

Query 7

- Find the names of sailors who have reserved boat 103.

Note !

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

Original Query

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

Result Set

- 582 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 16 11:58 PM". The central workspace shows a tree view of database objects under "Browser" (Servers, Databases, etc.) and a "Properties" tab for "schema3/postgres@postgres". Below this is a "Query" tab containing a SQL script. The script includes a `DROP INDEX` command and a `select count(*)` query for "sname" from "sailors" where "sid" is in the "reserves" table for "r.bid = 103". The output pane shows a single row with a count of 582. A message bar at the bottom right says "Successfully run. Total query runtime: 51 msec. 582 rows affected. Ln 98, Col 21".

```
82
83
84 DROP INDEX IF EXISTS reserves_pkey cascade;
85
86
87
88
89
90 -- ====== Query 7 ======
91
92 -- Query 7 COUNTING RESULT SET
93 select count(s.sname)
94 from sailors s
95 where
96 s.sid in( select r.sid
97 from reserves r
98 where r.bid = 103 );
99
100
101
```

| count | bigrint |
|-------|---------|
| 1 | 582 |

Total rows: 1 of 1 | Query complete 00:00:00.16 | ✓ Successfully run. Total query runtime: 51 msec. 582 rows affected. Ln 98, Col 21

Report

1. given query without an index :

pgAdmin 4 - pgAdmin 4

May 17 12:08 AM •

schema3/postgres@postgres*

Query History

```

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

Jun 9 2:11 PM •

pgAdmin 4

schema3/postgres@postgres*

Query History

```

124 -- Query 7 (STATISTICS)
125 set enable_hashagg = off;
126 set enable_hashjoin = off;
127
128 explain analyze select s.sname
129   from sailors s
130   where
131     s.sid in( select r.sid
132       from reserves r
133       where r.bid = 103 );
134
135
136 -- Query 7 optimized (STATISTICS)
137
138 -- view of Query 7
139 set enable_hashagg = off;
140 set enable_hashjoin = off;
141
142 create MATERIALIZED VIEW query_7
143 as
144   select r.sid
145     from reserves r
146     where r.bid =103 ;
147
148
149 explain analyze select s.sname
150   from sailors s
151   where exists (select R.sid
152                 from query_7 R
153                 where s.sid =R.sid)
154
155
156
157
158
159 == ====== Query 8 ======
160
161 -- Find the names of sailors who ha'ue reserved a red boat.
162 -- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673
163
164 explain analyze select count(s.sname)
165

```

Explain Analyze [Shift+F7]

Graphical Analysis Statistics

```

graph LR
    sailors[sailors] --> Sort1[Sort]
    Sort1 --> Merge[Merge Semi Join]
    reserves[reserves] --> Sort2[Sort]
    Sort2 --> Merge

```

Total rows: 1 of 1 Query complete 00:00:00.243

Ln 128, Col 16

```

-- Query 7 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;

explain analyze select s.sname
from sailors s
where
  s.sid in( select r.sid
  from reserves r
  where r.bid = 103 );

-- Query 7 optimized (STATISTICS)
-- view of Query 7
set enable_hashagg = off;
set enable_hashjoin = off;

create MATERIALIZED VIEW query_7
as
select r.sid
from reserves r
where r.bid =103 ;

explain analyze select s.sname
from sailors s
where exists (select R.sid
               from query_7 R
               where s.sid =R.sid);

-- ====== Query 8 ======
-- Find the names of sailors who ha'ue reserved a red boat.
-- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673
explain analyze select count(s.sname);

```

Total rows: 14 of 14 Query complete 0:00:00.146 Ln 125, Col 1

Explanation :

- Metrics :

Execution Time : 15.430 ms Total Expected Cost : 2457.26

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

2. given query with B+ trees indices only :

Activities pgAdmin 4 Jun 9 3:12 PM ● pgAdmin 4

Servers Local | postgres Data Query History Execute/Refresh (F5)

```

-- ====== Query 7 ======
-- Find the names of sailors who have reserved boat 103.
-- Query 7 COUNTING RESULT SET , Number Of Rows = 582

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 R_reservesBID ON query_7 USING btree(sid );

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_7';

select count(s.sname)
from sailors s
where
s.sid in( select r.sid
from reserves r
where r.bid = 103 );

-- Query 7 optimized (COUNTING RESULT SET) , Number Of Rows = 582

select count(s.sname)
from sailors s
where exists (select R.sid
from query_7 R
where s.sid =R.sid)

Total rows: 4 of 4   Query complete 00:00:00.141

```

Ln 95, Col 1

Activities pgAdmin 4 Jun 9 2:57 PM ● pgAdmin 4

Servers Local | postgres Data Query Explain Analyze (Shift+F7)

```

-- Query 7 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;

explain analyze select s.sname
from sailors s
where
s.sid in( select r.sid
from reserves r
where r.bid = 103 );

-- Query 7 optimized (STATISTICS)
-- view of Query 7
set enable_hashagg = off;
set enable_hashjoin = off;

-- Query 7 view table of reserves with bid =103
create MATERIALIZED VIEW query_7
as
select r.sid
from reserves r
where r.bid =103 ;

explain analyze select s.sname
from sailors s
where exists (select R.sid
from query_7 R
where s.sid =R.sid);

-- ====== Query 8 ======
-- Find the names of sailors 'who ha'ue reserved a red boat.

Total rows: 1 of 1   Query complete 00:00:00.055

```

Successfully run. Total query runtime: 55 msec. 1 rows aff
Ln 132, Col 17

The screenshot shows the pgAdmin 4 interface with a query editor. The left pane displays a tree view of database objects under 'schema3/postgres@postgres'. The main query editor window contains a multi-line SQL script. The right pane shows the 'Data output' tab, which displays the 'QUERY PLAN' for the last executed query. The plan details the execution steps, including a Merge Semi Join, Sort, and Bitmap Index Scan.

