

## Query 8

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

### Note !

- Flags Hashjoin and HashAgg here were disabled for future after many trials and errors , I've discovered the best way to show the difference in terms of the cost and to beat the Postgres Query Optimizer Algorithm to be able to show indices effect and cost differences .
- The Execution time was changing by 10 Ms in each Execution which is considered high and I can't take it as a measurable Metric because it was Linux (Ubuntu) Operating System performance and I took permission from Prof. Wael as do not take it as my main objective I take the Overall Cost and Compare it .

### Original Query

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

### Result Set

- 673 Rows

The screenshot shows the pgAdmin 4 interface. On the left is a sidebar with various icons for database management tasks. The main area has a title bar "pgAdmin 4" and a status bar at the bottom indicating "May 17 12:03 AM". The central part is a query editor window. The query itself is a multi-step analysis of a query, starting with an explain analyze select statement and followed by several other statements including joins and subqueries. Below the query text is a "Data output" tab showing a single row of results:

count	bigint
1	673

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

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

## Report

1. given query without an index :

pgAdmin 4

May 17 12:08 AM

schema3/postgres@postgres\*

```

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

```

Data output

schemaname	tablename	indexname	tablespace	indexdef
name	name	name	name	text

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

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

Ln 35, Col 1

pgAdmin 4

Jun 9 6:05 PM

schema3/postgres@postgres\*

Explain Analyze [Shift F7]

```

208
209
210
211 DROP INDEX IF EXISTS b_sailorsSID cascade;
212 DROP INDEX IF EXISTS b_reservesSID cascade;
213 DROP INDEX IF EXISTS b_reservesBID cascade;
214 DROP INDEX IF EXISTS R_reservesBID cascade;
215
216
217 set enable_hashagg = off;
218 set enable_hashjoin = off;
219 set enable_seqscan = on;
220 -- Query 8 (STATISTICS)
221
222
223
224 explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
Total rows: 1 of 1    Query complete 00:00:00.054

```

Graphical Analysis Statistics

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

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

Ln 224, Col 17

```

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

set enable_hashagg = off;
set enable_hashjoin = off;
set enable_seqscan = on;
-- Query 8 (STATISTICS)

explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
-- Query 8 optimized (STATISTICS)

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

-- ====== Query 9 ======

```

Total rows: 23 of 23    Query complete 00:00:00.082

DATA output    Messages    Explain    Notifications

QUERY PLAN text

- Merge Semi Join (cost=5495.01..5664.74 rows=4982 width=21) (actual time=16.262..16.794 rows=673 loops=1)
  - Merge Cond: (s.sid = r.sid)
  - > Sort (cost=1699.30..1746.80 rows=19000 width=25) (actual time=5.422..5.563 rows=2999 loops=1)
    - > Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.011..2.045 rows=19000 loops=1)
    - > Sort (cost=3795.70..3808.16 rows=4982 width=4) (actual time=10.837..10.889 rows=1136 loops=1)
      - Sort Key: r.sid
  - Sort Method: quicksort Memory: 102kB
  - > Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.011..2.045 rows=19000 loops=1)
    - Sort Key: r.sid
  - Sort Method: quicksort Memory: 102kB
  - > Merge Semi Join (cost=3262.79..3489.75 rows=4982 width=4) (actual time=10.338..10.653 rows=1136 loops=1)
    - Merge Cond: (r.bid = b.bid)
    - > Sort (cost=181.64..3269.14 rows=35000 width=8) (actual time=9.985..10.044 rows=1137 loops=1)
      - Sort Key: b.bid
    - Sort Method: quicksort Memory: 300kB
    - > Seq Scan on reserves r (cost=0.00..540.00 rows=35000 width=8) (actual time=0.010..3.599 rows=35000 loops=1)
      - Sort Key: b.bid
    - > Sort (cost=81.16..82.22 rows=427 width=4) (actual time=0.349..0.370 rows=427 loops=1)
      - Sort Key: b.bid
    - Sort Method: quicksort Memory: 45kB
    - > Seq Scan on boat b (cost=0.00..62.50 rows=427 width=4) (actual time=0.012..0.306 rows=427 loops=1)
      - Filter: (color = 'red'::bpchar)
    - Rows Removed by Filter: 2573
  - Planning Time: 0.126 ms
  - Execution Time: 17.183 ms

✓ Successfully run. Total query runtime: 82 msec. 23 rows affected

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

Ln 217, Col 1

## Explanation :

- Metrics :

**Execution Time : 17.183 ms    Total Expected Cost : 5664.74**

- given query with B+ trees indices only :

The screenshot shows the pgAdmin 4 application window. The left sidebar displays the 'Servers' tree, with 'schema3/postgres@postgres' selected. The main area contains a query editor with the following SQL code:

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

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

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

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

set enable_hashagg = off;
set enable_hashjoin = off;
set enable_seqscan = on;
-- Query 8 (STATISTICS)

