

QUERY 10

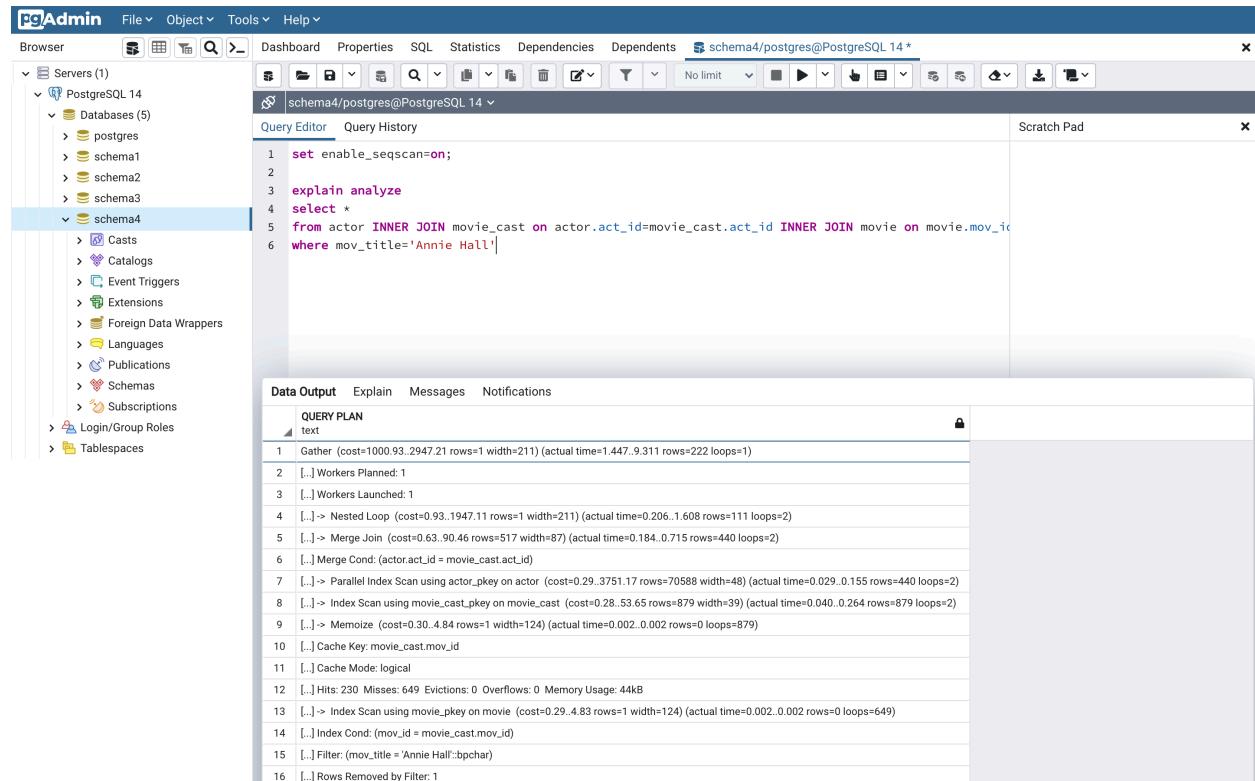
QUERY 10 AFTER OPTIMIZATION IS

```
select *  
from actor INNER JOIN movie_cast on actor.act_id=movie_cast.act_id INNER JOIN movie on  
movie.mov_id = movie_cast.mov_id  
where mov_title='Annie Hall'
```

This is optimized because INNER JOIN are much faster than subqueries so we replaced every IN with a join

Query 10 with no index

Planning and execution time :



The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database structure under 'Servers (1)'. The main area shows a query editor window with the following SQL code:

```
1 set enable_seqscan=on;  
2  
3 explain analyze  
4 select *  
5 from actor INNER JOIN movie_cast on actor.act_id=movie_cast.act_id INNER JOIN movie on movie.mov_id  
6 where mov_title='Annie Hall'
```

Below the query editor, the 'Data Output' tab is selected, showing the 'QUERY PLAN' section. The plan details the execution steps:

- 1 Gather (cost=1000.93..2947.21 rows=1 width=211) (actual time=1.447..9.311 rows=222 loops=1)
- 2 [...] Workers Planned: 1
- 3 [...] Workers Launched: 1
- 4 [...] > Nested Loop (cost=0.93..1947.11 rows=1 width=211) (actual time=0.206..1.608 rows=111 loops=2)
- 5 [...] > Merge Join (cost=0.63..90.46 rows=517 width=87) (actual time=0.184..0.715 rows=440 loops=2)
- 6 [...] Merge Cond: (actor.act_id = movie_cast.act_id)
- 7 [...] > Parallel Index Scan using actor_pkey on actor (cost=0.29..3751.17 rows=70588 width=48) (actual time=0.029..0.155 rows=440 loops=2)
- 8 [...] > Index Scan using movie_cast_pkey on movie_cast (cost=0.28..53.65 rows=879 width=99) (actual time=0.040..0.264 rows=879 loops=2)
- 9 [...] > Memoize (cost=0.30..4.84 rows=1 width=124) (actual time=0.002..0.002 rows=0 loops=879)
- 10 [...] Cache Key: movie_cast.mov_id
- 11 [...] Cache Mode: logical
- 12 [...] Hits: 230 Misses: 649 Evictions: 0 Overflows: 0 Memory Usage: 44kB
- 13 [...] > Index Scan using movie_pkey on movie (cost=0.29..4.83 rows=1 width=124) (actual time=0.002..0.002 rows=0 loops=649)
- 14 [...] Index Cond: (mov_id = movie_cast.mov_id)
- 15 [...] Filter: (mov_title = 'Annie Hall')::bpchar
- 16 [...] Rows Removed by Filter: 1

Execution Plan and costs :

Screenshot of PgAdmin 4 showing the execution plan for a query.

Query:

```

1 set enable_seqscan=on;
2
3 explain analyze
4 select *
5 from actor INNER JOIN movie_cast on actor.act_id=movie_cast.act_id INNER JOIN movie on movie.movie_id=movie_cast.movie_id
6 where mov_title='Annie Hall'

```

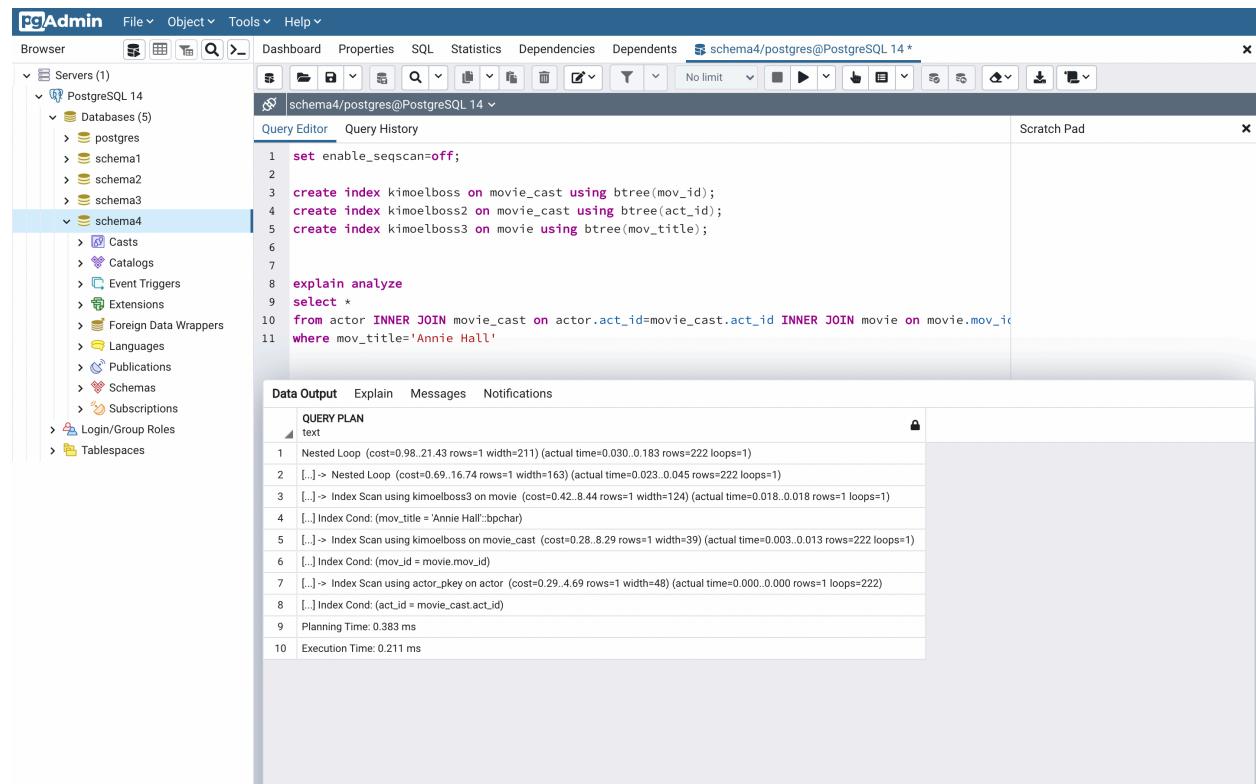
Execution Plan (Analysis tab):

#	Node	Timings			Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan		
1.	→ Gather (cost=1000.93..2947.21 rows=1 width=211) (actual=0.5..)	17.824 ms	19.184 ms	↓ 222	222	1	1	
2.	→ Nested Loop Inner Join (cost=0.93..1947.11 rows=1 width=211)	0.801 ms	1.36 ms	↓ 111	111	1	2	
3.	→ Merge Inner Join (cost=0.63..90.46 rows=517 width=8..)	0.397 ms	0.557 ms	↑ 0.59	440	517	2	
4.	→ Index Scan using actor_pkey on actor as actor (c...	0.063 ms	0.063 ms	↑ 80.22	440	70588	2	
5.	→ Index Scan using movie_cast_pkey on movie_cas...	0.097 ms	0.097 ms	↑ 0.5	879	879	2	
6.	→ Memoize (cost=0.3..4.84 rows=1 width=124) (actual=... Buckets: Batches: Memory Usage: 44 kB)	0.001 ms	0.002 ms	↓ 1	0	1	879	
7.	→ Index Scan using movie_pkey on movie as movie ... Filter: (mov_title = 'Annie Hall')::bpchar Index Cond: (mov_id = movie_cast.mov_id) Rows Removed by Filter: 1	0.001 ms	0.001 ms	↓ 1	0	1	649	

Query 10 using B+ index on : movie_cast (mov_id) , movie_cast (act_id) movie (mov_title)

Disabled flags: seqscan

Planning and execution time :



The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1)', 'PostgreSQL 14' is selected, showing 'Databases (5)' including 'postgres', 'schema1', 'schema2', 'schema3', and 'schema4'. 'schema4' is currently selected. Other sections like 'Casts', 'Catalogs', 'Event Triggers', etc., are also listed. In the main area, the 'Query Editor' tab is active, displaying the following SQL code:

```
1 set enable_seqscan=off;
2
3 create index kimelboss on movie_cast using btree(mov_id);
4 create index kimelbossz on movie_cast using btree(act_id);
5 create index kimelboss3 on movie using btree(mov_title);
6
7
8 explain analyze
9 select *
10 from actor INNER JOIN movie_cast on actor.act_id=movie_cast.act_id INNER JOIN movie on movie.mov_id
11 where mov_title='Annie Hall'
```

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

text
1 Nested Loop (cost=0.98.21.43 rows=1 width=211) (actual time=0.030..0.183 rows=222 loops=1)
2 [...] > Nested Loop (cost=0.69.16.74 rows=1 width=163) (actual time=0.023..0.045 rows=222 loops=1)
3 [...] > Index Scan using kimelboss3 on movie (cost=0.42..8.44 rows=1 width=124) (actual time=0.018..0.018 rows=1 loops=1)
4 [...] Index Cond: (mov_title = 'Annie Hall'::pchar)
5 [...] > Index Scan using kimelboss on movie_cast (cost=0.28..8.29 rows=1 width=39) (actual time=0.003..0.013 rows=222 loops=1)
6 [...] Index Cond: (mov_id = movie.mov_id)
7 [...] > Index Scan using actor_pkey on actor (cost=0.29..4.69 rows=1 width=48) (actual time=0.000..0.000 rows=1 loops=222)
8 [...] Index Cond: (act_id = movie_cast.act_id)
9 Planning Time: 0.383 ms
10 Execution Time: 0.211 ms

Here the B+ tree optimized the query as we put it on the columns that we search on like act_id and mov_id together as well as the mov_title as B+ tree search in O(logn) time so it is better than seqscan

Execution Plan and costs :

Screenshot of PgAdmin 4 showing the execution plan and costs for a query.