```
-- Query 7 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;
explain analyze select s.sname
from sailors s
where
s.sid in (select r.sid
from reserves r
where r.bid = 103 );
-- Query 7 optimized (STATISTICS)
view of Query 7
Tab 143
set enable_hashagg = off;
set enable_hashjoin = off;
-- Query 7 view table of reserves with bid =103
create MATERIALIZED VIEW query_7
as
select r.sid
from reserves r
where r.bid =103 ;
-- explain analyze select s.sname
-- from sailors s
-- where exists (select R.sid
--                 from query_7 R
--                 where s.sid =R.sid);
-- ====== Query 8 ======
-- Find the names of sailors who ha'ue reserved a red boat.
Total rows: 13 of 13   Query complete 00:00:00.080
```

Jun 9 2:57 PM • pgAdmin 4

Explanation :

- Metrics:

Execution Time : 0.449 ms Total Expected Cost : 956.28

- The B-tree helped in the performance it decreased the Execution Time and Expected Cost (BEST OF THEM ALL).
 - The Query Planner used the B-tree index because B-Tree is $O(\log n)$ performance with Exact Values .
 - Here it showed the improvement due to for every condition of Joining in the Query the Merge Join used index scan using ZigZag algorithm .
 - The Where clause condition on the ($bid = 103$) used B-tree which is more better as ($bid = 103$) are repeatable and sorted after each other at the leaves of the b-tree with $O(\log n)$ performance on all at one time .

3. given query with hash indices only :

Activities pgAdmin 4 Jun 9 3:32 PM ● pgAdmin 4

Servers Local PostgreSQL Data Query History

```

83 DROP INDEX IF EXISTS reserves_pkey cascade;
84
85
86
87
88
89
90 -- ====== Query 7 ======
91
92 -- Find the names of sailors who have reserved boat 103.
93 -- Query 7 COUNTING RESULT SET , Number Of Rows = 582
94
95
96 CREATE INDEX b_sailorsSID ON sailors USING hash(sid);
97 CREATE INDEX b_reservesSID ON reserves USING hash(reserveid);
98 CREATE INDEX b_reservesBID ON reserves USING hash(bid);
99 CREATE INDEX R_reservesBID ON query_7 USING hash(reserveid);

100 select *
101 from pg_indexes
102 where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tab
103
104
105
106
107
108
109 select count(s.sname)
110 from sailors s
111 where
112 s.sid in( select r.sid
113 from reserves r
114 where r.bid = 103 );
115
116
117
118 -- Query 7 optimized (COUNTING RESULT SET) , Number Of Rows = 582
119
120
121 select count(s.sname)
122 from sailors s
123 where exists (select R.sid
124 from query_7 R
125 where s.sid=R.sid)

Total rows: 4 of 4 Query complete 00:00:00.207

```

Data output Messages Explain Notifications

| schemaname | tablename | indexname | tablespace | indexdef |
|------------|-----------|---------------|------------|--|
| public | sailors | b_sailorsSID | [null] | CREATE INDEX b_sailorsSID ON public.sailors USING hash (sid) |
| public | reserves | b_reservesSID | [null] | CREATE INDEX b_reservesSID ON public.reserves USING hash (reserveid) |
| public | reserves | b_reservesBID | [null] | CREATE INDEX b_reservesBID ON public.reserves USING hash (bid) |
| public | query_7 | r_reservesBID | [null] | CREATE INDEX R_reservesBID ON public.query_7 USING hash (reserveid) |

Activities pgAdmin 4 Jun 9 3:36 PM ● pgAdmin 4

Servers Local PostgreSQL Data Query History Explain Analyze

```

118 -- Query 7 optimized (COUNTING RESULT SET)
119
120
121
122 select count(s.sname)
123 from sailors s
124 where exists (select R.sid
125 from query_7 R
126 where s.sid=R.sid)

127
128
129
130 -- Query 7 (STATISTICS)
131 set enable_hashagg = off;
132 set enable_hashjoin = off;
133
134 explain analyze select s.sname
135 from sailors s
136 where
137 s.sid in( select r.sid
138 from reserves r
139 where r.bid = 103 );
140
141
142 -- Query 7 optimized (STATISTICS)
143
144 -- view of Query 7
145 set enable_hashagg = off;
146 set enable_hashjoin = off;
147
148 -- Query 7 view table of reserves with bid =103
149
150 create MATERIALIZED VIEW query_7
151 as
152 select r.sid
153 from reserves r
154 where r.bid =103 ;
155
156
157
158 explain analyze select s.sname

```

Data output Messages Explain Notifications

Graphical Analysis Statistics

```

graph LR
    A[b_reservesBID] --> B[reserves]
    B -- Sort --> C[Unique]
    C --> D[Nested Loop Inner Join]
    E[b_sailorssid] -.-> D

```

Total rows: 1 of 1 Query complete 00:00:00.051

```

-- Query 7 optimized (COUNTING)
-- Number Of Rows = 582

121 select count(s.sname)
122   from sailors s
123 where exists (select R.sid
124                  from query_7 R
125                 where s.sid = R.sid)

-- Query 7 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;

explain analyze select s.sname
from sailors s
where
  s.sid in( select r.sid
            from reserves r
            where r.bid = 103 );
140
141
142 -- Query 7 optimized (STATISTICS)
143
144 -- view of Query 7
set enable_hashagg = off;
set enable_hashjoin = off;
147
148 -- Query 7 view table of reserves with bid =103
149
150 create MATERIALIZED VIEW query_7
151 as
152 select r.sid
153   from reserves r
154  where r.bid =103 ;
155
156
157
158 explain analyze select s.sname
Total rows: 14 of 14   Query complete 00:00:00.089

```

Successfully run. Total query runtime: 89 msec. 14 rows affected
Ln 131, Col 1

Explanation :

- Metrics :

Execution Time : 0.993 ms Total Expected Cost : 1159.32

- 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 (sid=s.sid) which approximatly maded to be O(n) performance .
- Here it showed the improvement due to for condition of (bid = 103) with performance of O(1) .

4. given query with BRIN indices only :

Activities pgAdmin 4 Jun 9 4:32 PM ● pgAdmin 4

Dat Query History Execute/Refresh