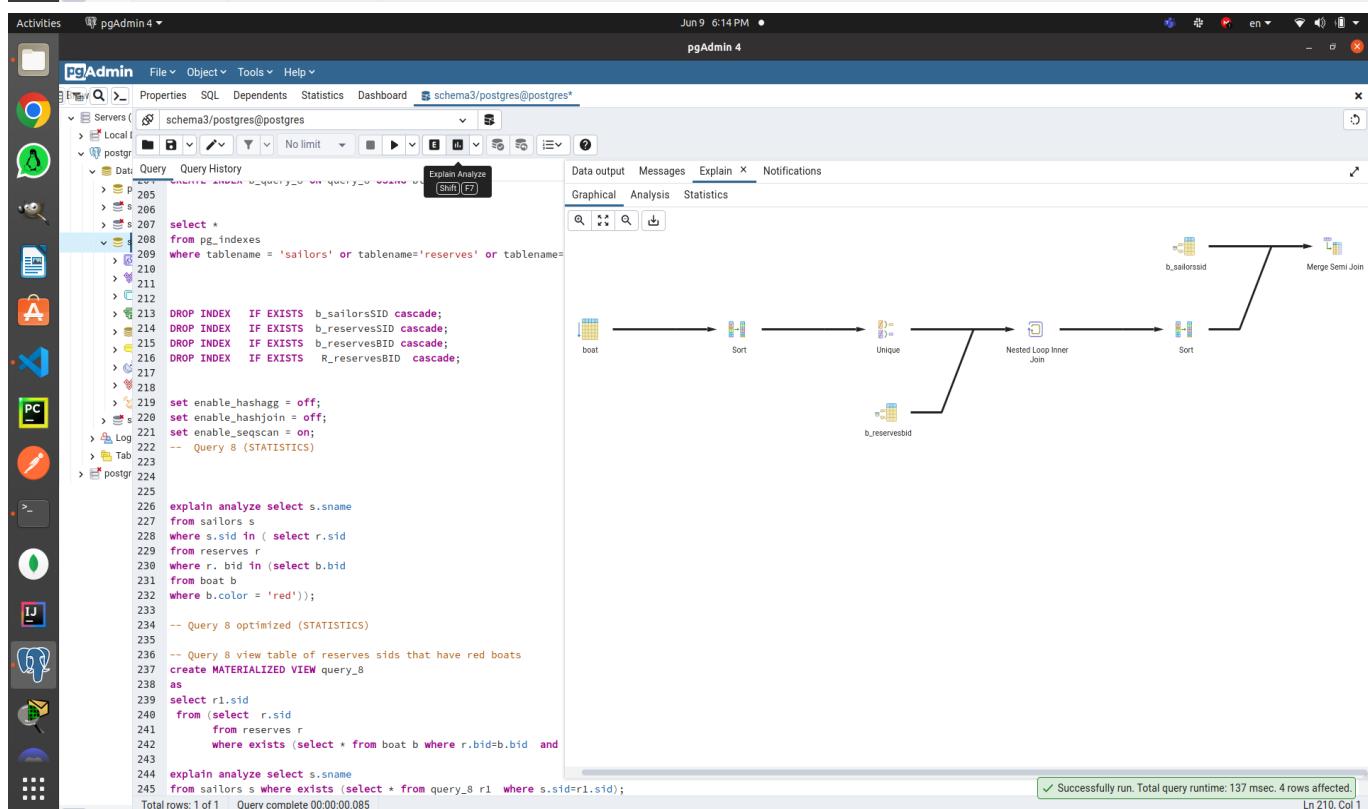
explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
```

The status bar at the bottom indicates "Total rows: 5 of 5 Query complete 00:00:00.070".

The right panel shows the results of the last query in a grid format:

schemaname	tablename	indexname	tablespace	Indexdef
1 public	sailors	b_sailorssid	[null]	CREATE INDEX b_sailorssid ON public.sailors USING btree (sid )
2 public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesSID ON public.reserves USING btree (sid )
3 public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesBID ON public.reserves USING btree (bid)
4 public	boat	b_boat	[null]	CREATE INDEX b_boat ON public.boat USING btree (bid,color )
5 public	query_8	b_query_8	[null]	CREATE INDEX b_query_8 ON public.query_8 USING btree (sid )

A message bar at the bottom right says "Successfully run. Total query runtime: 137 msec. 4 rows affected." and "Ln 206, Col 1".



```

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

```

Execution Plan:

```

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

```

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

### Explanation :

- Metrics :

**Execution Time : 2.421 ms    Total Expected Cost : 2779.35**

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

### 3. given query with hash indices only :

Activities pgAdmin 4 Jun 9 6:19 PM ● pgAdmin 4

Servers Local PostgreSQL schema3/postgres\* Data Query History

```

> p 196
> s 197
> s 198
> s 199
> s 200 CREATE INDEX b_sailorsSID ON sailors USING hash(sid );
> s 201 CREATE INDEX b_reservesID ON reserves USING hash(sid );
> s 202 CREATE INDEX b_reservesBID ON reserves USING hash(bid );
> s 203 CREATE INDEX b_boat1 ON boat USING hash(bid );
> s 204 CREATE INDEX b_boat2 ON boat USING hash(color );
> s 205
> s 206 CREATE INDEX b_query_8 ON query_8 USING hash(sid );
> s 207
> s 208
> s 209 select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_8';
> s 210
> s 211
> s 212
> Log Tab 214
> postgres 215 DROP INDEX IF EXISTS b_sailorsSID cascade;
216 DROP INDEX IF EXISTS b_reservesID cascade;
217 DROP INDEX IF EXISTS b_reservesBID cascade;
218 DROP INDEX IF EXISTS b_boat1 cascade;
219 DROP INDEX IF EXISTS b_boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = on;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
-- ====== Query 9 ======
256

```

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

Data output Messages Explain Notifications

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

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

Activities pgAdmin 4 Jun 9 6:20 PM ● pgAdmin 4

Servers Local PostgreSQL schema3/postgres\* Data Query History

```

> p 216 DROP INDEX IF EXISTS b_reservesID cascade;
> s 217 CREATE INDEX b_reservesBID ON reserves USING hash(bid );
> s 218 CREATE INDEX b_boat1 ON boat USING hash(bid );
> s 219 CREATE INDEX b_boat2 ON boat USING hash(color );
> s 220
> s 221 DROP INDEX IF EXISTS b_query_8 cascade;
> s 222
> s 223
> s 224 set enable_hashagg = off;
> s 225 set enable_hashjoin = off;
> s 226 set enable_seqscan = on;
> s 227 -- Query 8 (STATISTICS)
> s 228
> s 229
> s 230
> s 231 explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
> s 232
> s 233
> s 234
> s 235
> s 236
> s 237
> s 238
> s 239
> s 240
> s 241 -- Query 8 view table of reserves sids that have red boats
> s 242 create MATERIALIZED VIEW query_8
> s 243 as
> s 244 select r1.sid
> s 245 from (select r.sid
> s 246 from reserves r
> s 247 where exists (select * from boat b where r.bid=b.bid and color='red' ) )
> s 248
> s 249 explain analyze select s.sname
> s 250 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
> s 251
> s 252
> s 253
> s 254
> s 255
-- ====== Query 9 ======
> s 256

```

Data output Messages Explain Analysis Statistics