Query Editor:

```

1 set enable_seqscan=off;
2
3 create index kimoelboss on movie_cast using btree(mov_id);
4 create index kimoelboss2 on movie_cast using btree(act_id);
5 create index kimoelboss3 on movie using btree(mov_title);
6
7
8 explain analyze
9 select *
10 from actor INNER JOIN movie_cast on actor.act_id=movie_cast.act_id INNER JOIN movie on movie.mov_id
11 where mov_title='Annie Hall'

```

Explain Analysis:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=0.98..21.43 rows=1 width=211) (actual=0.459 ms rows=1)	0.459 ms	0.56 ms	↓ 222	222	1	1
2.	→ Nested Loop Inner Join (cost=0.69..16.74 rows=1 width=163) (actual=0.043 ms rows=1)	0.043 ms	0.1 ms	↓ 222	222	1	1
3.	→ Index Scan using kimoelboss3 on movie as movie (cost=0.01..0.01 rows=1 width=163) (actual=0.01 ms rows=1)	0.01 ms	0.01 ms	↑ 1	1	1	1
4.	→ Index Scan using kimoelboss on movie_cast as movie_cast (cost=0.048..0.048 rows=1 width=163) (actual=0.048 ms rows=1)	0.048 ms	0.048 ms	↓ 222	222	1	1
5.	→ Index Scan using actor_pkey on actor as actor (cost=0.29..4.66 rows=1 width=211) (actual=0.002 ms rows=1)	0.002 ms	0.002 ms	↑ 1	1	1	222

Status: Successfully run. Total query runtime: 42 msec. 1 rows affected.

Query 10 using Hash index on : movie_cast (mov_id), , movie_cast(act_id)), movie (mov_title)

Seq Scan off

Planning and execution time :

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under 'Servers (1)'. The 'schema4' node is selected, showing various objects like Casts, Catalogs, Event Triggers, etc. The main area is the 'Query Editor' tab, which contains the following SQL code:

```
1 set enable_seqscan=off;
2
3
4 create index kimoelboss on movie_cast using hash(mov_id);
5 create index kimoelbossz on movie_cast using hash(act_id);
6 create index kimoelboss3 on movie using hash(mov_title);
7
8
9 explain analyze
10 select *
11 from actor INNER JOIN movie_cast on actor.act_id=movie_cast.act_id INNER JOIN movie on movie.movie_id=movie_cast.movie_id
12 where mov_title='Annie Hall'
```

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

Step	Operation	Cost	Rows	Width	Actual Time
1	Nested Loop	(cost=0.29..20.73 rows=1 width=211)	(actual time=0.014..0.206 rows=222 loops=1)		
2	[...] > Nested Loop	(cost=0.00..16.05 rows=1 width=163)	(actual time=0.008..0.040 rows=222 loops=1)		
3	[...] > Index Scan using kimoelboss3 on movie	(cost=0.00..8.02 rows=1 width=124)	(actual time=0.003..0.004 rows=1 loops=1)		
4	[...] Index Cond: (mov_title = 'Annie Hall':bpchar)				
5	[...] > Index Scan using kimoelboss on movie_cast	(cost=0.00..8.02 rows=1 width=39)	(actual time=0.003..0.022 rows=222 loops=1)		
6	[...] Index Cond: (mov_id = movie.movie_id)				
7	[...] > Index Scan using actor_pkey on actor	(cost=0.29..4.69 rows=1 width=48)	(actual time=0.001..0.001 rows=1 loops=222)		
8	[...] Index Cond: (act_id = movie_cast.act_id)				
9	Planning Time:	0.396 ms			
10	Execution Time:	0.249 ms			

A green success message at the bottom right of the interface states: "Successfully run. Total query runtime: 184 msec. 10 rows affected."

Execution plan and costs:

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1)', there is one entry: 'PostgreSQL 14'. Under 'Databases (5)', there are five databases: 'postgres', 'schema1', 'schema2', 'schema3', and 'schema4'. 'schema4' is selected. The 'Query Editor' tab is active, displaying the following SQL code:

```

1 set enable_seqscan=off;
2
3
4 create index kimoelboss on movie_cast using hash(mov_id);
5 create index kimoelbossz on movie_cast using hash(act_id);
6 create index kimoelboss3 on movie using hash(mov_title);
7
8
9 explain analyze
10 select *
11 from actor INNER JOIN movie_cast on actor.act_id=movie_cast.act_id INNER JOIN movie on movie.mov_id=
12 where mov_title='Annie Hall'

```

Below the query editor, the 'Explain' tab is selected in the 'Data Output' section. The results show the execution plan with timing details:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=0.29..20.73 rows=1 width=211) (actual...)	0.36 ms	0.463 ms	↓ 222	222	1	1
2.	→ Nested Loop Inner Join (cost=0.16..0.5 rows=1 width=163) (act...	0.036 ms	0.103 ms	↓ 222	222	1	1
3.	→ Index Scan using kimoelboss3 on movie as movie (cost=0.00..0.00 width=163) (actual... Index Cond: (mov_title = 'Annie Hall'))	0.011 ms	0.011 ms	↑ 1	1	1	1
4.	→ Index Scan using kimoelboss on movie_cast as movie_ca... (cost=0.00..0.00 width=163) (actual... Index Cond: (mov_id = movie.mov_id))	0.056 ms	0.056 ms	↓ 222	222	1	1
5.	→ Index Scan using actor_pkey on actor as actor (cost=0.29..4.6... (actual... Index Cond: (act_id = movie_cast.act_id))	0.001 ms	0.001 ms	↑ 1	1	1	222

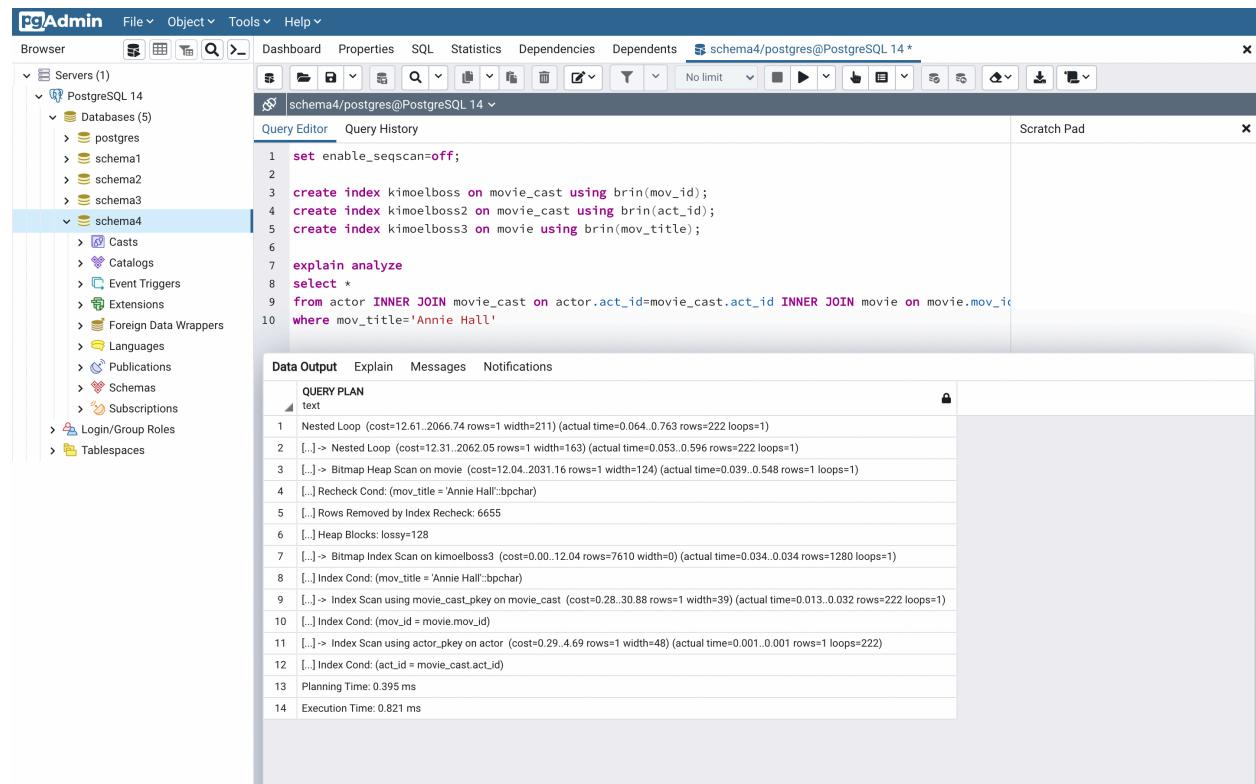
A green success message at the bottom right of the explain output area reads: "Successfully run. Total query runtime: 41 msec. 1 rows affected."

Here the hash index increased performance as it search in O(1) time so it helped in searching from `mov_title="Annie Hall"` and `mov_id` and `act_id`

Query 10 using BRIN index on : movie_cast (mov_id), , movie_cast(act_id)), movie (mov_title)

Disabled flags: seqscan

Planning and execution time :



The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database structure under 'Servers (1)'. The 'schema4' node is selected, showing various objects like Casts, Catalogs, Event Triggers, etc. The main window contains a 'Query Editor' tab with the following SQL code:

```
1 set enable_seqscan=off;
2
3 create index kimoelboss on movie_cast using brin(mov_id);
4 create index kimoelboss2 on movie_cast using brin(act_id);
5 create index kimoelboss3 on movie using brin(mov_title);
6
7 explain analyze
8 select *
9 from actor INNER JOIN movie_cast on actor.act_id=movie_cast.act_id INNER JOIN movie on movie.movie_id
10 where mov_title='Annie Hall'
```

Below the query editor is a 'Data Output' tab which is currently inactive. To its right is the 'Explain' tab, which is active and displays the query plan in 'text' format:

```
QUERY PLAN
text
1 Nested Loop (cost=12.61..2066.74 rows=1 width=211) (actual time=0.064..0.763 rows=222 loops=1)
2 [...] > Nested Loop (cost=12.31..2062.05 rows=1 width=163) (actual time=0.053..0.596 rows=222 loops=1)
3 [...] > Bitmap Heap Scan on movie (cost=12.04..2031.16 rows=1 width=124) (actual time=0.039..0.548 rows=1 loops=1)
4 [...] Recheck Cond: (mov_title = 'Annie Hall'::bpchar)
5 [...] Rows Removed by Index Recheck: 6655
6 [...] Heap Blocks: lossy=128
7 [...] > Bitmap Index Scan on kimoelboss3 (cost=0.00..12.04 rows=7610 width=0) (actual time=0.034..0.034 rows=1280 loops=1)
8 [...] Index Cond: (mov_title = 'Annie Hall'::bpchar)
9 [...] > Index Scan using movie_cast_pkey on movie_cast (cost=0.28..30.88 rows=1 width=39) (actual time=0.013..0.032 rows=222 loops=1)
10 [...] Index Cond: (mov_id = movie.movie_id)
11 [...] > Index Scan using actor_pkey on actor (cost=0.29..4.69 rows=1 width=48) (actual time=0.001..0.001 rows=1 loops=222)
12 [...] Index Cond: (act_id = movie_cast.act_id)
13 Planning Time: 0.395 ms
14 Execution Time: 0.821 ms
```

Here the BRIN index decreased the cost but not much (from 3207 to 2067). BRIN is used in small range queries in big ranges so it does not help that much in exact value queries which was mov_title, we wanted it to be an exact value so BRIN wasn't the most efficient index

Execution plan and costs:

pgAdmin 4.19 - PostgreSQL 14*

Browser

- Servers (1)
 - PostgreSQL 14
 - Databases (5)
 - postgres
 - schema1
 - schema2
 - schema3
 - schema4

Query Editor

```

1 set enable_seqscan=off;
2
3 create index kimoelboss on movie_cast using brin(mov_id);
4 create index kimoelboss2 on movie_cast using brin(act_id);
5 create index kimoelboss3 on movie using brin(mov_title);
6
7 explain analyze
8 select *
9 from actor INNER JOIN movie_cast on actor.act_id=movie_cast.act_id INNER JOIN movie on movie.mov_id
10 where mov_title='Annie Hall'