```

> p 84 DROP INDEX IF EXISTS reserves;
> s 85
> s 86
> s 87
> s 88
> s 89
> s 90 -- -----
> s 91 -- Find the names of sailors who have reserved boat 103.
> s 92 -- Query 7 COUNTING RESULT SET , Number Of Rows = 582
> s 93
> s 94
> s 95
> s 96 CREATE INDEX b_sailorsSID ON sailors USING BRIN(sid);
> s 97 CREATE INDEX b_reservesSID ON reserves USING BRIN(sid);
> s 98 CREATE INDEX b_reservesBID ON reserves USING BRIN(bid);
> s 99 CREATE INDEX R_reservesBID ON query_7 USING BRIN(sid);

> Log 100
> Tab 102
> postgr 103 select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_7';

104
105
106
107
108 DROP INDEX IF EXISTS b_sailorsSID cascade;
109 DROP INDEX IF EXISTS b_reservesSID cascade;
110 DROP INDEX IF EXISTS b_reservesBID cascade;
111 DROP INDEX IF EXISTS R_reservesBID cascade;
112
113
114
115
116 select count(s.sname)
from sailors s
117 where
118 s.sid in( select r.sid
119 from reserves r
120 where r.bid = 103 );
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159

```

Total rows: 4 of 4 Query complete 00:00:00.070 Ln 96, Col 1

Activities pgAdmin 4 Jun 9 5:07 PM ● pgAdmin 4

Dat Query History Explain Analyze Shift [F7]

```

> p 119 s.sid in( select r.sid
> s 120 from reserves r
> s 121 where r.bid = 103 );
> s 122
> s 123
> s 124
> s 125 -- Query 7 optimized (COUNTING RESULT SET) , Number Of Rows = 582
> s 126
> s 127
> s 128 select count(s.sname)
from sailors s
129 where exists (select R.sid
130 from query_7 R
131 where s.sid =R.sid);
132
133
134
135
136
137 -- Query 7 (STATISTICS)
138 set enable_hashagg = off;
139 set enable_hashjoin = off;
140 set enable_seqscan = off;
141
142
143 explain analyze select s.sname
from sailors s
144 where
145 s.sid in( select r.sid
146 from reserves r
147 where r.bid = 103 );
148
149
150
151 -- Query 7 optimized (STATISTICS)
152
153 -- view of Query 7
154 set enable_hashagg = off;
155 set enable_hashjoin = off;
156
157 -- Query 7 view table of reserves with bid =103
158
159 create MATERIALIZED VIEW query_7

```

Total rows: 1 of 1 Query complete 00:00:00.786 Ln 149, Col 1

Data output Messages Explain Notifications

Graphical Analysis Statistics

Successfully run. Total query runtime: 786 msec. 1 rows affected.

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

