

Query 7

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

Note !

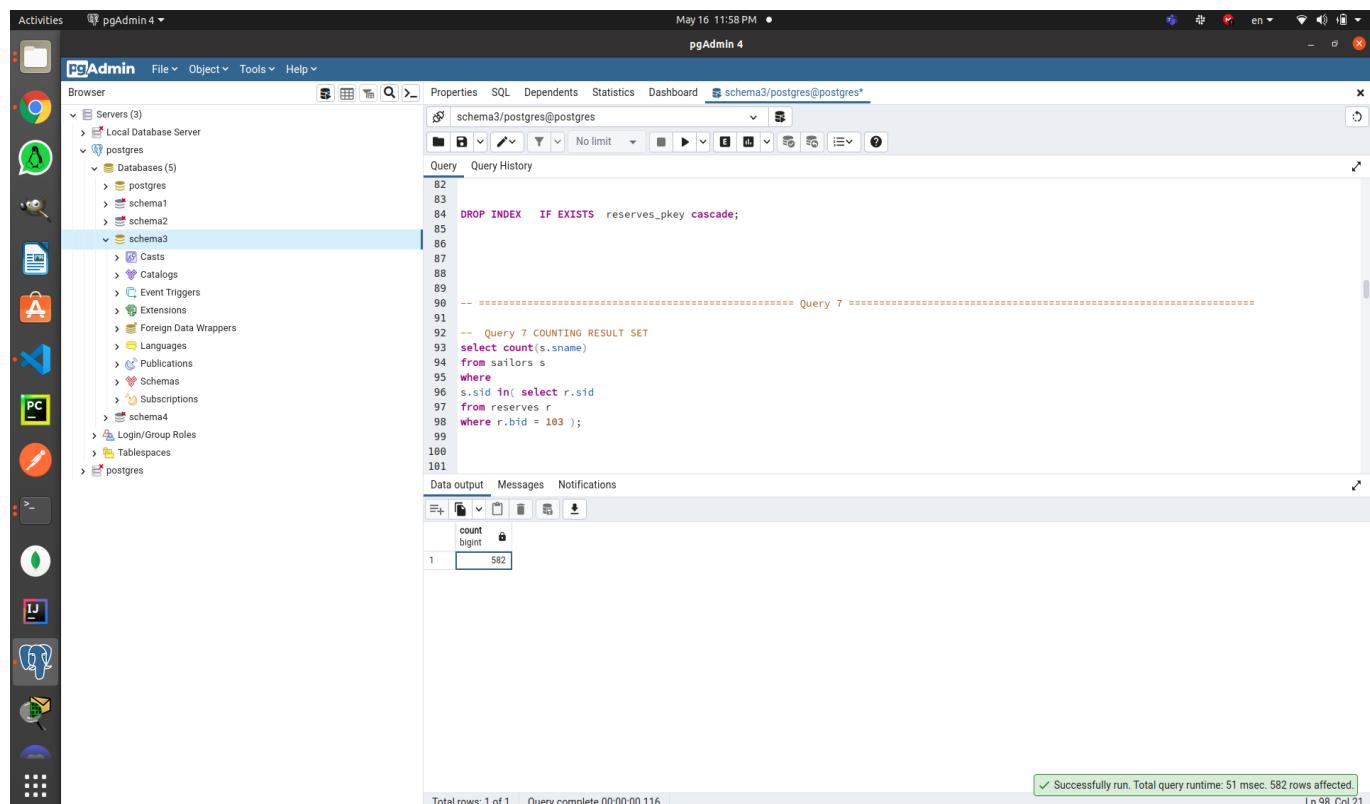
- Flags Hashjoin and HashAgg here were disabled for future after many trials and errors , I've discovered the best way to show the difference in terms of the cost and to beat the Postgres Query Optimizer Algorithm to be able to show indices effect and cost differences .
- The Execution time was changing 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 = 103 );
```

Result Set

- 582 Rows



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

- Toolbar:** Activities, pgAdmin 4, File, Object, Tools, Help.
- Servers:** Local Database Server, postgres (selected), schema3/postgres@postgres*
- Properties:** Properties tab selected.
- Query Editor:** Shows the SQL query and its execution history. The query is:

```
DROP INDEX IF EXISTS reserves_pkey cascade;
-- ====== Query 7 ======
-- Query 7 COUNTING RESULT SET
select count(s.sname)
from sailors s
where
s.sid in( select r.sid
from reserves r
where r.bid = 103 );
```
- Data Output:** A table showing the result of the query:

count	bigint
1	582
- Status Bar:** Total rows: 1 of 1, Query complete 00:00:00.116, Successfully run. Total query runtime: 51 msec. 582 rows affected, Ln 98, Col 21.

Report

1. given query without an index :

pgAdmin 4 - May 17 12:08 AM

```

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

```

Total rows: 0 of 0 Query complete 00:00:00.117 Successfully run. Total query runtime: 51 msec. 582 rows affected. Ln 35, Col 1

pgAdmin 4 - Jun 9 2:11 PM

```

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've reserved a red boat.
162 -- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673
163
164 explain analyze select count(s.sname)

```

Total rows: 1 of 1 Query complete 00:00:00.243 Ln 128, Col 16

```

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

```

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree view shows a connection to 'schema3/postgres'. In the center, a query editor window displays several SQL statements, including an 'explain analyze' command for Query 7. On the right, a 'Data output' pane shows the execution plan for the query, detailing the merge semi join, sort operations, and memory usage.

```

-- 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)
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

- given query with B+ trees indices only :

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

Servers Local | PostgreSQL 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 | PostgreSQL Data Query History 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'tue 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. The left sidebar has icons for various databases and tools. The main window has a title bar "pgAdmin 4" and a status bar "Jun 9 2:57 PM". The left pane shows a tree view of servers and databases, with "schema3/postgres@postgres" selected. The right pane has tabs for "Data output", "Messages", "Explain", and "Notifications". The "Explain" tab is active, displaying a query plan for a query. The query itself is as follows:

```
-- 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)
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

Ln 129, Col 1

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 .
 - 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

```

DROP INDEX IF EXISTS reserves_pkey cascade;
-- ====== 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 hash(sid);
CREATE INDEX b_reservesSID ON reserves USING hash(reserve_id);
CREATE INDEX b_reservesBID ON reserves USING hash(bid);
CREATE INDEX R_reservesBID ON query_7 USING hash(sid);

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tabl
109
110
111
112
113
114
115
116
117
118
119
120
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
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 (reserve_id)
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 (sid)

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

Servers Local PostgreSQL Data Query History Explain Analyze

```

-- Query 7 optimized (COUNTING RESULT SET )
-- Query 7 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;

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

-- 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

```

Data output Messages Explain Notifications

Graphical Analysis Statistics

```

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

```

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

```

-- Query 7 optimized (COUNTING)
-- Number Of Rows = 582
121 select count(s.sname)
from sailors s
where exists (select R.sid
               from query_7 R
              where s.sid = R.sid)
122
123
124
125
126
127
128
129
130 -- Query 7 (STATISTICS)
131 set enable_hashagg = off;
132 set enable_hashjoin = off;
133
134 explain analyze select s.sname
from sailors s
where
135   s.sid in( select r.sid
136     from reserves r
137     where r.bid = 103 );
138
139
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
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);

> s 100
> Log 101
> 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

```

graph LR
    A[b_reservesBID] --> B[sort]
    B --> C[unique]
    C --> D[Nested Loop Inner Join]
    E[b_sailorsSID] --> F[sailors]

```

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

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

```

Activities pgAdmin 4 Jun 9 5:06 PM ● pgAdmin 4
File Object Tools Help
Properties SQL Dependents Statistics Dashboard schema3/postgres@postgres*
Servers Local
pg
  Data Query History
    P 119 s.sid in( select r.sid
    > S 120 from reserves r
    > S 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)
    > S 129 from sailors s
    > S 130 where exists (select R.sid
    > S 131         from query_7 R
    > S 132         where s.sid =R.sid
    > S 133
    > S 134
    > S 135
    > S 136 -- Query 7 (STATISTICS)
    > S 137 set enable_hashagg = off;
    > S 138 set enable_hashjoin = off;
    > S 139 set enable_seqscan = off;
    > S 140
    > S 141
    > S 142
    > S 143 explain analyze select s.sname
    > S 144 from sailors s
    > S 145 where
    > S 146 s.sid in( select r.sid
    > S 147 from reserves r
    > S 148 where r.bid = 103 );
    > S 149
    > S 150
    > S 151 -- Query 7 optimized (STATISTICS)
    > S 152
    > S 153 -- view of Query 7
    > S 154 set enable_hashagg = off;
    > S 155 set enable_hashjoin = off;
    > S 156
    > S 157 -- Query 7 view table of reserves with bid =103
    > S 158
    > S 159 create MATERIALIZED VIEW query_7
Total rows: 23 of 23   Query complete 00:00:00.865

```

DATA output Messages Explain Notifications

QUERY PLAN