```

Data Output

Explain

Messages

Notifications

Graphical **Analysis** **Statistics**

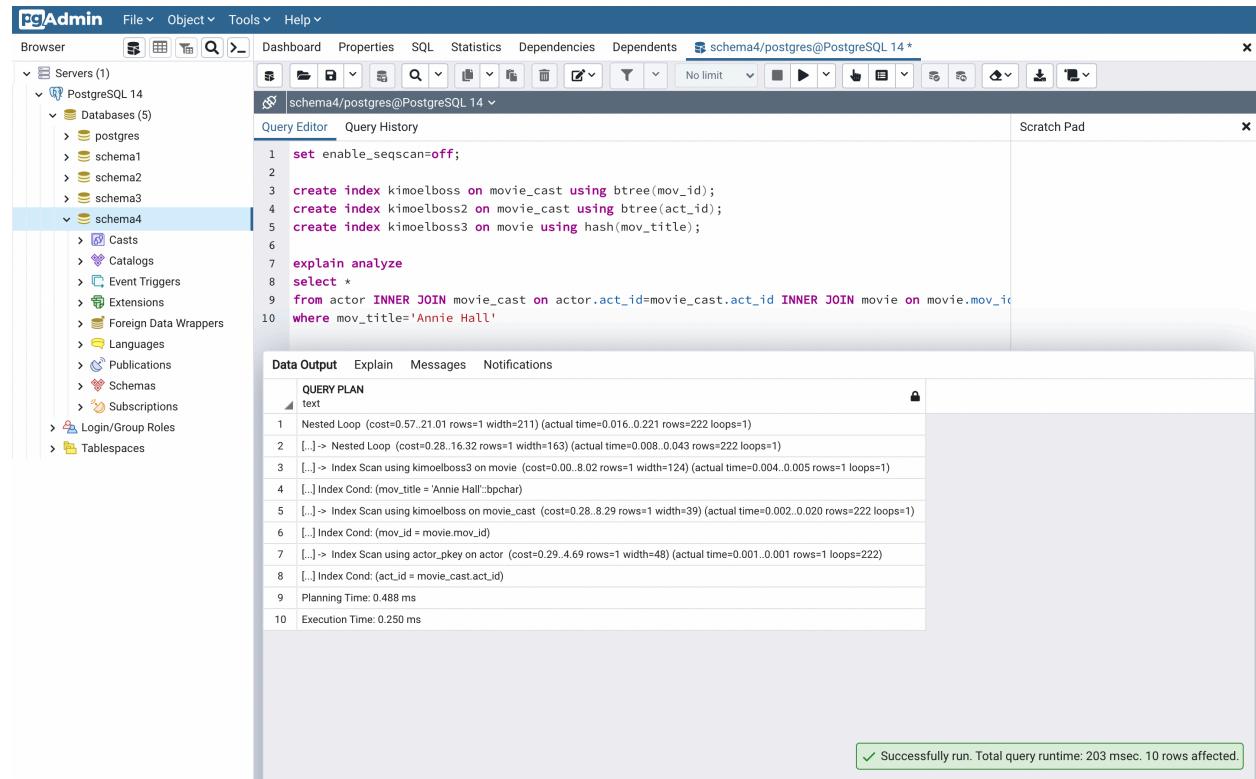
#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=12.61..2066.74 rows=1 width=211) (...) Index Cond: (act_id = movie_cast.act_id)	0.392 ms	2.357 ms	↓ 222	222	1	1
2.	→ Nested Loop Inner Join (cost=12.31..2062.05 rows=1 width=1...) (...) Index Cond: (mov_id = movie_cast.mov_id)	0.043 ms	1.965 ms	↓ 222	222	1	1
3.	→ Bitmap Heap Scan on movie as movie (cost=12.04..2031.00 rows=1 width=104) Recheck Cond: (mov_title = 'Annie Hall')::bpchar Heap Blocks: exact=0	1.815 ms	1.855 ms	↑ 1	1	1	1
4.	→ Bitmap Index Scan using kimoelboss3 (cost=0.12.00..12.00 rows=1 width=104) Index Cond: (mov_title = 'Annie Hall')::bpchar	0.04 ms	0.04 ms	↑ 5.95	1280	7610	1
5.	→ Index Scan using movie_cast_pkey on movie_cast as movie_cast (cost=0.29..4.60 rows=1 width=104) Index Cond: (act_id = movie.mov_id)	0.068 ms	0.068 ms	↓ 222	222	1	1
6.	→ Index Scan using actor_pkey on actor as actor (cost=0.29..4.60 rows=1 width=104) Index Cond: (act_id = movie_cast.act_id)	0.001 ms	0.001 ms	↑ 1	1	1	222

Successfully run. Total query runtime: 36 msec. 1 rows affected.

Query 10 using MIX index on : Btree-> movie_cast (mov_id), , movie_cast(act_id)), Hash -> movie (mov_title)

Disabled flags: seqscan

Planning and execution time :



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

- Servers:** PostgreSQL 14 (selected)
- Databases:** postgres, schema1, schema2, schema3, schema4 (selected)
- Query Editor:** Contains the following SQL code:

```
1 set enable_seqscan=off;
2
3 create index kimoelboss on movie_cast using btree(mov_id);
4 create index kimoelbossz on movie_cast using btree(act_id);
5 create index kimoelboss3 on movie using hash(mov_title);
6
7 explain analyze
8 select *
9 from actor INNER JOIN movie_cast on actor.act_id=movie_cast.act_id INNER JOIN movie on movie.mov_id
10 where mov_title='Annie Hall'
```
- Data Output:** Shows the **QUERY PLAN** tab with the following text:

```
1 Nested Loop (cost=0.57..21.01 rows=1 width=211) (actual time=0.016..0.221 rows=222 loops=1)
2 [...] > Nested Loop (cost=0.28..16.32 rows=1 width=163) (actual time=0.008..0.043 rows=222 loops=1)
3 [...] > Index Scan using kimoelboss3 on movie (cost=0.00..8.02 rows=1 width=124) (actual time=0.004..0.005 rows=1 loops=1)
4 [...] Index Cond: (mov_title = 'Annie Hall'.bpchar)
5 [...] > Index Scan using kimoelboss on movie_cast (cost=0.28..8.29 rows=1 width=39) (actual time=0.002..0.020 rows=222 loops=1)
6 [...] Index Cond: (mov_id = movie.mov_id)
7 [...] > Index Scan using actor_pkey on actor (cost=0.29..4.69 rows=1 width=48) (actual time=0.001..0.001 rows=1 loops=222)
8 [...] Index Cond: (act_id = movie_cast.act_id)
9 Planning Time: 0.488 ms
10 Execution Time: 0.250 ms
```
- Messages:** A green message at the bottom right says "Successfully run. Total query runtime: 203 msec. 10 rows affected."

Here the hash index increased performance and less cost as it search in O(1) time so it helped in searching from mov_title="Annie Hall" and B+ tree optimized the searched for the ids of movie and actor in O(logn) so overall much better performance

Execution plan and costs:

The screenshot shows the pgAdmin interface with a query editor and an explain plan analysis.

Query Editor:

```
1 set enable_seqscan=off;
2
3 create index kimoelboss on movie_cast using btree(mov_id);
4 create index kimoelbossz on movie_cast using btree(act_id);
5 create index kimoelboss3 on movie using hash(mov_title);
6
7 explain analyze
8 select *
9 from actor INNER JOIN movie_cast on actor.act_id=movie_cast.act_id INNER JOIN movie on movie.mov_id
10 where mov_title='Annie Hall'
```

Explain Plan Analysis:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=0.57..21.01 rows=1 width=211) (actual=	0.382 ms	0.474 ms	↓ 222	222	1	1
2.	→ Nested Loop Inner Join (cost=0.28..16.32 rows=1 width=163) (actual=	0.04 ms	0.091 ms	↓ 222	222	1	1
3.	→ Index Scan using kimoelboss3 on movie as movie (cost=0.00..0.01 rows=1 width=211) (actual=	0.011 ms	0.011 ms	↑ 1	1	1	1
4.	→ Index Scan using kimoelboss on movie_cast as movie_cast (cost=0.00..0.01 rows=1 width=211) (actual=	0.04 ms	0.04 ms	↓ 222	222	1	1
5.	→ Index Scan using actor_pkey on actor as actor (cost=0.29..4.66 rows=1 width=211) (actual=	0.001 ms	0.001 ms	↑ 1	1	1	222

Status: Successfully run. Total query runtime: 39 msec. 1 rows affected.

QUERY 11

QUERY 11 AFTER OPTIMIZATION IS

```
select dir_fname, dir_lname
from director d
where exists(
    select dir_id
    from movie_direction md
    where md.dir_id=d.dir_id and exists(
        select mov_id
        from movie_cast mc
        where mc.mov_id=md.mov_id and mc.role =any( select role
            from movie_cast mc2
            where exists(
                select mov_id
                from movie m3
                where
                    mov_title='Eyes Wide Shut' and m3.mov_id=mc2.mov_id ))));
```

This query doesn't have a more optimized alternative. This is more of an equivalent alternative query that produces the same exact result set.

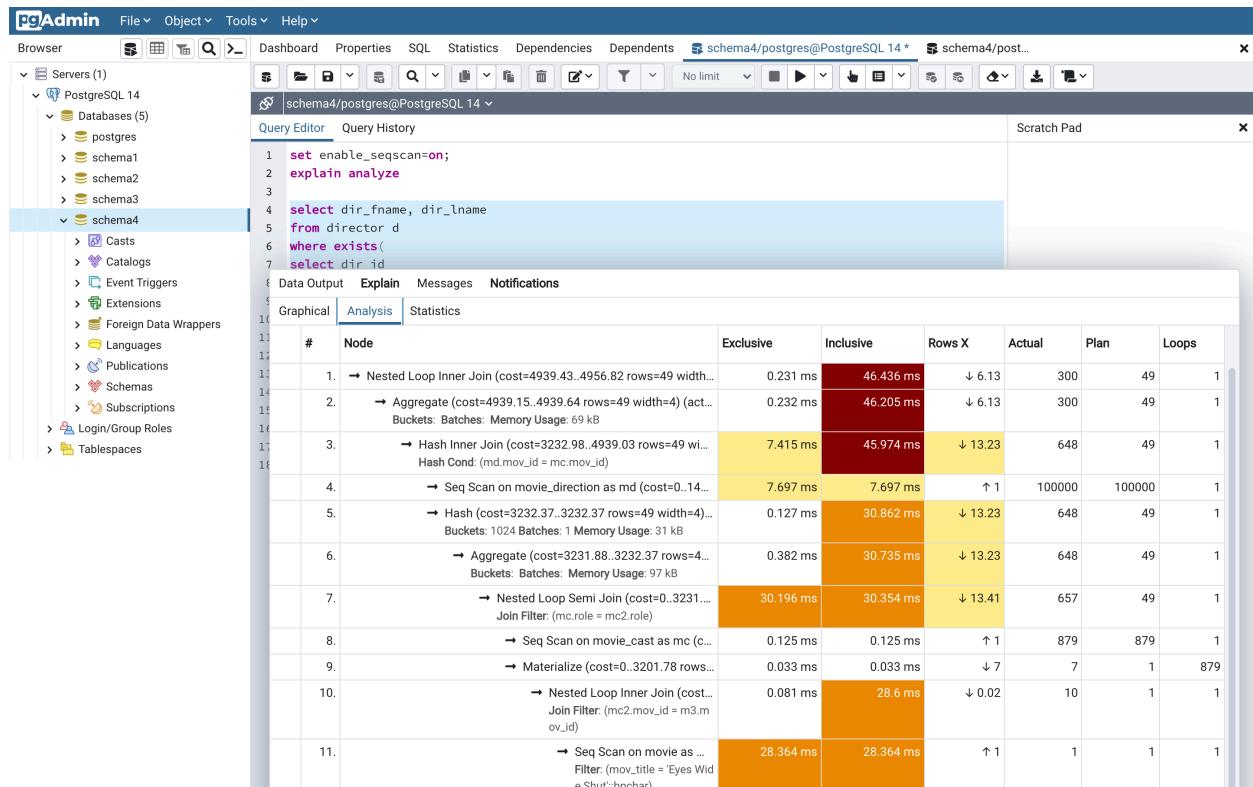
Query 11 with no index

Planning and execution time :

The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database structure under 'Servers (1)'. A tree view shows 'PostgreSQL 14' with 'Databases (5)' containing 'postgres', 'schema1', 'schema2', 'schema3', and 'schema4'. 'schema4' is selected. The main window shows the query plan for a specific query. The title bar says 'schema4/postgres@PostgreSQL 14 * schema4/post...'. The toolbar has various icons for file operations. The main area is divided into sections: 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Explain' section is active and displays the query plan as a numbered list of steps:

- 1 Nested Loop (cost=4939.43..4956.82 rows=49 width=42) (actual time=31.723..31.957 rows=300 loops=1)
- 2 [...] > HashAggregate (cost=4939.15..4939.64 rows=49 width=4) (actual time=31.709..31.725 rows=300 loops=1)
- 3 [...] Group Key: md.dir_id
- 4 [...] Batches: 1 Memory Usage: 69kB
- 5 [...] > Hash Join (cost=3232.98..4939.03 rows=49 width=4) (actual time=19.818..31.537 rows=648 loops=1)
- 6 [...] Hash Cond: (md.mov_id = mc.mov_id)
- 7 [...] > Seq Scan on movie_direction md (cost=0.00..1443.00 rows=100000 width=8) (actual time=0.008..4.851 rows=100000 loops=1)
- 8 [...] > Hash (cost=3232.37..3232.37 rows=49 width=4) (actual time=19.797..19.798 rows=648 loops=1)
- 9 [...] Buckets: 1024 Batches: 1 Memory Usage: 31kB
- 10 [...] > HashAggregate (cost=3231.88..3232.37 rows=49 width=4) (actual time=19.621..19.706 rows=648 loops=1)
- 11 [...] Group Key: mc.mov_id
- 12 [...] Batches: 1 Memory Usage: 97kB
- 13 [...] > Nested Loop Semi Join (cost=0.00..3231.76 rows=49 width=4) (actual time=18.327..19.353 rows=657 loops=1)
- 14 [...] Join Filter: (mc.role = mc2.role)
- 15 [...] Rows Removed by Join Filter: 5166
- 16 [...] > Seq Scan on movie_cast mc (cost=0.00..16.79 rows=879 width=35) (actual time=0.006..0.107 rows=879 loops=1)
- 17 [...] > Materialize (cost=0.00..3201.78 rows=1 width=31) (actual time=0.000..0.021 rows=1 loops=879)
- 18 [...] > Nested Loop (cost=0.00..3201.78 rows=1 width=31) (actual time=0.056..17.797 rows=10 loops=1)
- 19 [...] Join Filter: (mc2.mov_id = m3.mov_id)
- 20 [...] Rows Removed by Join Filter: 869
- 21 [...] > Seq Scan on movie m3 (cost=0.00..3174.00 rows=1 width=4) (actual time=0.006..17.607 rows=1 loops=1)
- 22 [...] Filter: (mov_title = 'Eyes Wide Shut':bpchar)
- 23 [...] Rows Removed by Filter: 99999
- 24 [...] > Seq Scan on movie_cast mc2 (cost=0.00..16.79 rows=879 width=35) (actual time=0.004..0.099 rows=879 loops=1)
- 25 [...] > Index Scan using director_pkey on director d (cost=0.28..0.35 rows=1 width=46) (actual time=0.001..0.001 rows=1 loops=300)
- 26 [...] Index Cond: (dir_id = md.dir_id)
- 27 Planning Time: 0.600 ms
- 28 Execution Time: 32.035 ms

Execution plan and costs:



Query 11 using B+ indices on movie (mov_title) , director (dir_id), movie_direction (mov_id) , movie_cast (role)

Disabled flags: seqscan

Planning and execution time :

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

- Servers:** PostgreSQL 14 (selected)
- Databases:** postgres, schema1, schema2, schema3, schema4 (selected)
- Schema:** schema4/postgres@PostgreSQL 14*
- Query Editor:** The query `set enable_seqscan=off;` is entered.
- Data Output:** The results of the query are displayed as a query plan, which includes the following steps:
 - Nested Loop (cost=65.64..83.03 rows=49 width=42) (actual time=0.353..0.542 rows=300 loops=1)
 - [...] > HashAggregate (cost=65.35..65.84 rows=49 width=4) (actual time=0.348..0.360 rows=300 loops=1)
 - [...] Group Key: md.dir_id
 - [...] Batches: 1 Memory Usage: 69kB
- Merge Semi Join (cost=41.60..65.23 rows=49 width=4) (actual time=0.177..0.293 rows=648 loops=1)
 - [...] > Merge Cond: (md.movie_id = mc.movie_id)
- [...] > Index Scan using kimelboss3 on movie_direction md (cost=0.29..3050.29 rows=100000 width=8) (actual time=0.003..0.038 rows=650 loops=1)
- [...] > Sort (cost=41.30..41.43 rows=49 width=4) (actual time=0.172..0.185 rows=657 loops=1)
 - [...] Sort Key: mc.movie_id
- [...] Sort Method: quicksort Memory: 55kB
- [...] > Nested Loop (cost=39.60..39.93 rows=49 width=4) (actual time=0.044..0.117 rows=657 loops=1)
 - [...] > HashAggregate (cost=39.33..39.34 rows=1 width=31) (actual time=0.041..0.042 rows=10 loops=1)
 - [...] Group Key: mc.role
 - [...] Batches: 1 Memory Usage: 24kB
- [...] > Nested Loop (cost=0.69..39.32 rows=1 width=31) (actual time=0.031..0.038 rows=10 loops=1)
 - [...] > Index Scan using kimelboss3 on movie m3 (cost=0.42..8.44 rows=1 width=4) (actual time=0.021..0.022 rows=1 loops=1)
 - [...] Index Cond: (movie_title = 'Eyes Wide Shut':bpchar)
 - [...] > Index Scan using movie_cast_pkey on movie_cast mc2 (cost=0.28..30.88 rows=1 width=35) (actual time=0.008..0.014 rows=10 loops=1)
 - [...] Index Cond: (movie_id = m3.movie_id)
 - [...] > Index Scan using kimelboss4 on movie_cast mc (cost=0.28..0.55 rows=4 width=35) (actual time=0.001..0.005 rows=66 loops=10)
 - [...] Index Cond: (role = mc2.role)
 - [...] > Index Scan using kimelboss2 on director d (cost=0.28..0.35 rows=1 width=46) (actual time=0.000..0.000 rows=1 loops=300)
 - [...] Index Cond: (dir_id = md.dir_id)
- Planning Time: 0.603 ms
- Execution Time: 0.595 ms
- Explain:** Shows the detailed query plan with various stages and their costs.
- Messages:** No messages present.
- Notifications:** No notifications present.

Here the B+ tree optimized the query as we put it on the columns that we search on like dir_id and mov_id and the mov_title and the role as B+ tree search in $O(\log n)$ time so it is better than seqscan

Execution plan and costs:

PgAdmin 4.14 - PostgreSQL 14

Servers (1) Databases (5) Schemas (4) Tables (1) Functions (0) Procedures (0) Triggers (0)

File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Dependents schema4/postgres@PostgreSQL 14* schema4/post...

Query Editor Query History

```
1 set enable_seqscan=off;
2
3 create index kimoelboss on movie using btree(mov_title);
```

Data Output Explain Messages Notifications

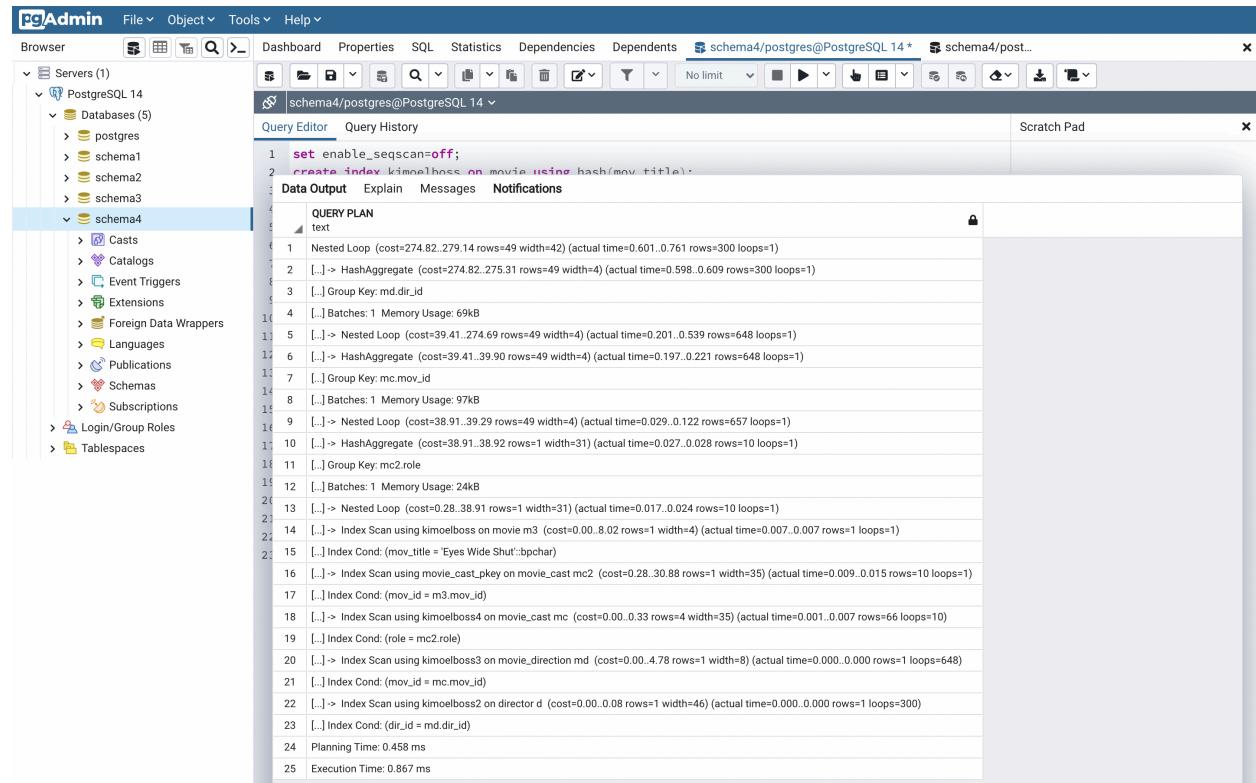
Graphical Analysis Statistics

#	Node	Timings		Rows		
		Exclusive	Inclusive	Rows X	Actual	Plan
1.	→ Nested Loop Inner Join (cost=65.64..83.03 rows=49 width=4... Buckets: Batches: Memory Usage: 69 kB	0.529 ms	1.872 ms	↓ 6.13	300	49
2.	→ Aggregate (cost=65.35..65.84 rows=49 width=4) (actual... Buckets: Batches: Memory Usage: 69 kB	0.219 ms	1.343 ms	↓ 6.13	300	49
3.	→ Merge Semi Join (cost=41.6..65.23 rows=49 width=4... Buckets: Batches: Memory Usage: 69 kB	0.225 ms	1.124 ms	↓ 13.23	648	49
4.	→ Index Scan using kimoelboss3 on movie_director Index Cond: (mov_id = m3.mov_id)	0.25 ms	0.25 ms	↑ 153.85	650	100000
5.	→ Sort (cost=41.3..41.43 rows=49 width=4) (actual... Buckets: Batches: Memory Usage: 69 kB	0.189 ms	0.65 ms	↓ 13.41	657	49
6.	→ Nested Loop Inner Join (cost=39.6..39.93 ... Buckets: Batches: Memory Usage: 24 kB	0.357 ms	0.461 ms	↓ 13.41	657	49
7.	→ Aggregate (cost=39.33..39.34 rows=... Buckets: Batches: Memory Usage: 24 kB	0.013 ms	0.078 ms	↓ 10	10	1
8.	→ Nested Loop Inner Join (cost=0.... Buckets: Batches: Memory Usage: 24 kB	0.006 ms	0.065 ms	↓ 10	10	1
9.	→ Index Scan using kimoelboss... Index Cond: (mov.title = 'Eyes Wide Shut':bpchar)	0.024 ms	0.024 ms	↑ 1	1	1
10.	→ Index Scan using movie_ca... Index Cond: (mov_id = m3.mov_id)	0.035 ms	0.035 ms	↓ 10	10	1
11.	→ Index Scan using kimoelboss4 on mo... Index Cond: (role = mc2.role)	0.026 ms	0.026 ms	↓ 16.5	66	4
12.	→ Index Scan using kimoelboss2 on director as d (cost=0.2... Index Cond: (dir_id = md.dir_id)	0.001 ms	0.001 ms	↑ 1	1	300

Query 11 using Hash index on movie (mov_title) , director (dir_id), movie_direction (mov_id) , movie_cast (role)

Disabled flags: seqscan

Planning and execution time :