```

119 s.sid in( select r.sid
120   from reserves r
121   where r.bid = 103 );
122
123
124
125 -- Query 7 optimized (COUNTING RESULT SET) , Number Of Rows = 582
126
127
128 select count(s.sname)
129   from sailors s
130   where exists (select R.sid
131                  from query_7 R
132                 where s.sid =R.sid)
133
134
135
136 -- Query 7 (STATISTICS)
137 set enable_hashagg = off;
138 set enable_hashjoin = off;
139 set enable_seqscan = off;
140
141
142
143 explain analyze select s.sname
144   from sailors s
145   where
146     s.sid in( select r.sid
147       from reserves r
148      where r.bid = 103 );
149
150
151 -- Query 7 optimized (STATISTICS)
152
153 -- view of Query 7
154 set enable_hashagg = off;
155 set enable_hashjoin = off;
156
157 -- Query 7 view table of reserves with bid =103
158
159 create MATERIALIZED VIEW query_7
Total rows: 23 of 23   Query complete 0:00:00.865

```

Successfully run. Total query runtime: 137 msec. 4 rows affected
Ln 149, Col 1

Explanation :

- Metrics :

Execution Time : 851.120 ms Total Expected Cost : 4080651.27

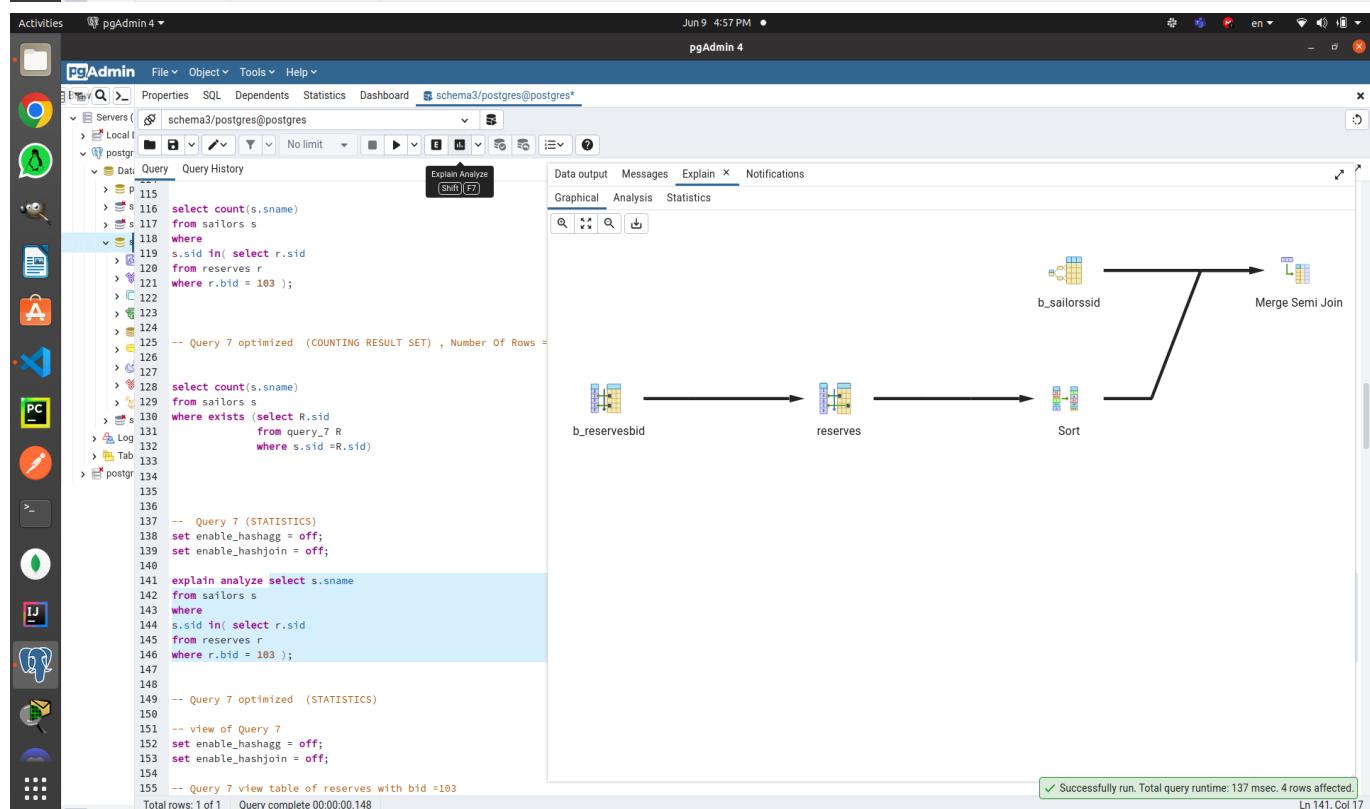
- Here the BRIN was not used in the original Query Plan settings (Hashjoin and HashAgg are off) so I've made seqscan=off too.
- The Execution Time and Expected Cost became the (WORST OF THEM ALL) .
- This happened because the Query Optimizer didn't use it from the first place due to BRIN Usage here was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

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

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

- Top Bar:** Activities, pgAdmin 4, Jun 9 4:56 PM, en.
- Servers:** Localhost (localhost), schema3/postgres@postgres
- Data Tab:** Shows a list of queries and their execution status. The current query is "DROP INDEX IF EXISTS reserves;".
- Query History:** Displays the history of executed queries, including:
 - Line 84: DROP INDEX IF EXISTS reserves;
 - Line 85: -- Find the names of sailors who have reserved boat 103.
 - Line 86: -- Query 7 COUNTING RESULT SET , Number Of Rows = 582
 - Line 87: CREATE INDEX b_sailorsSID ON sailors USING btree(sid);
 - Line 88: CREATE INDEX b_reservesSID ON reserves USING hash(sid);
 - Line 89: CREATE INDEX b_reservesBID ON reserves USING hash(bid);
 - Line 90: CREATE INDEX R_reservesBID ON query_7 USING btree(sid);
 - Line 91: select * from pg_indexes where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_7';
 - Line 92: DROP INDEX IF EXISTS b_sailorsSID cascade;
 - Line 93: DROP INDEX IF EXISTS b_reservesSID cascade;
 - Line 94: DROP INDEX IF EXISTS b_reservesBID cascade;
 - Line 95: DROP INDEX IF EXISTS R_reservesBID cascade;
 - Line 96: select count(s.sname)
 - Line 97: from sailors s
 - Line 98: where
 - Line 99: s.sid in (select r.sid
 - Line 100: from reserves r
 - Line 101: where r.bid = 103);
- Output Tab:** Data output, Messages, Explain, Notifications. It shows the results of the last query:

| schemaname | tablename | indexname | tablespace | indexdef |
|------------|-----------|---------------|------------|--|
| public | sailors | b_sailorsSID | [null] | CREATE INDEX b_sailorsSID ON public.sailors USING btree (sid) |
| public | reserves | b_reservesSID | [null] | CREATE INDEX b_reservesSID ON public.reserves USING hash (sid) |
| public | reserves | b_reservesBID | [null] | CREATE INDEX b_reservesBID ON public.reserves USING hash (bid) |
| public | query_7 | R_reservesBID | [null] | CREATE INDEX R_reservesBID ON public.query_7 USING btree (sid) |
- Status Bar:** Total rows: 4 of 4, Query complete 00:00:00.056, Jun 9 4:56 PM, Line 95, Col 1. A green checkmark icon indicates success.



```

115
116 select count(s.sname)
117 from sailors s
118 where
119   s.sid in( select r.sid
120   from reserves r
121   where r.bid = 103 );
122
123
124
125
126
127
128 select count(s.sname)
129 from sailors s
130 where exists (select R.sid
131   from query_7 R
132   where s.sid =R.sid);
133
134
135
136
137 -- Query 7 (STATISTICS)
138 set enable_hashagg = off;
139 set enable_hashjoin = off;
140
141 explain analyze select s.sname
142 from sailors s
143 where
144   s.sid in( select r.sid
145   from reserves r
146   where r.bid = 103 );
147
148
149 -- Query 7 optimized (STATISTICS)
150
151 -- view of Query 7
152 set enable_hashagg = off;
153 set enable_hashjoin = off;
154
155 -- Query 7 view table of reserves with bid =103
Total rows: 13 of 13   Query complete 00:00:00.174

```

Successfully run. Total query runtime: 137 msec. 4 rows affected
Ln 138, Col 1

Explanation :

- Metrics :

Execution Time : 0.845 ms Total Expected Cost : 960.03

- 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 $O(n \log n)$.
- And It used the Hash indexed scan on bid =103.
- The Mix indices helped in the performance it decreased the Execution Time and Expected Cost .

Optimized Query

