0	5
	Filter: (color = 'green'::bpchar)		
	Index Cond: (bid = r2.bid)		
	Rows Removed by Filter: 1		
10.	Index Only Scan using sailors_pkey on sailors as s2 (rows=11500 loops=1)	11500	1
11.	Index Only Scan using reserves_pkey on reserves as r (rows=34001 loops=1)	34001	1
12.	Hash (rows=1497 loops=1)	1497	1
	Buckets: 2048 Batches: 1 Memory Usage: 69 kB		
13.	Bitmap Heap Scan on boat b (rows=1497 loops=1)	1497	1
	Recheck Cond: (color = 'red'::bpchar)		
	Heap Blocks: exact=13		
14.	Bitmap Index Scan using colorbtree (rows=1497 loops=1)	1497	1
	Index Cond: (color = 'red'::bpchar)		

Total rows: 1 of 1 Query complete 00:00:00.061

B+tree index optimised here as it was used to search(in O(logn)) for boat.color = 'red' and boat.color = 'green'

Optimised Query 9 with Hash index on boat(color) Flags["seqscan = off", "hashjoin=on", "mergejoin = on", boat_pkey indisvalid = false else are true]

Planning and Execution time

The screenshot shows the pgAdmin 4 interface with the Explain tab selected. The query plan is displayed in a table format:

Step	Operation	Cost	Rows	Memory Usage
13	-> Memoize (cost=0.29..0.31 rows=1 width=4)	0.000..0.000	0	35000
14	Cache Key: r2.bid			
15	Cache Mode: logical			
16	Hits: 34995 Misses: 5 Evictions: 0 Overflows: 0 Memory Usage: 1kB			
17	-> Index Scan using boat_pkey on boat b2 (cost=0.28..0.30 rows=1 width=4)	0.003..0.003	0	5
18	Index Cond: (bid = r2.bid)			
19	Filter: (color = 'green':bpchar)			
20	Rows Removed by Filter: 1			
21	-> Index Only Scan using sailors_pkey on sailors s2 (cost=0.29..0.501 rows=19000 width=4)	0.010..0.054	11500	1
22	Heap Fetches: 0			
23	-> Index Only Scan using reserves_pkey on reserves r (cost=0.29..0.917 rows=35000 width=8)	0.008..0.271	34001	1
24	Heap Fetches: 0			
25	-> Hash (cost=103.31..103.31 rows=1497 width=4)	0.410..0.411	1497	1
26	Buckets: 2048 Batches: 1 Memory Usage: 69kB			
27	-> Bitmap Heap Scan on boat b (cost=59.60..103.31 rows=1497 width=4)	0.058..0.271	1497	1
28	Recheck Cond: (color = 'red':bpchar)			
29	Heap Blocks: exact=13			
30	-> Bitmap Index Scan on colorhash (cost=0.00..59.23 rows=1497 width=0)	0.047..0.047	1497	1
31	Index Cond: (color = 'red':bpchar)			
32	Planning Time: 1.013 ms			
33	Execution Time: 18.294 ms			

Total rows: 33 of 33 Query complete 00:00:00.049

Ln 16, Col 37

Execution Plan and Costs

The screenshot shows the pgAdmin 4 interface with the Explain tab selected. The execution plan is displayed in a table format:

Step	Operation	Cost	Rows	Memory Usage
1.	-> Hash Inner Join (rows=500 loops=1) Hash Cond: (r.bid = b.bid)	500	1	
2.	-> Merge Inner Join (rows=1000 loops=1)	1000	1	
3.	-> Merge Semi Join (rows=500 loops=1)	500	1	
4.	-> Index Scan using sailors_pkey on sailors as s (rows=11501 loops=1)	11501	1	
5.	-> Merge Inner Join (rows=500 loops=1)	500	1	
6.	-> Nested Loop Inner Join (rows=500 loops=1)	500	1	
7.	-> Index Only Scan using reserves_pkey on reserves as r2 (rows=35000 loops=1)	35000	1	
8.	-> Memoize (rows=0 loops=35000) Buckets: Batches: Memory Usage: 1 kB	0	35000	
9.	-> Index Scan using boat_pkey on boat as b2 (rows=0 loops=5) Filter: (color = 'green':bpchar) Index Cond: (bid = r2.bid) Rows Removed by Filter: 1	0	5	
10.	-> Index Only Scan using sailors_pkey on sailors as s2 (rows=11500 loops=1)	11500	1	
11.	-> Index Only Scan using reserves_pkey on reserves as r (rows=34001 loops=1)	34001	1	
12.	-> Hash (rows=1497 loops=1) Buckets: 2048 Batches: 1 Memory Usage: 69 kB	1497	1	
13.	-> Bitmap Heap Scan on boat as b (rows=1497 loops=1) Recheck Cond: (color = 'red':bpchar) Heap Blocks: exact=13	1497	1	
	-> Bitmap Index Scan using colorhash (rows=1497 loops=1)	1497	1	