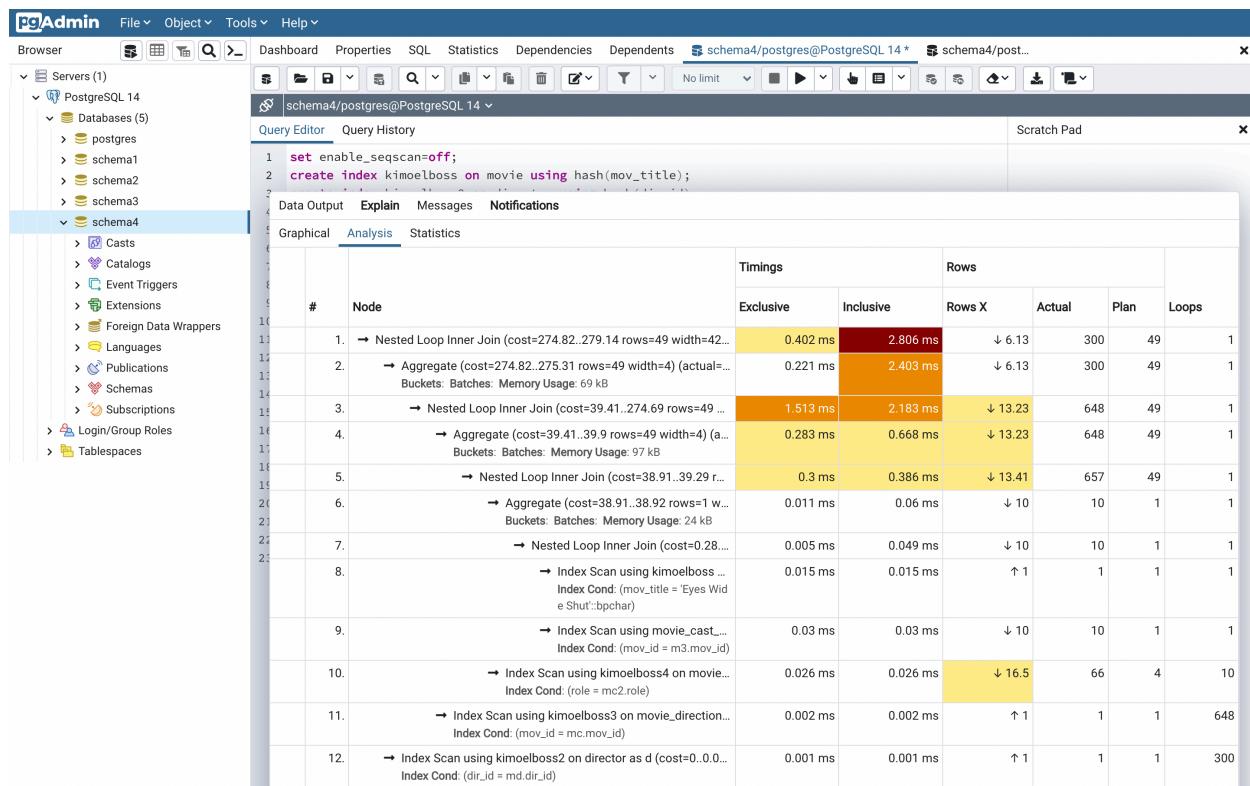
The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database structure with 'schema4' selected. The main area is the 'Query Editor' tab, which contains the following SQL code:

```
1 set enable_seqscan=off;
2 create index kimoeboss on movie using hash(mov_title);
```

Below the code, the 'QUERY PLAN' section shows the execution plan for the query. The plan details nested loops, hash aggregates, and index scans across multiple tables and indexes. The execution time is listed as 0.867 ms.

Here the Hash tree optimized the query as we put it on the columns that we search on like dir_id and mov_id and the mov_title and the role as Hash takes O(1)

Execution plan and costs:



**Query 11 using BRIN index on movie (mov_title) ,
director (dir_id), movie_direction (mov_id) , movie_cast
(role)**

Disabled flags: seqscan

PRIMARY KEY ON

Planning and execution time :

pgAdmin File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents schema4/postgres@PostgreSQL 14* schema4/post...

Servers (1)

- PostgreSQL 14
 - Databases (5)
 - postgres
 - schema1
 - schema2
 - schema3
 - schema4
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas
 - Subscriptions
 - Login/Group Roles
 - Tablespaces

Data Output Explain Messages Notifications

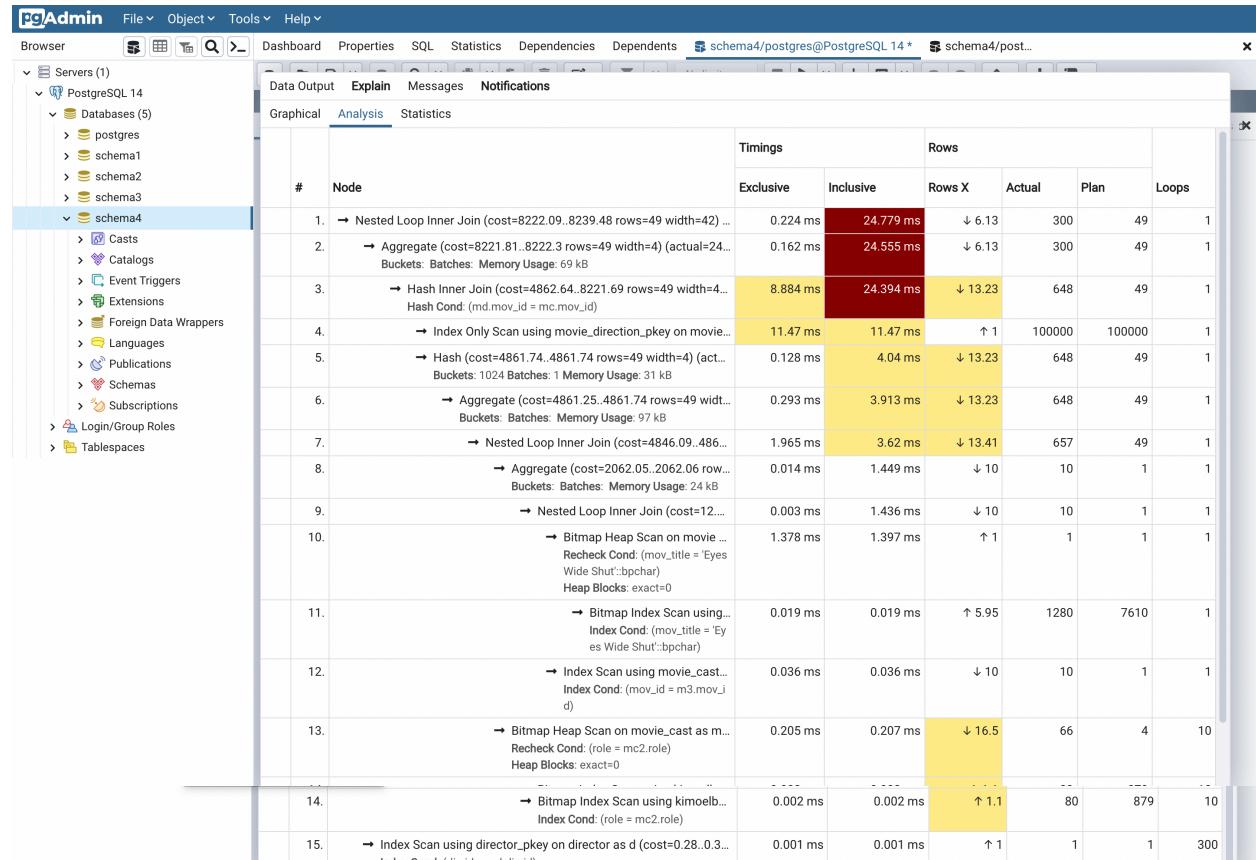
QUERY PLAN

```

1 Nested Loop (cost=8222.09..8239.48 rows=49 width=42) (actual time=10.788..11.051 rows=300 loops=1)
  2 [...] > HashAggregate (cost=8221.81..8222.30 rows=49 width=4) (actual time=10.762..10.780 rows=300 loops=1)
    3 [...] Group Key: md.dir_id
    4 [...] Batches: 1 Memory Usage: 69kB
    5 [...] > Hash Join (cost=4862.64..8221.69 rows=49 width=4) (actual time=1.689..10.688 rows=648 loops=1)
      6 [...] Hash Cond: (md.mov_id = mc.mov_id)
      7 [...] > Index Only Scan using movie_direction_pkey on movie_direction md (cost=0.29..3096.29 rows=100000 width=8) (actual time=0.007..4.655 rows=100000 loops=1)
        8 [...] Heap Fetches: 0
        9 [...] > Hash (cost=4861.74..4861.74 rows=49 width=4) (actual time=1.675..1.676 rows=648 loops=1)
        10 [...] Buckets: 1024 Batches: 1 Memory Usage: 31kB
        11 [...] > HashAggregate (cost=4861.25..4861.74 rows=49 width=4) (actual time=1.587..1.622 rows=648 loops=1)
        12 [...] Group Key: mc.mov_id
        13 [...] Batches: 1 Memory Usage: 97kB
        14 [...] > Nested Loop (cost=4846.09..4861.12 rows=49 width=4) (actual time=0.664..1.474 rows=657 loops=1)
        15 [...] > HashAggregate (cost=2062.05..2062.06 rows=1 width=31) (actual time=0.631..0.633 rows=10 loops=1)
        16 [...] Group Key: mc2.role
        17 [...] Batches: 1 Memory Usage: 24kB
        18 [...] > Nested Loop (cost=12.31..2062.05 rows=1 width=31) (actual time=0.036..0.627 rows=10 loops=1)
        19 [...] > Bitmap Heap Scan on movie m3 (cost=12.04..2031.16 rows=1 width=4) (actual time=0.021..0.604 rows=1 loops=1)
        20 [...] Recheck Cond: (mov.title = 'Eyes Wide Shut':bpchar)
        21 [...] Rows Removed by Index Recheck: 6655
        22 [...] Heap Blocks: lossy=128
        23 [...] > Bitmap Index Scan on kimolboss (cost=0.00..12.04 rows=7610 width=0) (actual time=0.017..0.017 rows=1280 loops=1)
        24 [...] Index Cond: (mov.title = 'Eyes Wide Shut':bpchar)
        25 [...] > Index Scan using movie_cast_pkey on movie_cast mc2 (cost=0.28..30.88 rows=1 width=35) (actual time=0.014..0.021 rows=10 loops=1)
        26 [...] Index Cond: (mov_id = m3.mov_id)
        27 [...] > Bitmap Heap Scan on movie_cast mc (cost=2784.03..2799.02 rows=4 width=35) (actual time=0.022..0.080 rows=66 loops=10)
        28 [...] Recheck Cond: (role = mc2.role)
        29 [...] Rows Removed by Index Recheck: 813
        30 [...] Heap Blocks: lossy=80
        31 [...] > Bitmap Index Scan on kimolboss4 (cost=0.00..2784.03 rows=879 width=0) (actual time=0.003..0.003 rows=80 loops=10)
        32 [...] Index Cond: (role = mc2.role)
        33 [...] > Index Scan using director_pkey on director d (cost=0.28..0.35 rows=1 width=46) (actual time=0.001..0.001 rows=1 loops=300)
        34 [...] Index Cond: (dir_id = md.dir_id)
        35 Planning Time: 0.593 ms
        36 Execution Time: 11.278 ms
  
```

Here the BRIN increased the cost significantly as it is most suitable in small range queries in big ranges. However that wasnot the case and the performance is much bad now