```

-- Query 7 view table of reserves with bid = 103

create MATERIALIZED VIEW query_7
as
select r.sid
from reserves r
where r.bid =103  ;

select s.sname
from sailors s
where exists (select R.sid
              from query_7 R
              where s.sid =R.sid);

```

Result Set

- 582 Rows

Report

1. given query without an index :

pgAdmin 4

```

Activities pgAdmin 4 Jun 9 2:22 PM ●
File Object Tools Help
Servers Local PostgreSQL schema3/postgres*
Data Query History Explain Analyze Shift [F7]
124 -- Query 7 (STATISTICS)
125 set enable_hashagg = off;
126 set enable_hashjoin = off;
127
128 explain analyze select s.sname
129   from sailors s
130   where
131     s.sid in( select r.sid
132       from reserves r
133       where r.bid = 103 );
134
135
136 -- Query 7 optimized (STATISTICS)
137
138 -- view of Query 7
139 set enable_hashagg = off;
140 set enable_hashjoin = off;
141
142 -- Query 7 view table of reserves with bid =103
143 create MATERIALIZED VIEW query_7
144 as
145   select r.sid
146   from reserves r
147   where r.bid =103 ;
148
149
150 explain analyze select s.sname
151   from sailors s
152   where exists (select R.sid
153     from query_7 R
154     where s.sid=R.sid);
155
156
157
158
159
160 -- ====== Query 8 ======
161
162 -- Find the names of sailors 'who ha'ue reserved a red boat.
163 -- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673
164
Total rows: 1 of 1 Query complete 00:00:00.103

```

Successfully run. Total query runtime: 103 msec. 1 rows affected.

Ln 150, Col 17

pgAdmin 4

```

Activities pgAdmin 4 Jun 9 2:22 PM ●
File Object Tools Help
Servers Local PostgreSQL schema3/postgres*
Data Query History Execute/Refresh F5
124 -- Query 7 (STATISTICS)
125 set enable_hashagg = off;
126 set enable_hashjoin = off;
127
128 explain analyze select s.sname
129   from sailors s
130   where
131     s.sid in( select r.sid
132       from reserves r
133       where r.bid = 103 );
134
135
136 -- Query 7 optimized (STATISTICS)
137
138 -- view of Query 7
139 set enable_hashagg = off;
140 set enable_hashjoin = off;
141
142 -- Query 7 view table of reserves with bid =103
143 create MATERIALIZED VIEW query_7
144 as
145   select r.sid
146   from reserves r
147   where r.bid =103 ;
148
149
150 explain analyze select s.sname
151   from sailors s
152   where exists (select R.sid
153     from query_7 R
154     where s.sid=R.sid);
155
156
157
158
159
160 -- ====== Query 8 ======
161
162 -- Find the names of sailors 'who ha'ue reserved a red boat.
163 -- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673
164
Total rows: 12 of 12 Query complete 00:00:00.068

```

Ln 149, Col 1

Explanation :

- Metrics :

Execution Time : 8.309 ms Total Expected Cost : 1746.50

- Reason :

- This Query Improved in the Execution time and Expected Cost than the Original Query from 2475.26 to 1746.50 .
- 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 r.bid =103) 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 so it helped in the intermediate results.

2. given query with B+ trees indices only :

pgAdmin 4

```

Activities pgAdmin 4 Jun 9 3:12 PM ●
File Object Tools Help
Servers Local schema3/postgres@postgres Properties SQL Dependents Statistics Dashboard
postgr Data Query History Execute/Refresh (F5)
> P 89
> S 90 -- ====== Query 7 ======
> S 91
> S 92 -- Find the names of sailors who have reserved boat 103.
> S 93 -- Query 7 COUNTING RESULT SET , Number Of Rows = 582
> S 94
> S 95
> S 96 CREATE INDEX b_sailorsSID ON sailors USING btree(sid );
> S 97 CREATE INDEX b_reservesSID ON reserves USING btree(sid );
> S 98 CREATE INDEX b_reservesBID ON reserves USING btree(bid);
> S 99 CREATE INDEX R_reservesBID ON query_7 USING btree(sid );
> S 100
> S 101
> S 102 select *
> S 103 from pg_indexes
> S 104 where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_7';
> S 105
> Log 106
> Tab 107
> postgr 108
select count(s.sname)
from sailors s
where
s.sid in( select r.sid
from reserves r
where r.bid = 103 );
115
116
117
-- Query 7 optimized (COUNTING RESULT SET) , Number Of Rows = 582
119
120
121 select count(s.sname)
122 from sailors s
123 where exists (select R.sid
124 from query_7 R
125 where s.sid =R.sid)
126
127
128
129
Total rows: 4 of 4 Query complete 00:00:00.141

```

Jun 9 3:13 PM ● pgAdmin 4

| Indexdef | text |
|----------|---|
| 1 | CREATE INDEX b_sailorssid ON public.sailors USING btree (sid) |
| 2 | CREATE INDEX b_reservesSID ON public.reserves USING btree (sid) |
| 3 | CREATE INDEX b_reservesBID ON public.reserves USING btree (bid) |
| 4 | CREATE INDEX R_reservesBID ON public.query_7 USING btree (sid) |

pgAdmin 4