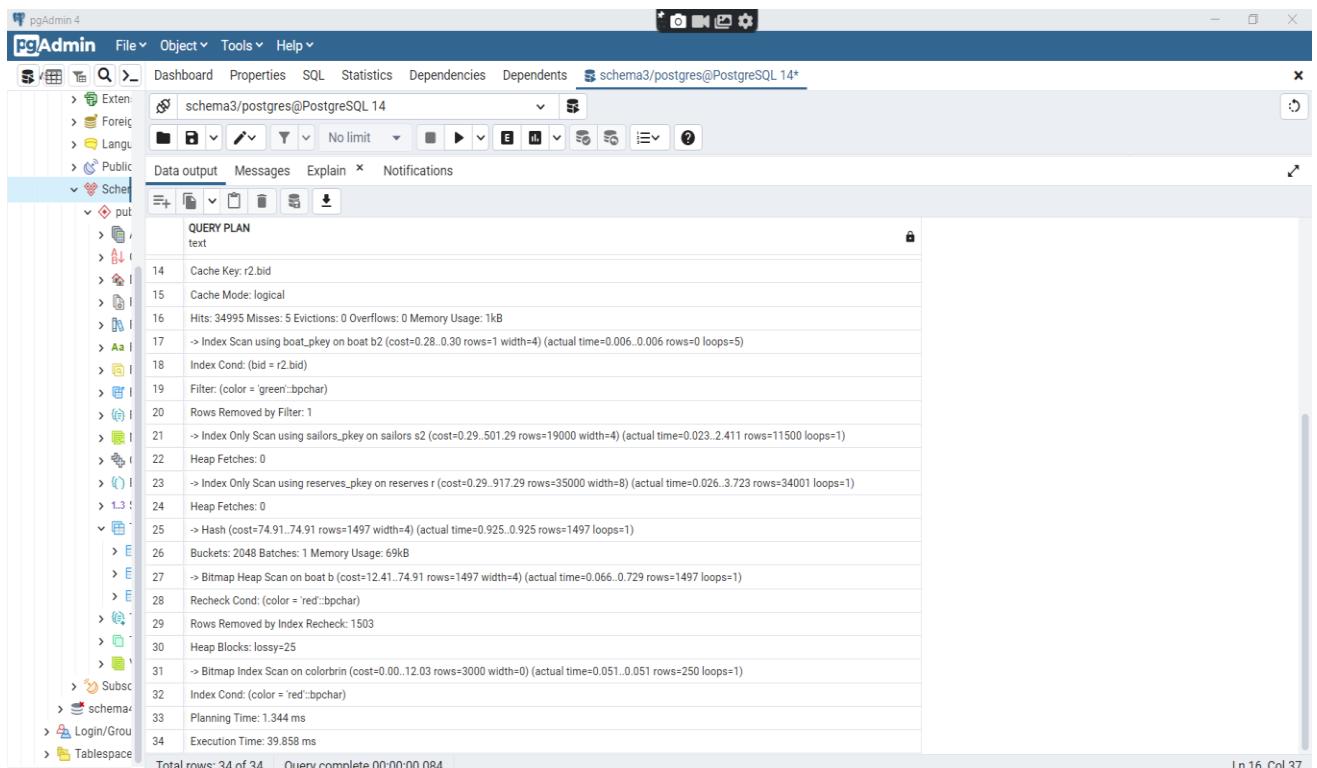
Total rows: 1 of 1 Query complete 00:00:00.146

Ln 16, Col 37

Hash index optimised here as it was used to search(in O(1)) for boat.color = 'red' and boat.color = 'green'

Optimised Query 9 with BRIN on boat(color) Flags[“seqscan = off”, “hashjoin=on”, “mergejoin = on”, boat_pkey indisvalid = false else are true]