```

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

```

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

```

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

```

Total rows: 15 of 15    Query complete 00:00:00.106

SuccessFully run. Total query runtime: 106 msec. 15 rows affected

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

Ln 229, Col 1

### Explanation :

- Metrics :

**Execution Time : 0.993 ms    Total Expected Cost : 2168.36**

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

### 4. given query with BRIN indices only :

pgAdmin 4

Jun 9 6:22 PM • pgAdmin 4

Servers Local PostgreSQL

Dat Query History

```

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

Total rows: 6 of 6 Query complete 00:00:00.067

Data output Messages Explain Notifications

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

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

Ln 208, Col 1

pgAdmin 4

Jun 9 6:24 PM • pgAdmin 4

Servers Local PostgreSQL

Dat Query History

```

> p 199 select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves'
> s 211
> s 212
> s 213
> s 214
> s 215
> s 216
> s 217
> s 218
> s 219
> s 220
> s 221
> s 222
> s 223
> s 224
> s 225
> s 226
> s 227
> s 228
> s 229
> s 230
> s 231
> s 232
> s 233
> s 234
> s 235
> s 236
> s 237
> s 238
> s 239
> s 240
> s 241
> s 242
> s 243
> s 244
> s 245
> s 246
> s 247
> s 248
> s 249
-- Query 8 optimized (STATISTICS)
-- Query 8 view table of reserves sids that have
create MATERIALIZED VIEW query_8
as
select r1.sid
from (select r.sid
from reserves r
where exists (select * from boat b where
b.bid = r.bid)) r1
join sailors s
on r1.sid = s.sid;
```