```

Activities pgAdmin 4 Jun 9 3:13 PM ●
File Object Tools Help
Servers Local schema3/postgres@postgres Properties SQL Dependents Statistics Dashboard
postgr Data Query History Explain Analyze (Shift [F7])
> P 130 -- Query 7 (STATISTICS)
> S 131 set enable_hashagg = off;
> S 132 set enable_hashjoin = off;
> S 133
> S 134 explain analyze select s.sname
from sailors s
where
s.sid in( select r.sid
from reserves r
where r.bid = 103 );
140
141
142 -- Query 7 optimized (STATISTICS)
143
144 -- view of Query 7
145 set enable_hashagg = off;
146 set enable_hashjoin = off;
> Log 147
> Tab 148 -- Query 7 view table of reserves with bid =103
> postgr 149
create MATERIALIZED VIEW query_7
as
select r.sid
from reserves r
where r.bid =103 ;
155
156
157
158 explain analyze select s.sname
from sailors s
where exists (select R.sid
from query_7 R
where s.sid =R.sid);
163
164
165
166
167 -- ====== Query 8 ======
168
169 -- Find the names of sailors 'who ha'ue reserved a red boat.
170 -- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673
178
Total rows: 1 of 1 Query complete 00:00:00.088

```

Jun 9 3:13 PM ● pgAdmin 4

```

graph TD
    A[b_sailorssid] --> B[Merge Semi Join]
    C[r_reservesbid] --> B

```

The screenshot shows the pgAdmin 4 interface. On the left, there's a sidebar with various icons. The main area has a title bar "pgAdmin 4" and a menu bar with "File", "Object", "Tools", "Help". Below the menu is a toolbar with icons for search, properties, SQL, dependents, statistics, dashboard, and a connection named "schema3/postgres@postgres". The central pane shows a "Query History" tab with several queries numbered 130 to 178. Query 133 is selected and expanded, showing its source code:

```

> 130 -- Query 7 (STATISTICS)
> 131 set enable_hashagg = off;
> 132 set enable_hashjoin = off;
> 133
> 134 explain analyze select s.sname
> 135   from sailors s
> 136   where
> 137     s.sid in( select r.sid
> 138       from reserves r
> 139       where r.bid = 103 );
> 140
> 141
> 142 -- Query 7 optimized (STATISTICS)
> 143
> 144 -- view of Query 7
> 145 set enable_hashagg = off;
> 146 set enable_hashjoin = off;
> 147
> 148 -- Query 7 view table of reserves with bid =103
> 149
> 150 create MATERIALIZED VIEW query_7
> 151 as
> 152 select r.sid
> 153   from reserves r
> 154   where r.bid =103 ;
> 155
> 156
> 157
> 158 explain analyze select s.sname
> 159   from sailors s
> 160   where exists (select R.sid
> 161     from query_7 R
> 162     where s.sid =R.sid);
> 163
> 164
> 165
> 166
> 167 -- ====== Query 8 ======
> 168
> 169 -- Find the names of sailors 'who ha'ue reserved a red boat.
> 170 -- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673

```

The status bar at the bottom says "Total rows: 7 of 7 Query complete 00:00:00.104". To the right of the query history, there's a "Data output" tab showing the "QUERY PLAN" for the selected query. The plan details the execution steps:

| | QUERY PLAN |
|---|---|
| 1 | Merge Semi Join (cost=0.60..60.37 rows=582 width=21) (actual time=0.019..0.354 rows=582 loops=1) |
| 2 | 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..0.093 rows=584 ..) |
| 4 | -> Index Only Scan using r_reservesbid on query_7 r (cost=0.28..31.00 rows=582 width=4) (actual time=0.011..0.129 rows=5..) |
| 5 | Heap Fetches: 582 |
| 6 | Planning Time: 0.154 ms |
| 7 | Execution Time: 0.386 ms |

Explanation :

- Metrics :

Execution Time : 0.386 ms Total Expected Cost : 60.37

- The B-tree helped in the performance it decreased the Execution Time and Expected Cost (BEST OF THEM ALL).
- The Query Planner used the B-tree index because B-Tree is $O(\log n)$ performance with Exact Values .
- Here it showed the improvement due to for (s.sid=R.sid (it used the index that was built on query_7 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 :

Activities pgAdmin 4 Jun 9 3:32 PM ● pgAdmin 4

Servers Local PostgreSQL Data Query History

```

83 DROP INDEX IF EXISTS reserves_pkey cascade;
84
85
86
87
88
89
90 -- ====== Query 7 ======
91
92 -- Find the names of sailors who have reserved boat 103.
93 -- Query 7 COUNTING RESULT SET , Number Of Rows = 582
94
95
96 CREATE INDEX b_sailorsSID ON sailors USING hash(sid);
97 CREATE INDEX b_reservesSID ON reserves USING hash(reserveid);
98 CREATE INDEX b_reservesBID ON reserves USING hash(bid);
99 CREATE INDEX R_reservesBID ON query_7 USING hash(reserveid);

100 select *
101 from pg_indexes
102 where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tab
103
104
105
106
107
108
109 select count(s.sname)
110 from sailors s
111 where
112 s.sid in (select r.sid
113 from reserves r
114 where r.bid = 103 );
115
116
117
118 -- Query 7 optimized (COUNTING RESULT SET) , Number Of Rows = 582
119
120
121 select count(s.sname)
122 from sailors s
123 where exists (select R.sid
124
Total rows: 4 of 4 Query complete 00:00:00.207

```

Data output Messages Explain Notifications

| schemaname | tablename | indexname | tablespace | indexdef |
|------------|-----------|---------------|------------|--|
| public | sailors | b_sailorsSID | [null] | CREATE INDEX b_sailorsSID ON public.sailors USING hash (sid) |
| public | reserves | b_reservesSID | [null] | CREATE INDEX b_reservesSID ON public.reserves USING hash (reserveid) |
| public | reserves | b_reservesBID | [null] | CREATE INDEX b_reservesBID ON public.reserves USING hash (bid) |
| public | query_7 | R_reservesBID | [null] | CREATE INDEX R_reservesBID ON public.query_7 USING hash (reserveid) |

Ln 95, Col 1

Activities pgAdmin 4 Jun 9 4:01 PM ● pgAdmin 4

Servers Local PostgreSQL Data Query History