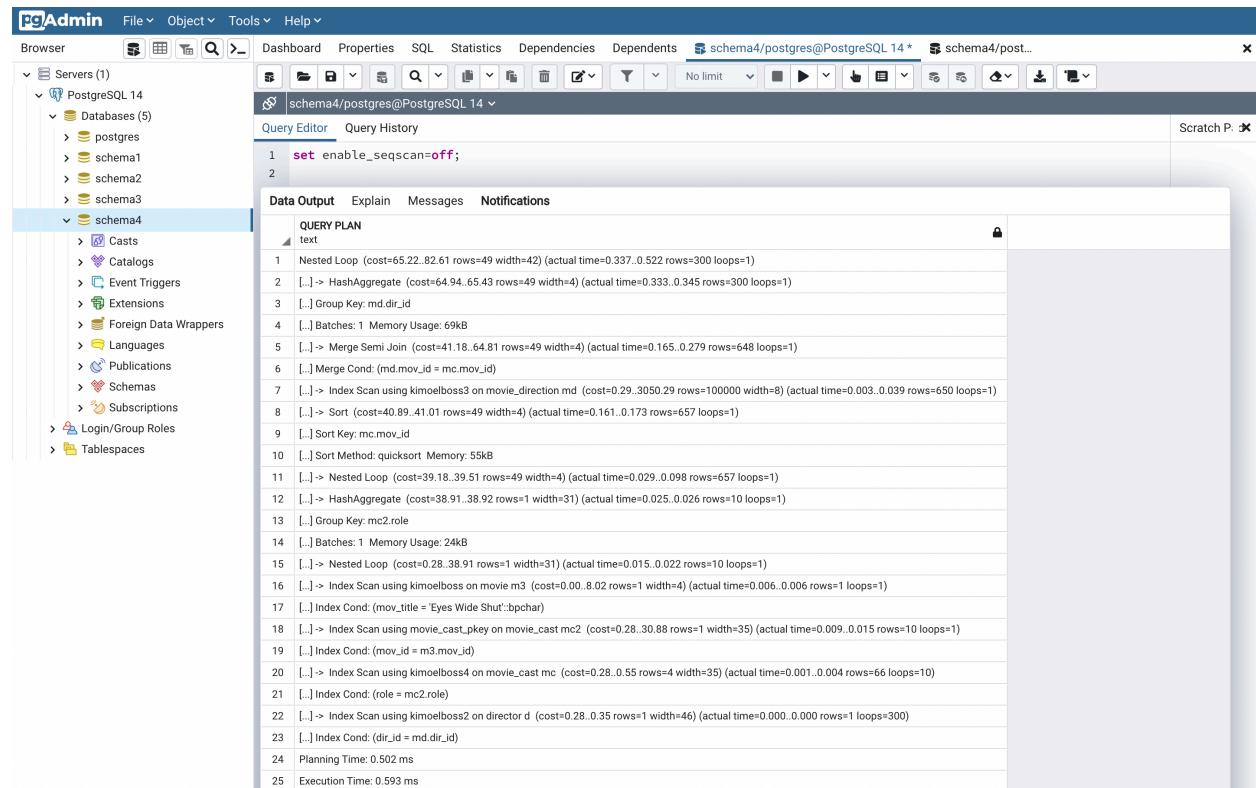
Execution plan and costs:



Query 11 using Hash index on movie (mov_title) , and B+ tree on director (dir_id), movie_direction (mov_id) , movie_cast (role)

Disabled flags: seqscan

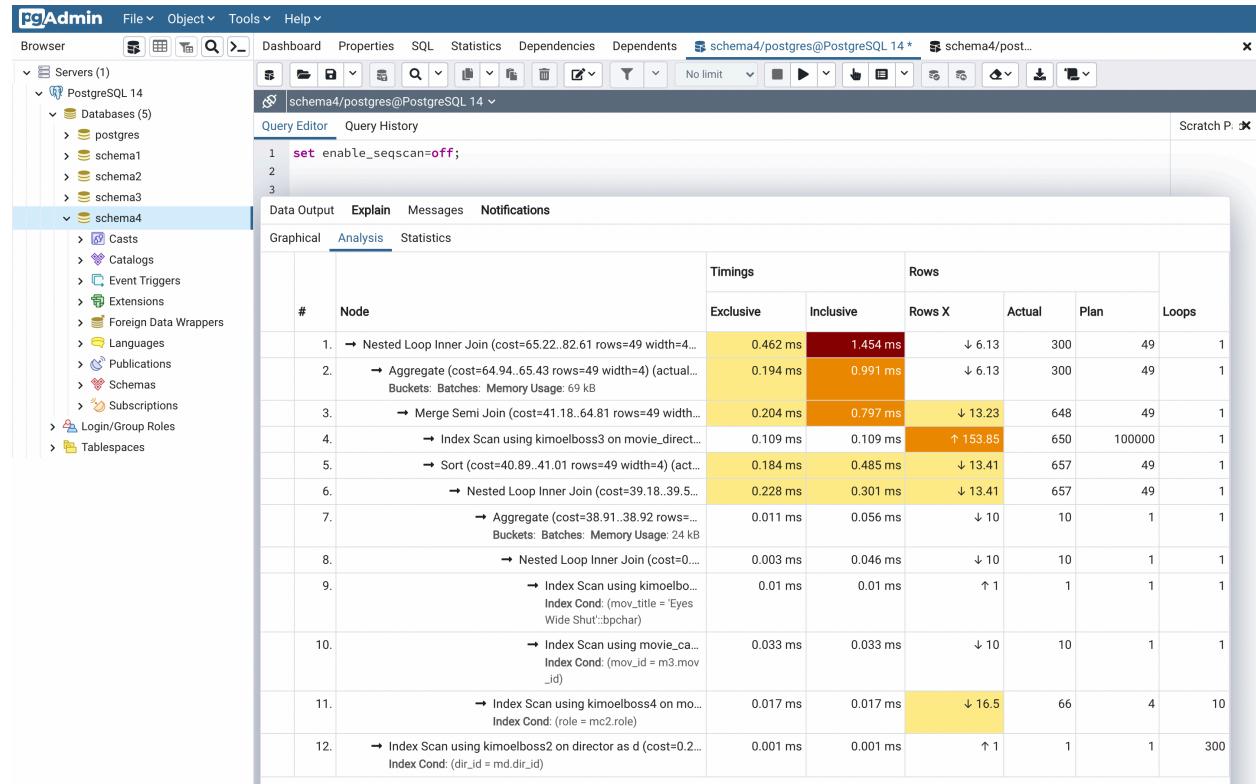
Planning and execution time :



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

- Servers:** PostgreSQL 14
- Databases:** postgres, schema1, schema2, schema3, schema4 (selected)
- Query Editor:** set enable_seqscan=off;
- Data Output:** Explain
- QUERY PLAN:**
 - text
 - 1 Nested Loop (cost=65.22..82.61 rows=49 width=42) (actual time=0.337..0.522 rows=300 loops=1)
 - 2 [...] -> HashAggregate (cost=64.94..65.43 rows=49 width=4) (actual time=0.333..0.345 rows=300 loops=1)
 - 3 [...] Group Key: md.dir_id
 - 4 [...] Batches: 1 Memory Usage: 69kB
 - 5 [...] -> Merge Semi Join (cost=41.18..64.81 rows=49 width=4) (actual time=0.165..0.279 rows=648 loops=1)
 - 6 [...] Merge Cond: (md.mov_id = mc.mov_id)
 - 7 [...] -> Index Scan using kimoeboss3 on movie_direction md (cost=0.29..3050.29 rows=100000 width=8) (actual time=0.003..0.039 rows=650 loops=1)
 - 8 [...] -> Sort (cost=40.89..41.01 rows=49 width=4) (actual time=0.161..0.173 rows=657 loops=1)
 - 9 [...] Sort Key: mc.mov_id
 - 10 [...] Sort Method: quicksort Memory: 55kB
 - 11 [...] -> Nested Loop (cost=39.18..39.51 rows=49 width=4) (actual time=0.029..0.098 rows=657 loops=1)
 - 12 [...] -> HashAggregate (cost=38.91..38.92 rows=1 width=31) (actual time=0.025..0.026 rows=10 loops=1)
 - 13 [...] Group Key: mc2.role
 - 14 [...] Batches: 1 Memory Usage: 24kB
 - 15 [...] -> Nested Loop (cost=0.28..38.91 rows=1 width=31) (actual time=0.015..0.022 rows=10 loops=1)
 - 16 [...] -> Index Scan using kimoeboss on movie m3 (cost=0.00..8.02 rows=1 width=4) (actual time=0.006..0.006 rows=1 loops=1)
 - 17 [...] Index Cond: (mov.title = 'Eyes Wide Shut')::bpchar
 - 18 [...] -> Index Scan using movie_cast_pkey on movie_cast mc2 (cost=0.28..30.88 rows=1 width=35) (actual time=0.009..0.015 rows=10 loops=1)
 - 19 [...] Index Cond: (mov_id = m3.mov_id)
 - 20 [...] -> Index Scan using kimoeboss4 on movie_cast mc (cost=0.28..0.55 rows=4 width=35) (actual time=0.001..0.004 rows=66 loops=10)
 - 21 [...] Index Cond: (role = mc2.role)
 - 22 [...] -> Index Scan using kimoeboss2 on director d (cost=0.28..0.35 rows=1 width=46) (actual time=0.000..0.000 rows=1 loops=300)
 - 23 [...] Index Cond: (dir_id = md.dir_id)
 - 24 Planning Time: 0.502 ms
 - 25 Execution Time: 0.593 ms

Execution plan and costs:



Both B+ tree and hash increased performance as the search in O(logn) and O(1) time

QUERY 12

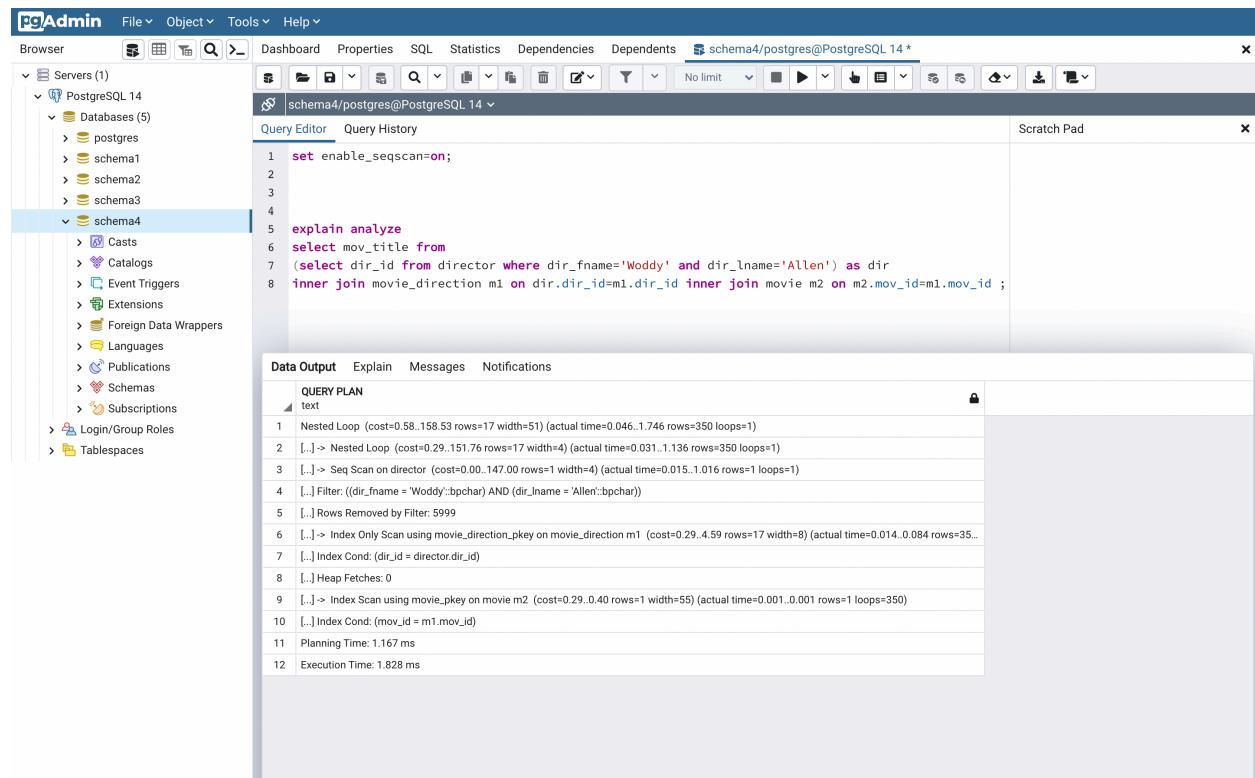
QUERY 12 AFTER OPTIMIZATION IS

```
select mov_title from
(select dir_id from director where dir_fname='Woddy' and dir_lname='Allen') as dir
inner join movie_direction m1 on dir.dir_id=m1.dir_id inner join movie m2 on
m2.mov_id=m1.mov_id ;
```

This is optimized due to the push down selection and the use of inner join instead of IN making it more optimized

Query 12 with no index

Planning and execution time :