Total rows: 1 of 1 Query complete 00:00:01.895

Explain Analyze

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

Ln 231, Col 17

```

Jun 9 6:23 PM • pgAdmin 4
pgAdmin 4

Activities pgAdmin 4 ▾ Jun 9 6:23 PM • pgAdmin 4

Servers schema3/postgres Local postgr
  Data Query History
    > P 209
    > S 210
    > S 211
    where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='que
    > S 212
    > S 213
    > S 214
    > S 215
    > S 216
    > S 217
    > S 218
    > S 219
    > S 220
    > S 221
    > S 222
    > S 223
    > S 224
    set enable_hashagg = off;
    set enable_hashjoin = off;
    set enable_seqscan = off;
    -- Query 8 (STATISTICS)
    > Log 225
    > Tab 226
    > postgr 227
    228
    229
    230
    231 explain analyze select s.sname
    from sailors s
    where s.sid in ( select r.sid
    from reserves r
    where r.bid in (select b.bid
    from boat b
    where b.color = 'red'));
    232
    -- Query 8 optimized (STATISTICS)
    233
    -- Query 8 view table of reserves sids that have red boats
    234 create MATERIALIZED VIEW query_8
    as
    235
    236
    237
    238
    239
    240
    241
    242
    243
    244
    245
    246
    247
    248
    249
    explain analyze select s.sname
    Total rows: 34 of 34 Query complete 0:00:01.887

Data output Messages Explain Notifications
QUERY PLAN
text
1 Nested Loop (cost=2602463.80..1126150803.66 rows=4982 width=21) (actual time=1064.570..1871.485 rows=1136 loops=1)
  -> Unique (cost=2377019.76..2377044.67 rows=4982 width=4) (actual time=1064.528..1065.172 rows=673 loops=1)
  3 Sort (cost=2377019.76..2377032.22 rows=4982 width=4) (actual time=1064.527..1064.774 rows=1136 loops=1)
  4 Sort Key: r.sid
  5 Sort Method: quicksort Memory: 102k8
  6 Sort (cost=5217.37..2376713.81 rows=4982 width=4) (actual time=134.092..1064.060 rows=1136 loops=1)
  7 -> Unique (cost=93.29..95.43 rows=427 width=4) (actual time=133.922..134.387 rows=427 loops=1)
  8 -> Sort (cost=93.29..94.36 rows=427 width=4) (actual time=133.921..134.070 rows=427 loops=1)
  9 Sort Key: b.bid
  10 Sort Method: quicksort Memory: 45kB
  11 -> Bitmap Heap Scan on boat b (cost=12.14..74.64 rows=427 width=4) (actual time=133.220..133.849 rows=427 loops=1)
  12 Recheck Cond: (color = 'red'::bpchar)
  13 Rows Removed by Index Recheck: 2573
  14 Heap Blocks: lossy=25
  15 -> Bitmap Index Scan on b.boat2 (cost=0.00..12.03 rows=3000 width=0) (actual time=0.041..0.041 rows=250 loops=1)
  16 Index Cond: (color = 'red'::bpchar)
  17 -> Bitmap Heap Scan on reserves r (cost=5124.07..5565.57 rows=28 width=8) (actual time=0.089..2.173 rows=3 loops=427)
  18 Recheck Cond: (bid = b.bid)
  19 Rows Removed by Index Recheck: 23677
  20 Heap Blocks: lossy=54656
  21 -> Bitmap Index Scan on b.reservebid (cost=0.00..5124.06 rows=35000 width=0) (actual time=0.012..0.012 rows=1280 loops=427)
  22 Index Cond: (bid = b.bid)
  23 -> Bitmap Heap Scan on sailors s (cost=225444.03..225566.78 rows=1 width=25) (actual time=0.041..1.195 rows=1 loops=673)
  24 Recheck Cond: (sid = r.sid)
  25 Rows Removed by Index Recheck: 15359
  26 Heap Blocks: lossy=86144
  27 -> Bitmap Index Scan on b.sailorssid (cost=0.00..225444.03 rows=9500 width=0) (actual time=0.011..0.011 rows=1280 loops=673)
  28 Index Cond: (sid = r.sid)
  29 Planning Time: 0.321 ms

Successfully run. Total query runtime: 137 msec. 4 rows affected.
Ln 224, Col 1

Activities pgAdmin 4 ▾ Jun 9 6:23 PM • pgAdmin 4
pgAdmin 4

Servers schema3/postgres Local postgr
  Data Query History
    > P 209
    > S 210
    > S 211
    where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='que
    > S 212
    > S 213
    > S 214
    > S 215
    > S 216
    > S 217
    > S 218
    > S 219
    > S 220
    > S 221
    > S 222
    > S 223
    > S 224
    set enable_hashagg = off;
    set enable_hashjoin = off;
    set enable_seqscan = off;
    -- Query 8 (STATISTICS)
    > Log 225
    > Tab 226
    > postgr 227
    228
    229
    230
    231
    232
    233
    234
    235
    236
    237
    238
    239
    240
    241
    242
    243
    244
    245
    246
    247
    248
    249
    explain analyze select s.sname
    Total rows: 34 of 34 Query complete 0:00:01.887

Data output Messages Explain Notifications
QUERY PLAN
text
6 -> Nested Loop (cost=5217.37..2376713.81 rows=4982 width=4) (actual time=134.092..1064.060 rows=1136 loops=1)
  7 -> Unique (cost=93.29..95.43 rows=427 width=4) (actual time=133.922..134.387 rows=427 loops=1)
  8 -> Sort (cost=93.29..94.36 rows=427 width=4) (actual time=133.921..134.070 rows=427 loops=1)
  9 Sort Key: b.bid
  10 Sort Method: quicksort Memory: 45kB
  11 -> Bitmap Heap Scan on boat b (cost=12.14..74.64 rows=427 width=4) (actual time=133.220..133.849 rows=427 loops=1)
  12 Recheck Cond: (color = 'red'::bpchar)
  13 Rows Removed by Index Recheck: 2573
  14 Heap Blocks: lossy=25
  15 -> Bitmap Index Scan on b.boat2 (cost=0.00..12.03 rows=3000 width=0) (actual time=0.041..0.041 rows=250 loops=1)
  16 Index Cond: (color = 'red'::bpchar)
  17 -> Bitmap Heap Scan on reserves r (cost=5124.07..5565.57 rows=28 width=8) (actual time=0.089..2.173 rows=3 loops=427)
  18 Recheck Cond: (bid = b.bid)
  19 Rows Removed by Index Recheck: 23677
  20 Heap Blocks: lossy=54656
  21 -> Bitmap Index Scan on b.reservebid (cost=0.00..5124.06 rows=35000 width=0) (actual time=0.012..0.012 rows=1280 loops=427)
  22 Index Cond: (bid = b.bid)
  23 -> Bitmap Heap Scan on sailors s (cost=225444.03..225566.78 rows=1 width=25) (actual time=0.041..1.195 rows=1 loops=673)
  24 Recheck Cond: (sid = r.sid)
  25 Rows Removed by Index Recheck: 15359
  26 Heap Blocks: lossy=86144
  27 -> Bitmap Index Scan on b.sailorssid (cost=0.00..225444.03 rows=9500 width=0) (actual time=0.011..0.011 rows=1280 loops=673)
  28 Index Cond: (sid = r.sid)
  29 Planning Time: 0.321 ms
  30 JIT:
  31 Functions: 16
  32 Options:Inlining true, Optimization true, Expressions true, Deforming true
  33 Timing: Generation 1.444 ms, Inlining 6.898 ms, Optimization 72.225 ms, Emission 53.792 ms, Total 134.359 ms
  34 Execution Time: 1873.233 ms

Successfully run. Total query runtime: 137 msec. 4 rows affected.
Ln 224, Col 1

```

## Explanation :

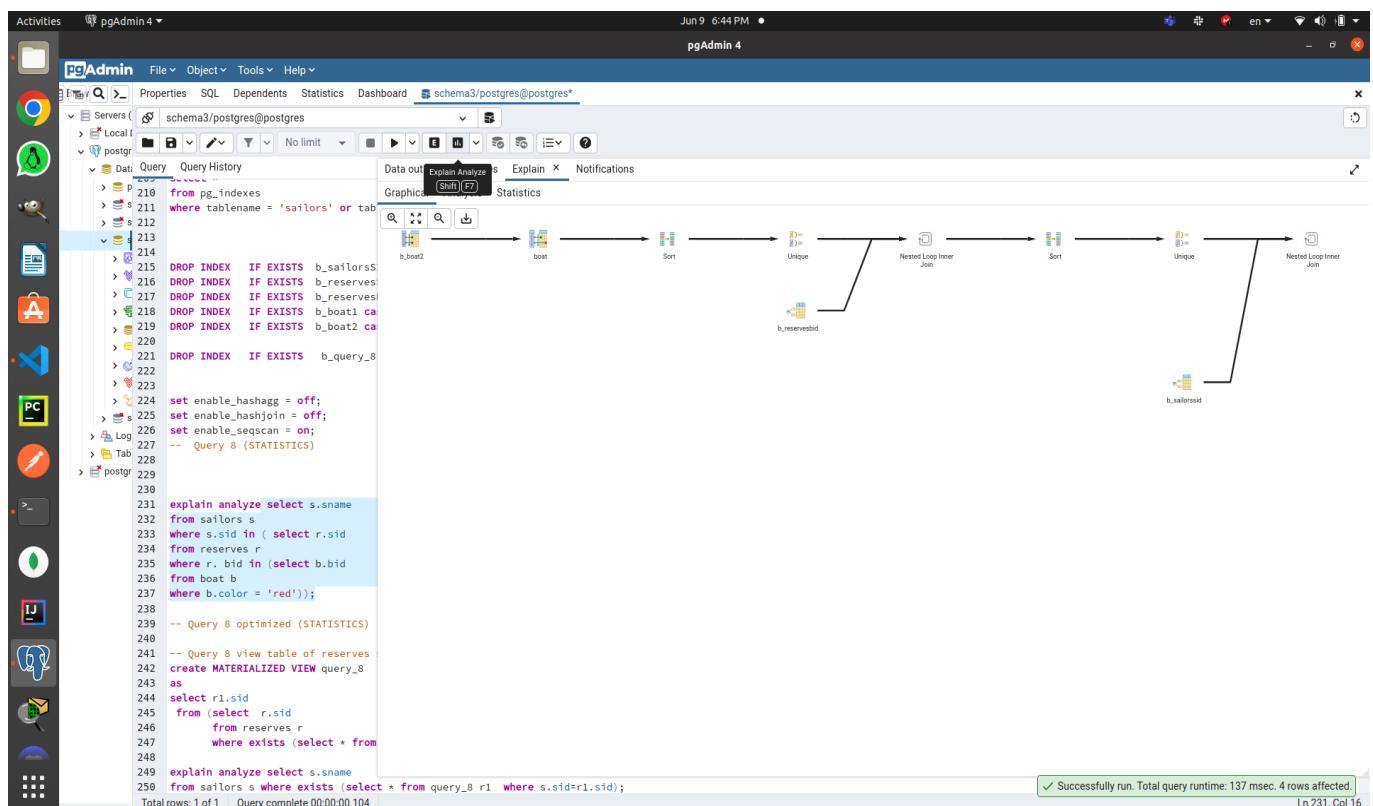
- Metrics :