```

150 create MATERIALIZED VIEW query_7
151 as
152 select r.sid
153 from reserves r
154 where r.bid=103 ;
155
156
157
158 explain analyze select s.sname
159 from sailors s
160 where exists (select R.sid
161 from query_7 R
162 where s.sid=R.sid);
163
164
165
166
167 -- ====== Query 8 ======
168
169 -- Find the names of sailors 'who ha've reserved a r
170 -- Query 8 (COUNTING RESULT SET) , Number Of Rows =
171
172 explain analyze select count(s.sname)
173 from sailors s
174 where s.sid in ( select r.sid
175 from reserves r
176 where r.bid in (select b.bid
177 from boat b
178 where b.color = 'red'));
179
180 -- Query 8 optimized (COUNTING RESULT SET) , Number
181
182
183 select count(s.sname)
184 from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);
185
186
187
188
189
190
Total rows: 1 of 1 Query complete 00:00:00.068

```

Explain Analyze (Shift+F7) Data output Messages Explain Notifications

Graphical Analysis Statistics

```

graph TD
    sailors[sailors] -->|Nested Loop Semi Join| r_reservesbid[r_reservesbid]

```

Ln 158, Col 17

The screenshot shows the pgAdmin 4 interface. On the left is a sidebar with various icons for database management. The main area has a title bar "pgAdmin 4" and a status bar at the bottom indicating "Jun 9 4:01 PM". The central part contains a query editor with several lines of SQL code. To the right of the query editor is a "Data output" tab which is currently inactive. Below the query editor is a "QUERY PLAN" section with a "text" tab selected, displaying the execution plan for the query. The plan includes details like cost, rows, and loops. At the bottom of the query editor, it says "Total rows: 6 of 6 Query complete 00:00:00.049" and "Ln 157, Col 1".

```

create MATERIALIZED VIEW query_7
as
select r.sid
from reserves r
where r.bid =103 ;
explain analyze select s.sname
from sailors s
where exists (select R.sid
from query_7 R
where s.sid =R.sid);
-- Find the names of sailors 'who ha'ue reserved a
-- Query 8 (COUNTING RESULT SET) , Number Of Rows =
explain analyze select count(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 (COUNTING RESULT SET) , Number
-- select count(s.sname)
from sailors s where exists (select * from query_8 r1 where s.sid=r1.sid);

```

Explanation :

- Metrics :

Execution Time : 11.828 ms Total Expected Cost : 715.32

- 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_7 view)=s.sid) which approximatly maded to be O(n) performance.

- given query with BRIN indices only :

Activities pgAdmin 4 Jun 9 4:32 PM ● pgAdmin 4

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

```

84 DROP INDEX IF EXISTS reserve;
85
86
87
88
89
90 -- ====== Query 7 ======
91
92 -- Find the names of sailors who have reserved boat 103.
93 -- Query 7 COUNTING RESULT SET , Number Of Rows = 582
94
95
96 CREATE INDEX b_sailorsSID ON sailors USING BRIN(sid);
97 CREATE INDEX b_reservesSID ON reserves USING BRIN(sid);
98 CREATE INDEX b_reservesBID ON reserves USING BRIN(bid);
99 CREATE INDEX R_reservesBID ON query_7 USING BRIN(bid);

100
101 select *
102 from pg_indexes
103 where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_7';
104
105
106
107
108 DROP INDEX IF EXISTS b_sailorsSID cascade;
109 DROP INDEX IF EXISTS b_reservesSID cascade;
110 DROP INDEX IF EXISTS b_reservesBID cascade;
111 DROP INDEX IF EXISTS R_reservesBID cascade;
112
113
114
115
116 select count(s.sname)
117 from sailors s
118 where
119 s.sid in( select r.sid
120 from reserves r
121 where r.bid = 103 );
122
123
124

```

Total rows: 4 of 4 Query complete 00:00:00.070 Ln 96, Col 1

Activities pgAdmin 4 Jun 9 5:09 PM ● pgAdmin 4

Servers schema3/postgres Local [Data Query History Explain analyze Shift (F7)]

```

143 explain analyze select s.sname
144 from sailors s
145 where
146 s.sid in( select r.sid
147 from reserves r
148 where r.bid = 103 );
149
150
151
152
153 -- view of Query 7
154 set enable_hashagg = off;
155 set enable_hashjoin = off;
156
157
158
159 create MATERIALIZED VIEW query_7
160 as
161 select r.sid
162 from reserves r
163 where r.bid =103 ;
164
165
166
167 explain analyze select s.sname
168 from sailors s
169 where exists (select R.sid
170 from query_7 R
171 where s.sid =R.sid);
172
173
174
175
176 -- ====== Query 8 ======
177
178 -- Find the names of sailors 'who ha've reserved a red boat.
179 -- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673
180
181 explain analyze select count(s.sname)
182 from sailors s
183 where s.sid in ( select r.sid
184 from reserves r

```

Explain Analyze Results:

```

graph LR
    A[query_7] --> B[Sort]
    B --> C[Unique]
    C --> D[Nested Loop Inner Join]
    E[b_sailorsSID] --> F[sailors]

```

Messages:

- Successfully run. Total query runtime: 769 msec. 1 rows affected.
- Successfully run. Total query runtime: 137 msec. 4 rows affected.

Total rows: 1 of 1 Query complete 00:00:00.769 Ln 167, Col 16