Planning and Execution time

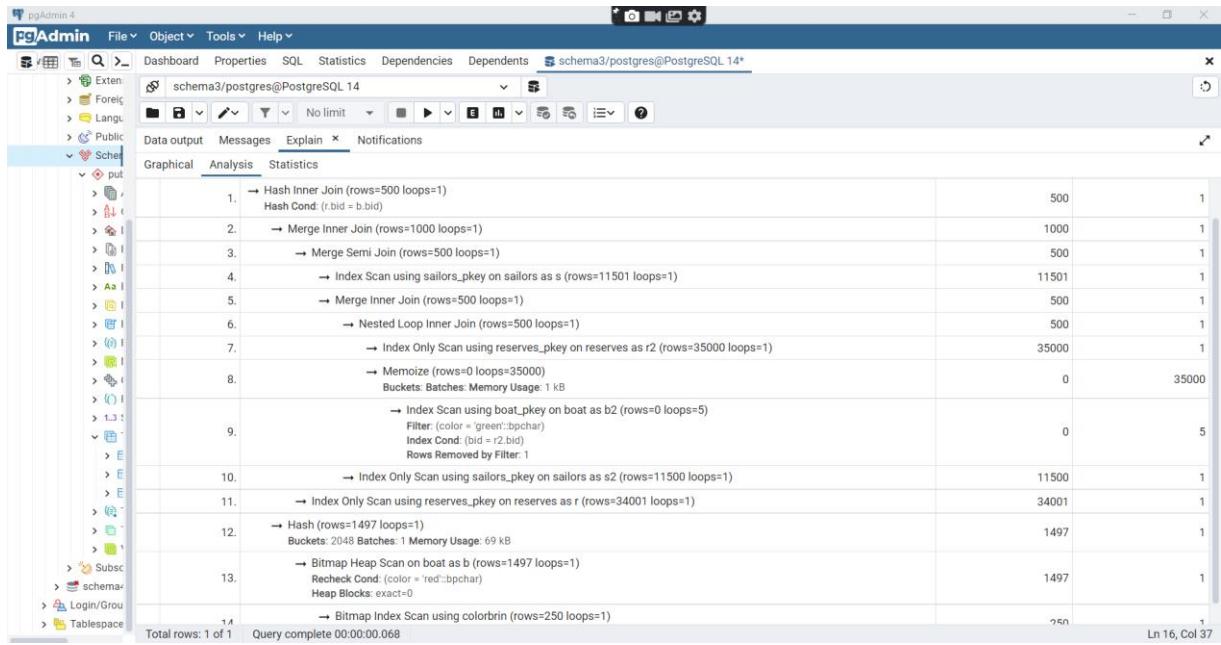


The screenshot shows the pgAdmin 4 interface with the Explain tab selected. The query plan is displayed in a table format, showing the execution steps and their details.

Step	Operation	Details
14	Cache Key:	r2.bid
15	Cache Mode:	logical
16	Hits:	34995 Misses: 5 Evictions: 0 Overflows: 0 Memory Usage: 1kB
17	> Index Scan using boat_pkey on boat b2	(cost=0.28..0.30 rows=1 width=4) (actual time=0.006..0.006 rows=0 loops=5)
18	Index Cond:	(bid = r2.bid)
19	Filter:	(color = 'green':bpchar)
20	Rows Removed by Filter:	1
21	> Index Only Scan using sailors_pkey on sailors s2	(cost=0.29..501.29 rows=19000 width=4) (actual time=0.023..2.411 rows=11500 loops=1)
22	Heap Fetches:	0
23	> Index Only Scan using reserves_pkey on reserves r	(cost=0.29..917.29 rows=35000 width=8) (actual time=0.026..3.723 rows=34001 loops=1)
24	Heap Fetches:	0
25	> Hash	(cost=74.91..74.91 rows=1497 width=4) (actual time=0.925..0.925 rows=1497 loops=1)
26	Buckets:	2048 Batches: 1 Memory Usage: 69kB
27	> Bitmap Heap Scan on boat b	(cost=12.41..74.91 rows=1497 width=4) (actual time=0.066..0.729 rows=1497 loops=1)
28	Recheck Cond:	(color = 'red':bpchar)
29	Rows Removed by Index Recheck:	1503
30	Heap Blocks:	lossy=25
31	> Bitmap Index Scan on colorbrin	(cost=0.00..12.03 rows=3000 width=0) (actual time=0.051..0.051 rows=250 loops=1)
32	Index Cond:	(color = 'red':bpchar)
33	Planning Time:	1.344 ms
34	Execution Time:	39.858 ms