**Execution Time : 18733.23 ms    Total Expected Cost : 1126150803.66**

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

- This happened because the Query Optimizer didn't use it from the first place due to BRIN Usage here was not suitable so we have used it to simulate seqscan behaviour only we traversed it all and followed all its pointers so it is worst index to use in this case.

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



```

210 from pg_indexes
211 where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query';
212
213
214
215 DROP INDEX IF EXISTS b_sailorsSID cascade;
216 DROP INDEX IF EXISTS b_reservesSID cascade;
217 DROP INDEX IF EXISTS b_reservesBID cascade;
218 DROP INDEX IF EXISTS b_boat1 cascade;
219 DROP INDEX IF EXISTS b_boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = on;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
232 from sailors s
233 where s.sid in ( select r.sid
234 from reserves r
235 where r.bid in (select b.bid
236 from boat b
237 where b.color = 'red'));
238
239 -- Query 8 optimized (STATISTICS)
240
241 -- Query 8 view table of reserves sids that have red boats
242 create MATERIALIZED VIEW query_8
243 as
244 select r1.sid
245 from (select r.sid
246 from reserves r
247 where exists (select * from boat b where r.bid=b.bid and color='red')) as r1;
248
249 explain analyze select s.sname
250 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
Total rows: 21 of 21   Query complete 00:00:00.093

```

### Explanation :

- Metrics :

**Execution Time : 2.486 ms    Total Expected Cost : 2373.87**

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

### Optimized Query

```

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

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

explain analyze select s.sname

```

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

## Result Set

- 673 Rows

## Report

1. given query without an index :

pgAdmin 4

```

Activities pgAdmin 4 Jun 9 6:07 PM ●
File Object Tools Help
Servers schema3/postgres Local PostgreSQL No limit
Data Query Explain Analyze Shift [F7]
210
211 set enable_hashagg = off;
212 set enable_hashjoin = off;
213 set enable_seqscan = on;
214 -- Query 8 (STATISTICS)
215
216
217
218
219
220
221
222
223
224 explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
225
226
227
228
229
230
231
232 -- Query 8 optimized (STATISTICS)
233
234 -- Query 9 view table of reserves sids that have red boats
create MATERIALIZED VIEW query_8
235
236 as
237 select r1.sid
238 from (select r.sid
239 from reserves r
240 where exists (select * from boat b where r.bid=b.bid and color='red' )) as r1 ;
241
242 explain analyze select s.sname
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
243
244
245
246
247
248 -- ====== Query 9 ======
249
250 -- Find the names of sailors who have reserved both a red and a green boat.
251
252 -- Query 9 (COUNTING RESULT SET) , Number Of Rows = 177
253
254
255 select count(s.sname)
256 from sailors s, reserves r, boat b
257 where
Total rows: 1 of 1 Query complete 00:00:00.051

```

Data output Messages Explain Notifications

Graphical Analysis Statistics

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

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

Ln 242, Col 17

pgAdmin 4

```

Activities pgAdmin 4 Jun 9 6:07 PM ●
File Object Tools Help
Servers schema3/postgres Local PostgreSQL No limit
Data Query Explain Refresh [F5]
210
211 set enable_hashagg = off;
212 set enable_hashjoin = off;
213 set enable_seqscan = on;
214 -- Query 8 (STATISTICS)
215
216
217
218
219
220
221
222
223
224 explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red'));
225
226
227
228
229
230
231
232 -- Query 8 optimized (STATISTICS)
233
234 -- Query 9 view table of reserves sids that have red boats
create MATERIALIZED VIEW query_8
235
236 as
237 select r1.sid
238 from (select r.sid
239 from reserves r
240 where exists (select * from boat b where r.bid=b.bid and color='red' )) as r1 ;
241
242 explain analyze select s.sname
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
243
244
245
246
247
248 -- ====== Query 9 ======
249
250 -- Find the names of sailors who have reserved both a red and a green boat.
251
252 -- Query 9 (COUNTING RESULT SET) , Number Of Rows = 177
253
254
255 select count(s.sname)
256 from sailors s, reserves r, boat b
257 where
Total rows: 12 of 12 Query complete 00:00:00.069