```

1 Nested Loop (cost=7602.47..4080651.27 rows=582 width=21) (actual time=67.466..849.911 rows=582 loops=1)
  2 -> Unique (cost=666.44..669.35 rows=578 width=4) (actual time=67.432..67.894 rows=582 loops=1)
  3 -> Sort (cost=666.44..667.89 rows=582 width=4) (actual time=67.431..67.596 rows=582 loops=1)
  4 Sort Key: r.sid
  5 Sort Method: quicksort Memory: 52kB
  6 -> Bitmap Heap Scan on reserves r (cost=12.21..639.71 rows=582 width=4) (actual time=65.412..67.374 rows=582 loops=1)
  7 Recheck Cond: (bid = 103)
  8 Rows Removed by Index Recheck: 23098
  9 Heap Blocks: lossy=128
  10 -> Bitmap Index Scan on b.reservesbid (cost=0.00..12.06 rows=35000 width=0) (actual time=0.015..0.015 rows=128...)
  11 Index Cond: (bid = 103)
  12 -> Bitmap Heap Scan on sailors s (cost=6936.03..7058.78 rows=1 width=25) (actual time=0.038..1.341 rows=1 loops=1)
  13 Recheck Cond: (sid = r.sid)
  14 Rows Removed by Index Recheck: 15359
  15 Heap Blocks: lossy=74496
  16 -> Bitmap Index Scan on b.sailorsid (cost=0.00..6936.03 rows=9500 width=0) (actual time=0.010..0.010 rows=1280...)
  17 Index Cond: (sid = r.sid)
  18 Planning Time: 0.142 ms
  19 JIT:
  20 Functions: 10
  21 Options:Inlining true, Optimization true, Expressions true, Deforming true
  22 Timing: Generation 0.994 ms, Inlining 6.522 ms, Optimization 36.736 ms, Emission 21.999 ms, Total 66.251 ms
  23 Execution Time: 851.120 ms

```

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 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) :

pgAdmin 4

Jun 9 4:56 PM • pgAdmin 4

Servers Local | PostgreSQL Query History Data output Messages Explain Notifications

```

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 btree(sid );
97 CREATE INDEX b_reservesSID ON reserves USING hash(sid );
98 CREATE INDEX b_reservesBID ON reserves USING hash(bid );
99 CREATE INDEX R_reservesBID ON query_7 USING btree(sid );
100
101
102 select *
103 from pg_indexes
104 where tablename = 'sailors' or tablename='reserves' or tablename='boat' or tablename='query_7';
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 00:00:00.056

```

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

Ln 95, Col 1

pgAdmin 4

Jun 9 4:57 PM • pgAdmin 4

Servers Local | PostgreSQL Query History Explain Analyze Graphical Analysis Statistics

```

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
-- Query 7 view table of reserves with bid =103
Total rows: 1 of 1 Query complete 00:00:00.148

```

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

Ln 141, Col 17

```

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

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 - Jun 9 2:22 PM • pgAdmin 4

```

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.
-- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673

```

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 - Jun 9 2:22 PM • pgAdmin 4

```

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.
-- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673

```

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 :

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

Servers Local | PostgreSQL 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';

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

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 3:13 PM ● pgAdmin 4

Servers Local | PostgreSQL Data Query History 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;

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

Total rows: 1 of 1 Query complete 00:00:00.088

```

Ln 158, Col 17

```

graph TD
    A[b_sailorsSID] --> B[Merge Semi Join]
    C[r_reservesBID] --> B

```

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

- Servers:** Local [schema3/postgres@postgres]
- Query History:**

```

> 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
Total rows: 7 of 7  Query complete 00:00:00.104

```
- Execute/Refresh (FS) button:** Located at the top of the query history panel.
- Query Plan:** A table showing the execution plan 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_sailorssid 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
- Text Editor:** Shows the SQL code for creating a materialized view and performing an explain analyze on it.
- Status Bar:** Shows "Ln 158, Col 1".

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 .
- 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 tabname='query_7';
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 red boat'
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 Of Rows =
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