```

explain analyze select s.sname
from sailors s
where
  s.sid in( select r.sid
  from reserves r
  where r.bid = 103 );
-- Query 7 optimized (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;
-- view of Query 7
create MATERIALIZED VIEW query_7
as
select r.sid
from reserves r
where r.bid =103 ;
164
165
166
167 explain analyze select s.sname
168 from sailors s
169 where exists (select R.sid
170   from query_7 R
171   where s.sid =R.sid);
172
173
174
175
176 -- ====== Query 8
177
178 -- Find the names of sailors 'who ha'ue reserved a red boat.
179 -- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673
180
181 explain analyze select count(s.sname)
182 from sailors s
183 where s.sid in ( select r.sid
184   from reserves r
Total rows: 18 of 18 Query complete 00:00:00.834

```

Successfully run. Total query runtime: 137 msec. 4 rows affected.
Ln 167, Col 1

Explanation :

- Metrics :

Execution Time : 816.981 ms Total Expected Cost : 10004136191.55

- Here the BRIN was not used in the original Query Plan settings (Hashjoin and HashAgg are off) so I've made seqscan=off too.
- The Execution Time and Expected Cost became the (WORST OF THEM ALL).
- This happened because the Query Optimizer 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.
- Because there were no Aggregation used and was not low selectivity Query so it was not helpful to the performance.
- The plan performs a (Bit map index scan) on all indices and then to (Bit map heap scan) to select relevant rows .

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

pgAdmin 4

Jun 9 4:56 PM • pgAdmin 4

Activities

File Object Tools Help

Servers Local PostgreSQL

Data Query History

```

> p 84 DROP INDEX IF EXISTS reserve
> s 85
> t 86
> r 87
> q 88
> e 89
> v 90
-- ====== Query 7 ======
> n 91
> c 92
-- Find the names of sailors who have reserved boat 103.
-- Query 7 COUNTING RESULT SET , Number Of Rows = 582
> h 93
> i 94
> o 95
> n 96 CREATE INDEX b_sailorsSID ON sailors USING btree(sid );
> n 97 CREATE INDEX b_reservesID ON reserves USING hash(sid );
> n 98 CREATE INDEX b_reservesBID ON reserves USING hash(bid );
> n 99 CREATE INDEX R_reservesBID ON query_7 USING btree(sid );
> Log 100
> Tab 101
> pgsql 102 select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_7';
103
104
105
106
107
108 DROP INDEX IF EXISTS b_sailorsSID cascade;
109 DROP INDEX IF EXISTS b_reservesID cascade;
110 DROP INDEX IF EXISTS b_reservesBID cascade;
111 DROP INDEX IF EXISTS R_reservesBID cascade;
112
113
114
115
116 select count(s.sname)
from sailors s
where
s.sid in( select r.sid
from reserves r
where r.bid = 103 );
117
118
119
120
121
122
123
124
--
```

Total rows: 4 of 4 Query complete 00:00:00.056

Data output Messages Explain Notifications

| schemaName | tableName | indexName | tableSpace | indexDef |
|------------|-----------|---------------|------------|--|
| public | sailors | b_sailorsSID | [null] | CREATE INDEX b_sailorsSID ON public.sailors USING btree (sid) |
| public | reserves | b_reserves... | [null] | CREATE INDEX b_reservesID ON public.reserves USING hash (sid) |
| public | reserves | b_reserves... | [null] | CREATE INDEX b_reservesBID ON public.reserves USING hash (bid) |
| public | query_7 | R_reservesBID | [null] | CREATE INDEX R_reservesBID ON public.query_7 USING btree (sid) |

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

Ln 95, Col 1

pgAdmin 4

Jun 9 4:59 PM • pgAdmin 4

Activities

File Object Tools Help

Servers Local PostgreSQL

Data Query History

```

> p 133
> s 134
> t 135
> r 136
> q 137
-- Query 7 (STATISTICS)
> n 138 set enable_hashagg = off;
> n 139 set enable_hashjoin = off;
> n 140
> n 141 explain analyze select s.sname
from sailors s
where
s.sid in( select r.sid
from reserves r
where r.bid = 103 );
> Log 142
> Tab 143
> pgsql 144
-- Query 7 optimized (STATISTICS)
145
146
147
148
149
-- view of Query 7
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
--
```

Total rows: 1 of 1 Query complete 00:00:00.067

Data output Messages Explain Notifications

Graphical Analysis Statistics

```

graph TD
    A[b_sailorssid] --> B[Merge Semi Join]
    C[r_reservesbid] --> B

```

Successfully run. Total query runtime: 67 msec. 1 rows affected.

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

Ln 165, Col 16

```

-- Query 7 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;

explain analyze select s.sname
from sailors s
where
s.sid in( select r.sid
from reserves r
where r.bid = 103 );

-- Query 7 optimized (STATISTICS)
-- view of Query 7
create MATERIALIZED VIEW query_7
as
select r.sid
from reserves r
where r.bid =103 ;
163
164
explain analyze select s.sname
from sailors s
where exists (select R.sid
              from query_7 R
              where s.sid =R.sid);
171
172
173
Total rows: 7 of 7   Query complete 00:00:00.047

```

Jun 9 4:59 PM • pgAdmin 4

Activities pgAdmin 4 ▾ Jun 9 4:59 PM • pgAdmin 4

Servers Object Tools Help

schema3/postgres@postgres

Local postgres

Data Query History Execute/Refresh

DATA Messages Explain Notifications

QUERY PLAN text

1 Merge Semi Join (cost=0.60..60.37 rows=582 width=+21) (actual time=0.020..0.358 rows=582 loops=1)

2 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..0.093 rows=582 loops=1)

4 -> Index Only Scan using _reservesbid on query_7 r (cost=0.28..31.00 rows=582 width=4) (actual time=0.012..0.132 rows=582 loops=1)

5 Heap Fetches: 582

6 Planning Time: 0.266 ms

7 Execution Time: 0.392 ms

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

Ln 165, Col 1

Explanation :

- Metrics :

Execution Time : 0.392 ms Total Expected Cost : 60.37

- The Query Planner used the Merge join by using both the ZigZag join and the Hash based algorithm together on 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 Mix indices helped in the performance it decreased the Execution Time and Expected Cost (BEST OF THEM ALL).