```

Data output Messages Explain Notifications

QUERY PLAN text

- Merge Semi Join (cost=1774.32..1806.34 rows=673 width=21) (actual time=5.560..6.097 rows=673 loops=1)
- Merge Cond: (s.sid = r1.sid)
- Sort (cost=1699.30..1746.80 rows=19000 width=25) (actual time=5.314..5.455 rows=2999 loops=1)
- Sort Key: s.sid
- Sort Method: quicksort Memory: 225kB
- Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.010..1.995 rows=19000 loops=1)
- Sort (cost=75.01..77.85 rows=1136 width=4) (actual time=0.243..0.295 rows=1136 loops=1)
- Sort Key: r1.sid
- Sort Method: quicksort Memory: 102kB
- Seq Scan on query\_8 r1 (cost=0.00..17.36 rows=1136 width=4) (actual time=0.009..0.087 rows=1136 loops=1)
- Planning Time: 0.078 ms
- Execution Time: 6.290 ms

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

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

Ln 242, Col 1

## Explanation :

- Metrics :

**Execution Time : 6.290 ms    Total Expected Cost : 1806.34**

- Reason :

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

## 2. given query with B+ trees indices only :

pgAdmin 4

```

select count(s.sname)
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
CREATE INDEX b_sailorsSID ON sailors USING btree(sid );
CREATE INDEX b_reservesSID ON reserves USING btree(sid );
CREATE INDEX b_reservesBID ON reserves USING btree(bid);
CREATE INDEX b_boat ON boat USING btree(bid,color );
CREATE INDEX b_query_8 ON query_8 USING btree(sid );
select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_8';
DROP INDEX IF EXISTS b_sailorsSID cascade;
DROP INDEX IF EXISTS b_reservesSID cascade;
DROP INDEX IF EXISTS b_reservesBID cascade;
DROP INDEX IF EXISTS b_boat cascade;
set enable_hashagg = off;
set enable_hashjoin = off;
set enable_seqscan = on;
-- Query 8 (STATISTICS)
explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r. bid in (select b.bid
from boat b
where b.color = 'red'));
-- Query 8 view table of reserves sids that have red boats
create MATERIALIZED VIEW query_8
as
select r1.sid
from (select r.sid
from reserves r
where exists (select * from boat b where r.bid=b.bid and color='red' )) as r1 ;
explain analyze select s.sname
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);

```

Total rows: 5 of 5    Query complete 00:00:00.070

Successfully run. Total query runtime: 137 msec. 4 rows affected. Ln 206, Col 1

pgAdmin 4

```

set enable_hashagg = off;
set enable_hashjoin = off;
set enable_seqscan = on;
-- Query 8 (STATISTICS)
explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r. bid in (select b.bid
from boat b
where b.color = 'red'));
-- Query 8 optimized (STATISTICS)
-- Query 8 view table of reserves sids that have red boats
create MATERIALIZED VIEW query_8
as
select r1.sid
from (select r.sid
from reserves r
where exists (select * from boat b where r.bid=b.bid and color='red' )) as r1 ;
explain analyze select s.sname
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);

```

===== Query 9 =====

```

-- Find the names of sailors who have reserved both a red and a green boat.
-- Query 9 (COUNTING RESULT SET) , Number Of Rows = 177
select count(s.sname)
from sailors s, reserves r, boat b
where

```

Total rows: 1 of 1    Query complete 00:00:00.058

Successfully run. Total query runtime: 58 msec. 1 rows affected. Ln 244, Col 17

Successfully run. Total query runtime: 137 msec. 4 rows affected. Ln 244, Col 17

```

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

```

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

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

### Explanation :

- Metrics :

**Execution Time : 1.157 ms    Total Expected Cost :191.19**

- The B-tree helped in the performance it decreased the Execution Time and Expected Cost .
- The Query Planner used the B-tree index because B-Tree is O(Log n) performance with Exact Values .
- Here it showed the improvement due to for (s.sid=R.sid (it used the index that was built on query\_8 view)) condition of Joining in the Query the Merge Semi Join used index scan using ZigZag and algorithm and these columns where built on it an b-tree index.

- given query with hash indices only :

pgAdmin 4

Jun 9 6:19 PM ● pgAdmin 4

Servers Local PostgreSQL

Dat Query History

```

> p 196
> s 197
> s 198
> s 199
> s 200 CREATE INDEX b_sailorsSID ON sailors USING hash(sid );
> s 201 CREATE INDEX b_reservesID ON reserves USING hash(sid );
> s 202 CREATE INDEX b_reservesBID ON reserves USING hash(bid );
> s 203 CREATE INDEX b_boat1 ON boat USING hash(bid );
> s 204 CREATE INDEX b_boat2 ON boat USING hash(color );
> s 205
> s 206 CREATE INDEX b_query_8 ON query_8 USING hash(sid );
> s 207
> s 208
> s 209 select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_8';
> s 210
> s 211
> s 212
> s 213
> s 214
> s 215 DROP INDEX IF EXISTS b_sailorsSID cascade;
> s 216 DROP INDEX IF EXISTS b_reservesID cascade;
> s 217 DROP INDEX IF EXISTS b_reservesBID cascade;
> s 218 DROP INDEX IF EXISTS b_boat1 cascade;
> s 219 DROP INDEX IF EXISTS b_boat2 cascade;
> s 220
> s 221 DROP INDEX IF EXISTS b_query_8 cascade;
> s 222
> s 223 set enable_hashagg = off;
> s 224 set enable_hashjoin = off;
> s 225 set enable_seqscan = on;
> s 226 -- Query 8 (STATISTICS)
> s 227
> s 228
> s 229
> s 230
> s 231 explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red')) );
> s 232
> s 233
> s 234
> s 235
> s 236
> s 237
> s 238
> s 239
> s 240
> s 241
> s 242
> s 243
> s 244
> s 245
> s 246
> s 247
> s 248
> s 249
> s 250
> s 251
> s 252
> s 253
> s 254
> s 255
-- ====== Query 9 ======
> s 256