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

- Servers:** PostgreSQL 14 (selected)
- Databases:** postgres, schema1, schema2, schema3, schema4 (selected)
- Query Editor:** Contains the query:

```
set enable_seqscan=on;
explain analyze
select mov_title from
(select dir_id from director where dir_fname='Woddy' and dir_lname='Allen') as dir
inner join movie_direction m1 on dir.dir_id=m1.dir_id inner join movie m2 on
m2.mov_id=m1.mov_id ;
```
- Data Output:** Shows the **QUERY PLAN** tab with the following text:

```
1 Nested Loop (cost=0.58..158.53 rows=17 width=51) (actual time=0.046..1.746 rows=350 loops=1)
  2 [...] > Nested Loop (cost=0.29..151.76 rows=17 width=4) (actual time=0.031..1.136 rows=350 loops=1)
  3 [...] > Seq Scan on director (cost=0.00..147.00 rows=1 width=4) (actual time=0.015..1.016 rows=1 loops=1)
  4 [...] Filter: ((dir_fname = 'Woddy':bpchar) AND (dir_lname = 'Allen':bpchar))
  5 [...] Rows Removed by Filter: 5999
  6 [...] > Index Only Scan using movie_direction_pkey on movie_direction m1 (cost=0.29..4.59 rows=17 width=8) (actual time=0.014..0.084 rows=35...
  7 [...] Index Cond: (dir_id = director.dir_id)
  8 [...] Heap Fetches: 0
  9 [...] > Index Scan using movie_pkey on movie m2 (cost=0.29..0.40 rows=1 width=55) (actual time=0.001..0.001 rows=1 loops=350)
  10 [...] Index Cond: (mov_id = m1.mov_id)
  11 Planning Time: 1.167 ms
  12 Execution Time: 1.828 ms
```

Execution plan and costs:

PgAdmin 4.14 - schema4/postgres@PostgreSQL 14*

Browser **File** ▾ **Object** ▾ **Tools** ▾ **Help** ▾

Servers (1) Databases (5) Tables (0) Functions (0) Indexes (0) Triggers (0) Views (0) Materialized Views (0) Sequences (0) Schemas (4) Casts (0) Catalogs (0) Event Triggers (0) Extensions (0) Foreign Data Wrappers (0) Languages (0) Publications (0) Subscriptions (0) Login/Group Roles (0) Tablespaces (0)

Query Editor Query History Scratch Pad

```

1 set enable_seqscan=on;
2
3
4
5 explain analyze
6 select mov_title from
7 (select dir_id from director where dir_fname='Woody' and dir_lname='Allen') as dir
8 inner join movie_direction m1 on dir.dir_id=m1.dir_id inner join movie m2 on m2.mov_id=m1.mov_id ;

```

Data Output Explain Messages Notifications

Graphical **Analysis** Statistics

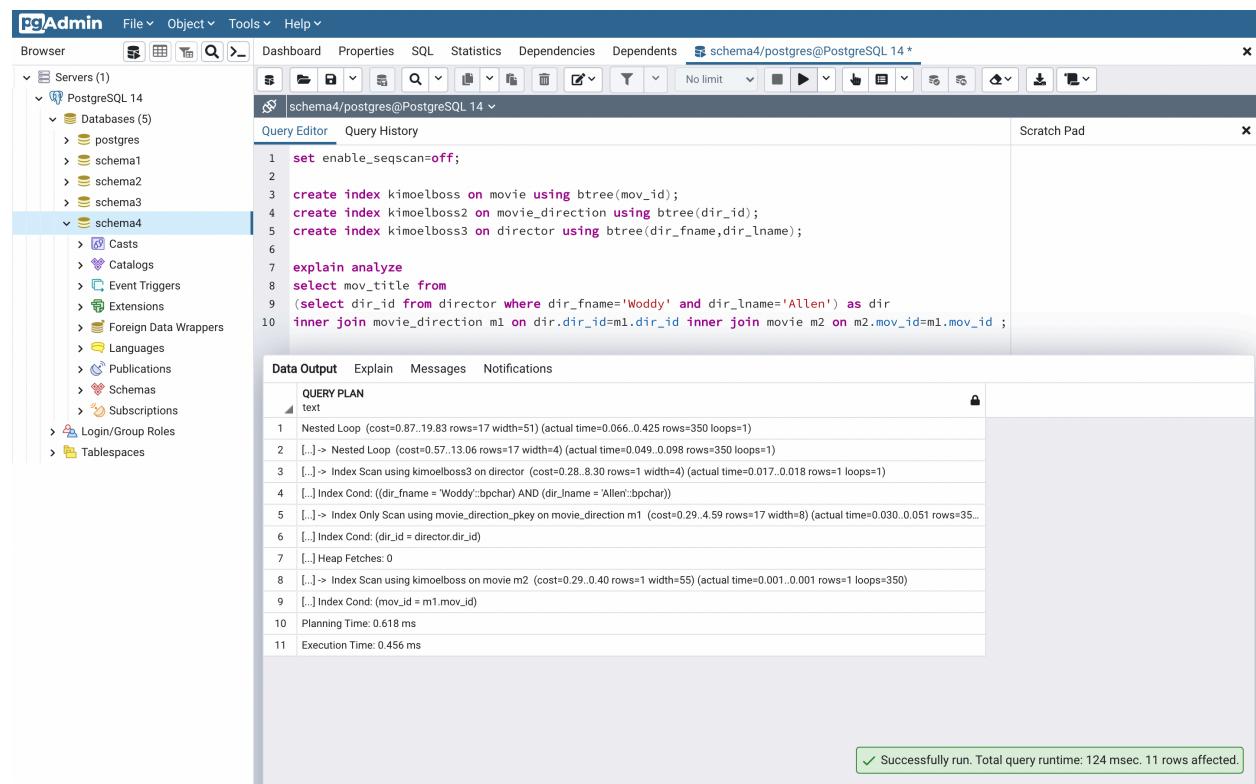
#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=0.58..158.53 rows=17 width=51) (actual=0.595 ms rows=17)	0.595 ms	1.504 ms	↓ 20.59	350	17	1
2.	→ Nested Loop Inner Join (cost=0.29..151.76 rows=17 width=4) (actual=0.037 ms rows=17)	0.037 ms	0.908 ms	↓ 20.59	350	17	1
3.	→ Seq Scan on director as director (cost=0..147 rows=1 width=51) Filter: ((dir_fname = 'Woody')::bpchar) AND ((dir_lname = 'Allen')::bpchar) Rows Removed by Filter: 5999	0.759 ms	0.759 ms	↑ 1	1	1	1
4.	→ Index Only Scan using movie_direction_pkey on movie_dir... Index Cond: (dir_id = director.dir_id)	0.113 ms	0.113 ms	↓ 20.59	350	17	1
5.	→ Index Scan using movie_pkey on movie as m2 (cost=0.29..0.4 ... Index Cond: (mov_id = m1.mov_id)	0.001 ms	0.001 ms	↑ 1	1	1	350

✓ Successfully run. Total query runtime: 41 msec. 1 rows affected.

Query 12 using B+ indices on movie (mov_id) , director (dir_fname,dir_lname), movie_direction (dir_id)

Disabled flags: seqscan

Planning and execution time :



The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database structure under 'Servers (1) > PostgreSQL 14 > Databases (5)'. The 'schema4' database is selected. The main area shows a query editor with the following SQL code:

```
1 set enable_seqscan=off;
2
3 create index kimoelboss on movie using btree(mov_id);
4 create index kimoelboss2 on movie_direction using btree(dir_id);
5 create index kimoelboss3 on director using btree(dir_fname,dir_lname);
6
7 explain analyze
8 select mov_title from
9 (select dir_id from director where dir_fname='Woddy' and dir_lname='Allen') as dir
10 inner join movie_direction m1 on dir.dir_id=m1.dir_id inner join movie m2 on m2.mov_id=m1.mov_id ;
```

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

- 1 Nested Loop (cost=0.87..19.83 rows=17 width=51) (actual time=0.066..0.425 rows=350 loops=1)
- 2 [...] > Nested Loop (cost=0.57..13.06 rows=17 width=4) (actual time=0.049..0.098 rows=350 loops=1)
- 3 [...] > Index Scan using kimoelboss3 on director (cost=0.28..8.30 rows=1 width=4) (actual time=0.017..0.018 rows=1 loops=1)
- 4 [...] Index Cond: ((dir_fname = 'Woddy'::bpchar) AND (dir_lname = 'Allen'::bpchar))
- 5 [...] > Index Only Scan using movie_direction_pkey on movie_direction m1 (cost=0.29..4.59 rows=17 width=8) (actual time=0.030..0.051 rows=35..)
- 6 [...] Index Cond: (dir_id = director.dir_id)
- 7 [...] Heap Fetches: 0
- 8 [...] > Index Scan using kimoelboss on movie m2 (cost=0.29..0.40 rows=1 width=55) (actual time=0.001..0.001 rows=1 loops=350)
- 9 [...] Index Cond: (mov_id = m1.mov_id)
- 10 Planning Time: 0.618 ms
- 11 Execution Time: 0.456 ms

A green success message at the bottom right of the interface states: "Successfully run. Total query runtime: 124 msec. 11 rows affected."

Here the B+ tree optimized the query as we put it on the columns that we search on like dir_fname and dir_lname TOGETHER and mov_id and the dir_id and the role as B+ tree search in O(logn) time so it is better than seqscan

Execution plan and costs:

Screenshot of PgAdmin 4 showing the execution plan and costs for a query.

Query Editor:

```

1 set enable_seqscan=off;
2
3 create index kimoelboss on movie using btree(mov_id);
4 create index kimoelboss2 on movie_direction using btree(dir_id);
5 create index kimoelboss3 on director using btree(dir_fname,dir_lname);
6
7 explain analyze
8 select mov_title from
9 (select dir_id from director where dir_fname='Woddy' and dir_lname='Allen') as dir
10 inner join movie_direction m1 on dir.dir_id=m1.dir_id inner join movie m2 on m2.mov_id=m1.mov_id ;

```

Explain Plan (Analysis tab):

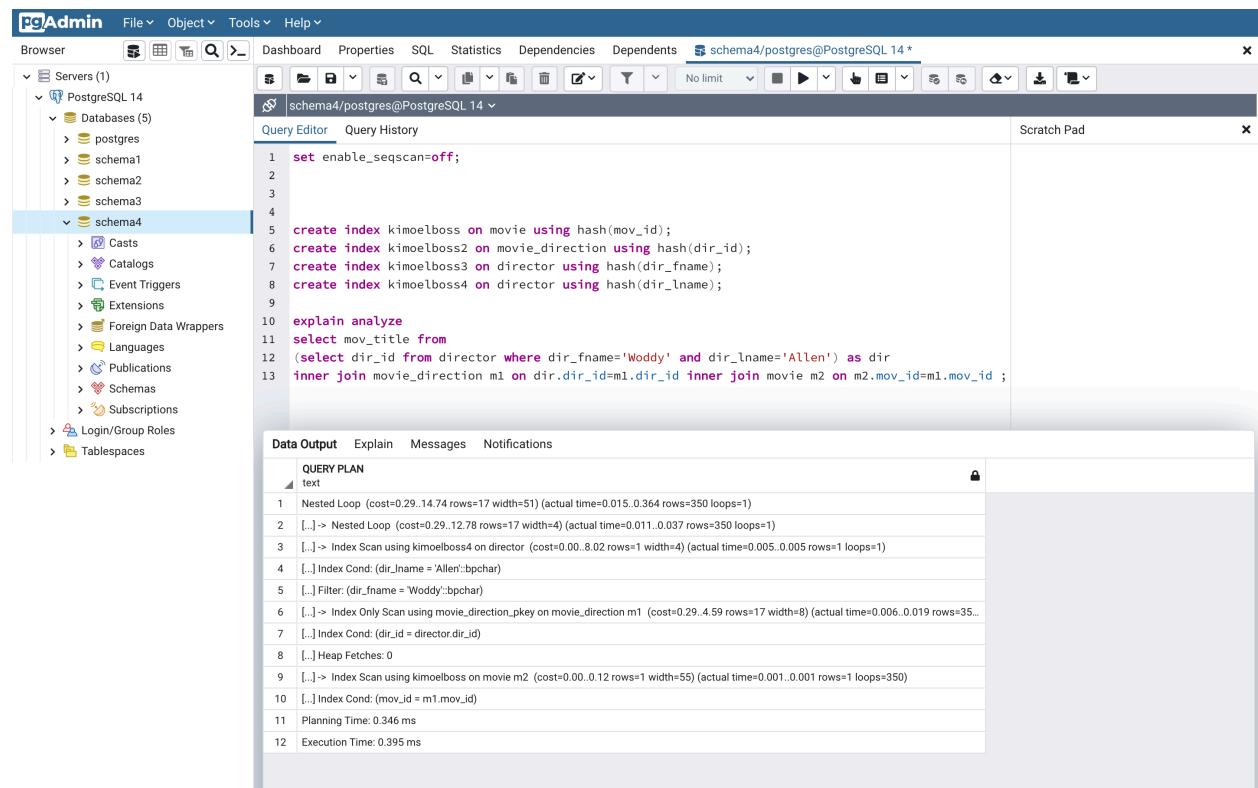
#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=0.87..19.83 rows=17 width=51) (actual=0.55 ms)	0.55 ms	0.665 ms	↓ 20.59	350	17	1
2.	→ Nested Loop Inner Join (cost=0.57..13.06 rows=17 width=4) (actual=0.038 ms)	0.038 ms	0.114 ms	↓ 20.59	350	17	1
3.	→ Index Scan using kimoelboss3 on director as director (cost=0.009 ms) (actual=0.009 ms)	0.009 ms	0.009 ms	↑ 1	1	1	1
4.	→ Index Only Scan using movie_direction_pkey on movie_dir... (cost=0.068 ms) (actual=0.068 ms)	0.068 ms	0.068 ms	↓ 20.59	350	17	1
5.	→ Index Scan using kimoelboss on movie as m2 (cost=0.29..0.4 ms) (actual=0.001 ms)	0.001 ms	0.001 ms	↑ 1	1	1	350

Status Bar: Successfully run. Total query runtime: 33 msec. 1 rows affected.