Total rows: 34 of 34 Query complete 00:00:00.084 Ln 16, Col 37

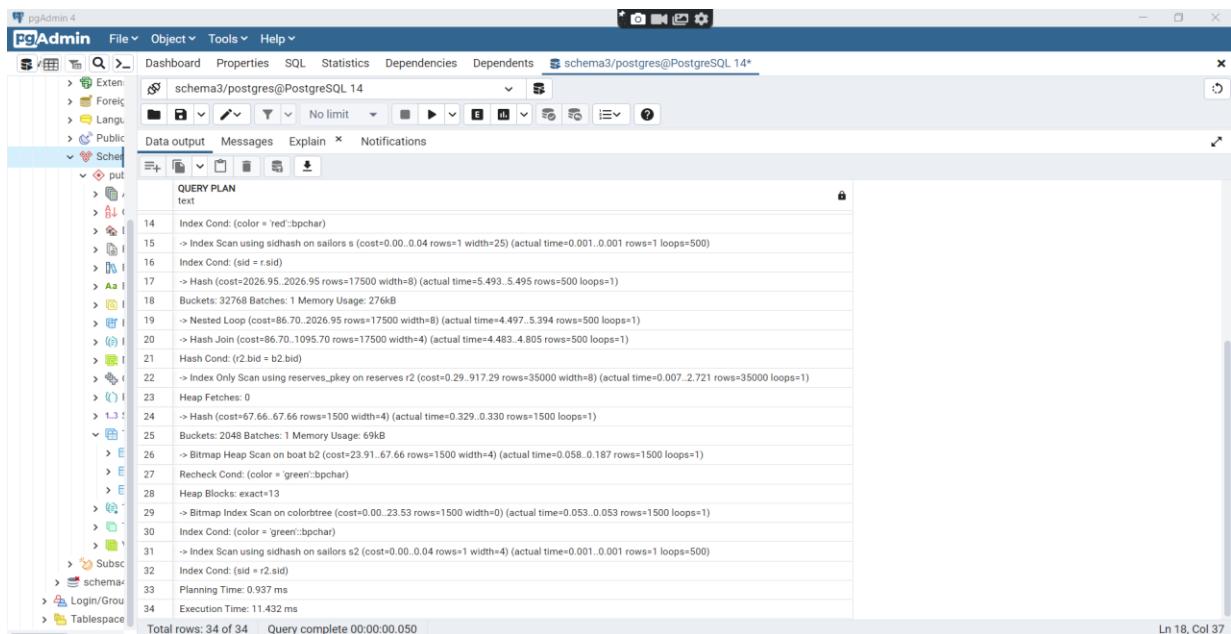
Execution Plan and Costs



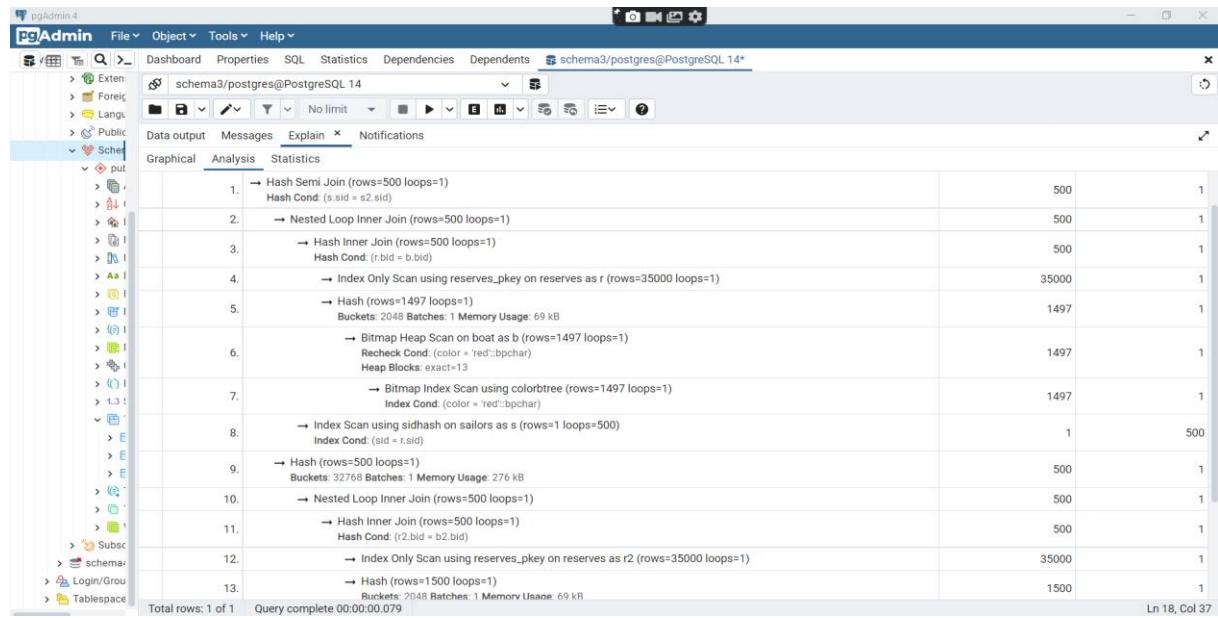
BRIN index here didn't optimize here as the search on boat.color was an exact value and the search BRIN is suitable for small range queries in a table with a very large range

Optimised Query 9 with B+tree index on boat(color) and Hash index on sailors(sid) Flags[“seqscan = off”, “hashjoin=on”, “mergejoin = on”, sailors_pkey and boat_pkey indisvalid = false else are true]]

Planning and Execution time



Execution Plan and Costs



Since the search on `b.color = 'red'` and `b.color = 'green'` is an exact value and search on `sailors.sid = reserves.sid` is an exact value having a hash index(search time = O(1)) on `sailors.sid` and B+tree index(search time = O(logn)) on `boat.color` improved query performance