```

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

Data output Messages Explain Notifications

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

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

pgAdmin 4

Jun 9 6:21 PM ● pgAdmin 4

Servers Local PostgreSQL

Dat Query History

```

> p 216
> s 217
> s 218
> s 219
> s 220
> s 221
> s 222
> s 223
> s 224
set enable_hashagg = off;
> s 225
set enable_hashjoin = off;
> s 226
set enable_seqscan = on;
> s 227
-- Query 8 (STATISTICS)
> s 228
> s 229
> s 230
> s 231
explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
from reserves r
where r.bid in (select b.bid
from boat b
where b.color = 'red')) );
> s 232
> s 233
> s 234
> s 235
> s 236
> s 237
> s 238
> s 239
-- Query 8 optimized (STATISTICS)
> s 240
-- Query 8 view table of reserves sids that have red boats
> s 241
create MATERIALIZED VIEW query_8
as
> s 242
select r1.sid
from (select r.sid
      from reserves r
      where exists (select * from boat b where r.bid=b.bid and color='red' ) )
> s 243
explain analyze select s.sname
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
> s 244
> s 245
> s 246
> s 247
> s 248
> s 249
> s 250
> s 251
> s 252
> s 253
> s 254
-- ====== Query 9 ======
> s 255
> s 256

```

Graphical Analysis Statistics

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

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

The screenshot shows the pgAdmin 4 interface. On the left, there's a sidebar with various icons for database management. The main area has a tree view under 'Servers' for 'Local' and 'postgres'. A 'Query' tab is selected in the top navigation bar. The query editor contains several lines of SQL code, including `DROP INDEX` statements, `set enable\_hashagg = off;`, and `explain analyze` blocks. The results pane on the right displays the 'QUERY PLAN' and execution statistics. The status bar at the bottom indicates 'Total rows: 6 of 6' and 'Query complete 00:00:00.063'.

```

216 DROP INDEX IF EXISTS b.reserve CASCADE;
217 DROP INDEX IF EXISTS b.reserve$ID cascade;
218 DROP INDEX IF EXISTS b.boat1 cascade;
219 DROP INDEX IF EXISTS b.boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = on;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
232 from sailors s
233 where s.sid in ( select r.sid
234 from reserves r
235 where r.bid in (select b.bid
236 from boat b
237 where b.color = 'red'));
238
239 -- Query 8 optimized (STATISTICS)
240
241 -- Query 8 view table of reserves sids that have red boats
242 create MATERIALIZED VIEW query_8
243 as
244 select r1.sid
245 from (select r.sid
246 from reserves r
247 where exists (select * from boat b where r.bid=b.bid and color='red')) r1;
248
249 explain analyze select s.sname
250 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
251
252
253
254
255 -- ====== Query 9 ======
256

```

Execution Time : 11.896 ms    Total Expected Cost : 721.68

QUERY PLAN

text

1 Nested Loop Semi Join (cost=0.00..721.68 rows=673 width=21) (actual time=0.016..11.861 rows=673 loops=1)

2 -> Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.006..1.457 rows=19000 loops=1)

3 -> Index Scan using b\_query\_8 on query\_8 r1 (cost=0.00..0.04 rows=2 width=4) (actual time=0.000..0.000 rows=0 loops=19000)

4 Index Cond: (sid = s.sid)

5 Planning Time: 0.150 ms

6 Execution Time: 11.896 ms

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

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

### Explanation :

- Metrics :

**Execution Time : 11.896 ms    Total Expected Cost : 721.68**

- The Hash helped in the performance it decreased the Execution Time (but in the execution time processor was overwhelmed) and Expected Cost .
- The Query Planner used the Hash index because Hash is O(1) performance with Exact Values and considered the best for exact values queries.
- Here it showed the improvement due to for every condition of Joining in the Query the Nested Loop Semi Join using index scan using Hash based algorithm on the condition (sid(from query\_8 view)=s.sid) which approximatly maded to be O(n) performance.

- given query with BRIN indices only :

pgAdmin 4

Jun 9 6:22 PM •

Servers schema3/postgres Local PostgreSQL Data Query History

```

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

```

Data output Messages Explain Notifications

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

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

pgAdmin 4

Jun 9 6:25 PM •

Servers schema3/postgres Local PostgreSQL Data Query History

```

> p 217 DROP INDEX IF EXISTS b_reservesBID cascade;
> s 218
> s 219
> s 220
> s 221
> s 222
> s 223
> s 224 set enable_hashagg = off;
> s 225 set enable_hashjoin = off;
> s 226 set enable_seqscan = off;
> s 227 -- Query 8 (STATISTICS)
> s 228
> s 229
> s 230
> s 231
> s 232
> s 233
> s 234
> s 235
> s 236
> s 237
> s 238
> s 239
> s 240
> s 241
> s 242
> s 243
> s 244
> s 245
> s 246
> s 247
> s 248
> s 249
> s 250
> s 251
> s 252
> s 253
> s 254
> s 255
> s 256
> s 257
-- ====== Query 9 ======
-- Find the names of sailors who have reserved both a red and a green boat.
Total rows: 1 of 1 Query complete 00:00:00.952

```

Explain Analyze [Shift+F7]

Data output Messages Explain Statistics

```

graph LR
    query_8[query_8] --> Sort[Sort]
    Sort --> Unique[Unique]
    Unique --> NLLJ[Nested Loop Inner Join]
    b_sailorsSID[b_sailorsSID] --> sailors[sailors]
    sailors --> NLLJ