```

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);

```

Total rows: 6 of 6 Query complete 00:00:00.049 Ln 157, Col 1

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]

```

> p 84 DROP INDEX IF EXISTS reserve_7;
> s 85
> s 86
> s 87
> s 88
> s 89
> s 90 -- ====== Query 7 ======
> 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(bid);

> Log 100
> Tab 102
> pgsql 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
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 00:00:00.070

```

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)]

```

> p 143 explain analyze select s.sname
from sailors s
where
s.sid in( select r.sid
from reserves r
where r.bid = 103 );
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176 -- ====== Query 7 ======
177 -- Find the names of sailors 'who ha've reserved a red boat.
178 -- Query 8 (COUNTING RESULT SET ) , Number Of Rows = 673
179
180
181
182
183
184
-- 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);
Total rows: 1 of 1 Query complete 00:00:00.769

```

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

Data output Messages Explain Notifications

Graphical Analysis Statistics

```

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

```

Successful run. Total query runtime: 769 msec. 1 rows affected.

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

```

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;
-- 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.
-- Query 8 (COUNTING RESULT SET) , Number Of Rows = 673
explain analyze select count(s.sname)
from sailors s
where s.sid in ( select r.sid
from reserves r
Total rows: 18 of 18 Query complete 00:00:00.834

```

QUERY PLAN

1	Nested Loop (cost=10000007019.58..10004136191.55 rows=582 width=21) (actual time=61.042..816.068 rows=582 loops=1)
2	-> Unique (cost=1000000035.55..1000000038.46 rows=582 width=4) (actual time=0.094..0.560 rows=582 loops=1)
3	-> Sort (cost=1000000035.55..1000000037.00 rows=582 width=4) (actual time=0.093..0.256 rows=582 loops=1)
4	Sort Key: r.sid
5	Sort Method: quicksort Memory: 52kB
6	-> Seq Scan on query_7 r (cost=10000000000.00..10000000008.82 rows=582 width=4) (actual time=0.009..0.049 rows=582 loops=1)
7	-> Bitmap Heap Scan on sailors s (cost=6984.03..7106.78 rows=1 width=25) (actual time=0.143..1.398 rows=1 loops=582)
8	Recheck Cond: (sid = r.sid)
9	Row Removed by Index Recheck: 15359
10	Heap Blocks: lossy=74496
11	-> Bitmap Index Scan on b_sailorssid (cost=0.00..5984.03 rows=9500 width=0) (actual time=0.011..0.011 rows=1280 loops=582)
12	Index Cond: (sid = r.sid)
13	Planning Time: 0.111 ms
14	JIT:
15	Functions: 6
16	Options: Inlining true, Optimization true, Expressions true, Deforming true
17	Timing: Generation 0.704 ms, Inlining 6.616 ms, Optimization 34.494 ms, Emission 19.675 ms, Total 61.489 ms
18	Execution Time: 816.981 ms

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

Explanation :

- Metrics :

Execution Time : 4679 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 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) :

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 ======
> q 91
> q 92
-- Find the names of sailors who have reserved boat 103.
-- Query 7 COUNTING RESULT SET , Number Of Rows = 582
> q 93
> q 94
> q 95
> q 96 CREATE INDEX b_sailorsSID ON sailors USING btree(sid );
> q 97 CREATE INDEX b_reservesID ON reserves USING hash(sid );
> q 98 CREATE INDEX b_reservesBID ON reserves USING hash(bid );
> q 99 CREATE INDEX R_reservesBID ON query_7 USING btree(sid );
> s 100
> Log 101
> Tab 102
> pgsql 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_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)
> q 138 set enable_hashagg = off;
> q 139 set enable_hashjoin = off;
> q 140
> q 141 explain analyze select s.sname
from sailors s
where
s.sid in( select r.sid
from reserves r
where r.bid = 103 );
> s 142
> Log 143
> Tab 144
> pgsql 145
-- Query 7 optimized (STATISTICS)
> q 146 set enable_hashagg = off;
> q 147 set enable_hashjoin = off;
> s 148
> Log 149
-- view of Query 7
> Tab 150
> pgsql 151
-- view of Query 7
set enable_hashagg = off;
set enable_hashjoin = off;
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
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);
--
```

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

Data output Messages Explain Statistics

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

```

Total rows: 7 of 7 Query complete 00:00:00.047

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 which improved the Execution time and the expected cost way more better which made the join in O($n \log n$).