```

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

The screenshot shows the pgAdmin 4 interface. On the left is a sidebar with various icons for managing databases, schemas, and objects. The main window has a toolbar at the top with buttons for File, Object, Tools, and Help. Below the toolbar is a menu bar with PgAdmin, Properties, SQL, Dependents, Statistics, Dashboard, and a tab for schema3/postgres@postgres\*. The central area contains a query editor with several lines of SQL code. To the right of the query editor is a results pane titled 'Data output' which displays the 'QUERY PLAN' for the executed query. The plan details the execution steps, including nested loops, unique scans, and sort operations. At the bottom right of the results pane, there is a message indicating the query was successfully run with a total runtime of 137 msec and 4 rows affected. The status bar at the bottom of the pgAdmin window shows the date and time as Jun 9 6:25 PM.

```

217 DROP INDEX IF EXISTS b_reserve;
218 DROP INDEX IF EXISTS b_boat1 cascade;
219 DROP INDEX IF EXISTS b_boat2 cascade;
220
221 DROP INDEX IF EXISTS b_query_8 cascade;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = off;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
232 from sailors s
233 where s.sid in ( select r.sid
234 from reserves r
235 where r.bid in (select b.bid
236 from boat b
237 where b.color = 'red'));
238
239 -- Query 8 optimized (STATISTICS)
240
241 -- Query 8 view table of reserves sids that have red boats
242 create MATERIALIZED VIEW query_8
243 as
244 select r1.sid
245 from (select r.sid
246 from reserves r
247 where exists (select * from boat b where r.bid=b.bid and color='red')) as r1;
248
249 explain analyze select s.sname
250 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
251
252
253
254
255
256
257 -- ====== Query 9 ======
-- Find the names of sailors who have reserved both a red and a green boat.
Total rows: 18 of 18   Query complete 00:00:00.875

```

### Explanation :

- Metrics :

**Execution Time : 859.141 ms    Total Expected Cost : 10005517867.87**

- Here the BRIN was not used in the original Query Plan settings (Hashjoin and HashAgg are off) so I've made seqscan=off too.
- The Execution Time and Expected Cost became the Worst of all .
- This happened because the Query Optimizer didnt used it from the first place due to BRIN Usage here was not suitable so we have used it to simulate seqscan behaviour only we traversed it all and followed all its pointers so it is worst index to use in this case.

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

Activities pgAdmin 4 Jun 9 6:43 PM ● pgAdmin 4

Servers Local PostgreSQL Data Query History Execute/Refresh

```

CREATE INDEX b_sailorssid ON sailors USING hash(sid);
CREATE INDEX b_reservesID ON reserves USING btree(sid);
CREATE INDEX b_reservesBID ON reserves USING btree(bid);
CREATE INDEX b_boat1 ON boat USING btree(bid);
CREATE INDEX b_boat2 ON boat USING btree(color);
CREATE INDEX b_query_8 ON query_8 USING btree(sid);

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

DROP INDEX IF EXISTS b_sailorssid cascade;
DROP INDEX IF EXISTS b_reservesID cascade;
DROP INDEX IF EXISTS b_reservesBID cascade;
DROP INDEX IF EXISTS b_boat1 cascade;
DROP INDEX IF EXISTS b_boat2 cascade;
DROP INDEX IF EXISTS b_query_8 cascade;

set enable_hashagg = off;
set enable_hashjoin = off;
set enable_seqscan = on;
-- Query 8 (STATISTICS)

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

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

Data output Messages Explain Notifications

schemaname	tablename	indexname	tablespace	indexdef
1 public	sailors	b_sailorssid	[null]	CREATE INDEX b_sailorssid ON public.sailors USING hash(sid)
2 public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesID ON public.reserves USING btree(sid)
3 public	reserves	b_reserves...	[null]	CREATE INDEX b_reservesBID ON public.reserves USING btree(bid)
4 public	boat	b_boat1	[null]	CREATE INDEX b_boat1 ON public.boat USING btree(bid)
5 public	boat	b_boat2	[null]	CREATE INDEX b_boat2 ON public.boat USING btree(color)
6 public	query_8	b_query_8	[null]	CREATE INDEX b_query_8 ON public.query_8 USING btree(sid)

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

Activities pgAdmin 4 Jun 9 6:45 PM ● pgAdmin 4

Servers Local PostgreSQL Data Query History Execute/Refresh

```

DROP INDEX IF EXISTS b_query_8 cascade;

set enable_hashagg = off;
set enable_hashjoin = off;
set enable_seqscan = on;
-- Query 8 (STATISTICS)

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

-- Query 8 optimized (STATISTICS)

```

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

```

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

-- ====== Query 9 ======
-- Find the names of sailors who have reserved both a red and a green boat.

-- Query 9 (COUNTING RESULT SET) , Number Of Rows = 177
```

Total rows: 1 of 1    Query complete 00:00:00.053

Data output Messages Explain Notifications

Graphical Analysis Statistics

```

graph LR
    query_8 --> Sort[Sort]
    Sort --> Unique[Unique]
    Unique --> NLJoin[Nested Loop Inner Join]
    NLJoin --> b_sailorssid[b_sailorssid]

```

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

```

221 DROP INDEX IF EXISTS b_query;
222
223
224 set enable_hashagg = off;
225 set enable_hashjoin = off;
226 set enable_seqscan = on;
227 -- Query 8 (STATISTICS)
228
229
230
231 explain analyze select s.sname
232 from sailors s
233 where s.sid in ( select r.sid
234 from reserves r
235 where r.bid in (select b.bid
236 from boat b
237 where b.color = 'red'));
238
239 -- Query 8 optimized (STATISTICS)
240
241 -- Query 8 view table of reserves sids that have red boats
242 create MATERIALIZED VIEW query_8
243 as
244 select r1.sid
245 from (select r.sid
246 from reserves r
247 where exists (select * from boat b where r.bid=b.bid and color='red')) a
248
249 explain analyze select s.sname
250 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
251
252
253
254
255 -- ====== Query 9 ======
256
257 -- Find the names of sailors who have reserved both a red and a green boat.
258
259 -- Query 9 (COUNTING RESULT SET) , Number Of Rows = 177
260
261

```

Total rows: 10 of 10    Query complete 00:00:00.071

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

Ln 249, Col 1

### Explanation :

- Metrics :

**Execution Time : 1.209 ms    Total Expected Cost : 999.20**

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