Query 12 using Hash indices on movie (mov_id) , director (dir_fname), director dir_lname), movie_direction (dir_id)

Disabled flags: seqscan

Planning and execution time :



The screenshot shows the PgAdmin 4 interface. In the left sidebar, under 'Servers (1)', 'PostgreSQL 14' is selected, showing 'Databases (5)' including 'postgres', 'schema1', 'schema2', 'schema3', and 'schema4'. The 'schema4' node is currently selected. In the main area, the 'Query Editor' tab is active, displaying the following SQL code:

```
1 set enable_seqscan=off;
2
3
4
5 create index kimoelboss on movie using hash(mov_id);
6 create index kimoelboss2 on movie_direction using hash(dir_id);
7 create index kimoelboss3 on director using hash(dir_fname);
8 create index kimoelboss4 on director using hash(dir_lname);
9
10 explain analyze
11 select mov_title from
12 (select dir_id from director where dir_fname='Woody' and dir_lname='Allen') as dir
13 inner join movie_direction m1 on dir.dir_id=m1.dir_id inner join movie m2 on m2.mov_id=m1.mov_id ;
```

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

text
1 Nested Loop (cost=0.29..14.74 rows=17 width=51) (actual time=0.015..0.364 rows=350 loops=1)
2 [...] > Nested Loop (cost=0.29..12.78 rows=17 width=4) (actual time=0.011..0.037 rows=350 loops=1)
3 [...] > Index Scan using kimoelboss4 on director (cost=0.00..0.02 rows=1 width=4) (actual time=0.005..0.005 rows=1 loops=1)
4 [...] Index Cond: (dir_lname = 'Allen'::bpchar)
5 [...] Filter: (dir_fname = 'Woody'::bpchar)
6 [...] > Index Only Scan using movie_direction_pkey on movie_direction m1 (cost=0.29..4.59 rows=17 width=8) (actual time=0.006..0.019 rows=35..)
7 [...] Index Cond: (dir_id = director.dir_id)
8 [...] Heap Fetches: 0
9 [...] > Index Scan using kimoelboss on movie m2 (cost=0.00..0.12 rows=1 width=55) (actual time=0.001..0.001 rows=1 loops=350)
10 [...] Index Cond: (mov_id = m1.mov_id)
11 Planning Time: 0.346 ms
12 Execution Time: 0.395 ms

Here the Hash tree optimized the query as we put it on the columns that we search on like dir_id and mov_id and the dir_fname and lname and the role as Hash takes O(1)

Execution plan and costs:

pgAdmin 4.27 - PostgreSQL 14*

Servers (1) Databases (5) Schemas (4) Tables (1) Functions (0) Procedures (0) Triggers (0) Events (0) Extensions (0) Foreign Data Wrappers (0) Languages (0) Publications (0) Schemas (0) Subscriptions (0) Login/Group Roles (0) Tablespaces (0)

Query Editor Query History Scratch Pad

```

1 set enable_seqscan=off;
2
3
4
5 create index kimoelboss on movie using hash(mov_id);
6 create index kimoelboss2 on movie_direction using hash(dir_id);
7 create index kimoelboss3 on director using hash(dir_fname);
8 create index kimoelboss4 on director using hash(dir_lname);
9
10 explain analyze
11 select mov_title from
12 (select dir_id from director where dir_fname='Woddy' and dir_lname='Allen') as dir
13 inner join movie_direction m1 on dir.dir_id=m1.dir_id inner join movie m2 on m2.mov_id=m1.mov_id ;

```

Data Output Explain Messages Notifications

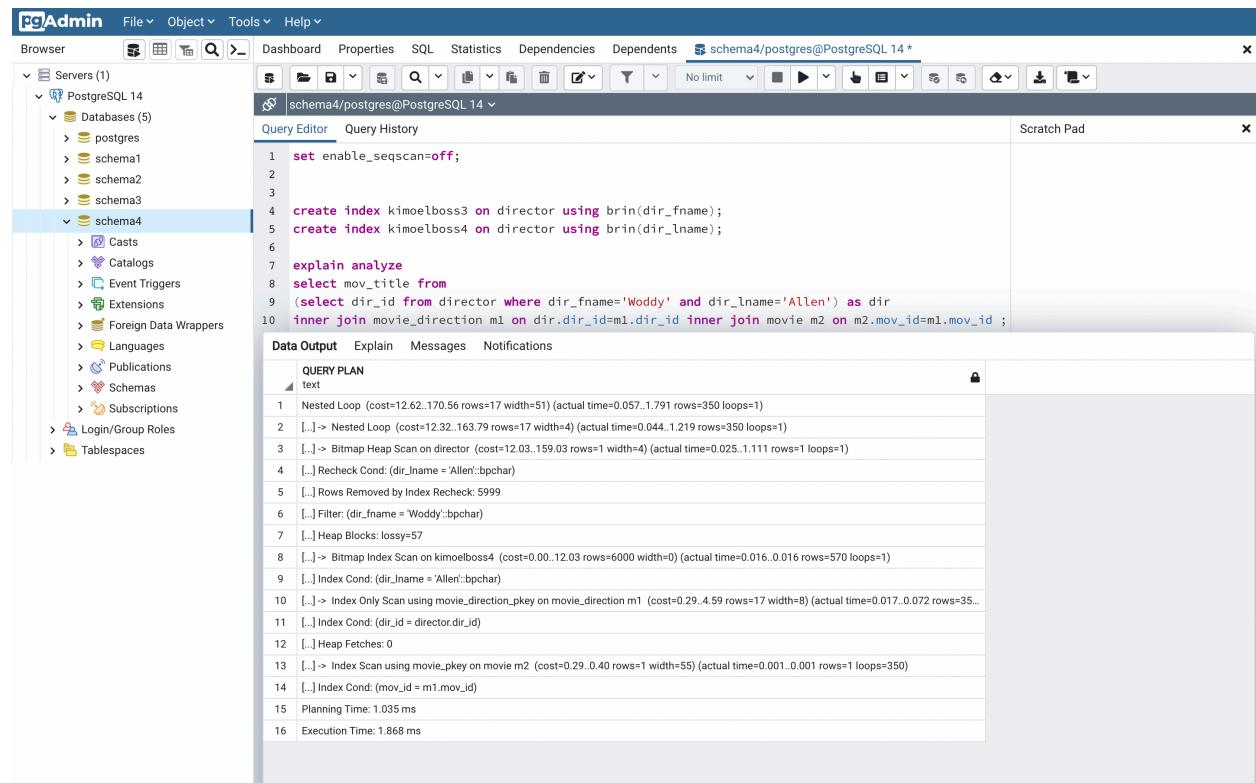
Graphical Analysis Statistics

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=0.29..14.74 rows=17 width=51) (actual=0.861 ms rows=350)	0.861 ms	0.994 ms	↓ 20.59	350	17	1
2.	→ Nested Loop Inner Join (cost=0.29..12.78 rows=17 width=4) (actual=0.039 ms rows=350)	0.039 ms	0.131 ms	↓ 20.59	350	17	1
3.	→ Index Scan using kimoelboss4 on director as director (cost=0.018 ms rows=1) (actual=0.018 ms rows=1)	0.018 ms	0.018 ms	↑ 1	1	1	1
4.	→ Index Only Scan using movie_direction_pkey on movie_direction (cost=0.075 ms rows=1) (actual=0.075 ms rows=1)	0.075 ms	0.075 ms	↓ 20.59	350	17	1
5.	→ Index Scan using kimoelboss on movie as m2 (cost=0..0.12 rows=1) (actual=0.002 ms rows=1)	0.002 ms	0.002 ms	↑ 1	1	1	350

Query 12 using BRIN indices on director (dir_fname), director dir_lname),

Disabled flags: seqscan

Planning and execution time :



The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database structure under 'Servers (1)'. The 'schema4' database is selected. The main area contains a 'Query Editor' tab with the following SQL code:

```
1 set enable_seqscan=off;
2
3
4 create index kimoelboss3 on director using brin(dir_fname);
5 create index kimoelboss4 on director using brin(dir_lname);
6
7 explain analyze
8 select mov_title from
9 (select dir_id from director where dir_fname='Woody' and dir_lname='Allen') as dir
10 inner join movie_direction m1 on dir.dir_id=m1.dir_id inner join movie m2 on m2.mov_id=m1.mov_id;
```

Below the query editor is a 'Data Output' tab showing the 'QUERY PLAN' section. The plan details the execution steps, including nested loops, bitmap scans, and index conditions, which are typical for BRIN index usage.

Here the BRIN increased the cost as it is most suitable in small range queries in big ranges. However that wasnot the case and the performance is bad now from 293 to 305. We used then on director first name and last name

Execution plan and costs:

Screenshot of PgAdmin 4 showing the execution plan for a query.

The query in the Query Editor is:

```

1 set enable_seqscan=off;
2
3
4 create index kimoelboss3 on director using brin(dir_fname);
5 create index kimoelboss4 on director using brin(dir_lname);
6
7 explain analyze
8 select mov_title from
9 (select dir_id from director where dir_fname='Woddy' and dir_lname='Allen') as dir
10 inner join movie_direction m1 on dir.dir_id=m1.dir_id inner join movie m2 on m2.mov_id=m1.mov_id ;

```

The Explain Analysis results table is as follows:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=12.62..170.56 rows=17 width=51) (a... Filter: (dir_fname = 'Woddy'::bpchar) Rows Removed by Filter: 0 Recheck Cond: (dir_lname = 'Allen'::bpchar) Heap Blocks: exact=0	0.561 ms	1.624 ms	↓ 20.59	350	17	1
2.	→ Nested Loop Inner Join (cost=12.32..163.79 rows=17 width=4... Filter: (dir_lname = 'Allen'::bpchar)	0.037 ms	1.062 ms	↓ 20.59	350	17	1
3.	→ Bitmap Heap Scan on director as director (cost=12.03..1... Filter: (dir_fname = 'Woddy'::bpchar) Rows Removed by Filter: 0 Recheck Cond: (dir_lname = 'Allen'::bpchar) Heap Blocks: exact=0	0.958 ms	0.971 ms	↑ 1	1	1	1
4.	→ Bitmap Index Scan using kimoelboss4 (cost=0..12.0... Index Cond: (dir_lname = 'Allen'::bpchar)	0.013 ms	0.013 ms	↑ 10.53	570	6000	1
5.	→ Index Only Scan using movie_direction_pkey on movie_di... Index Cond: (dir_id = director.dir_id)	0.055 ms	0.055 ms	↓ 20.59	350	17	1
6.	→ Index Scan using movie_pkey on movie as m2 (cost=0.29..0.4... Index Cond: (mov_id = m1.mov_id)	0.001 ms	0.001 ms	↑ 1	1	1	350

Message bar: ✓ Successfully run. Total query runtime: 42 msec. 1 rows affected.

Query 12 using Hash indices on director (dir_fname), director dir_lname) and Btree on movie (mov_id) , movie_direction (dir_id)

Disabled flags: seqscan

Planning and execution time :

The screenshot shows the PgAdmin 4 interface. In the left sidebar, under 'Servers (1)', 'PostgreSQL 14' is selected, and 'schema4' is highlighted. The main area contains a query editor window with the following SQL code:

```
1 set enable_seqscan=off;
2
3 create index kimoelboss on movie using btree(mov_id);
4 create index kimoelboss2 on movie_direction using btree(dir_id);
5 create index kimoelboss3 on director using hash(dir_fname);
6 create index kimoelboss4 on director using hash(dir_lname);
7
8
9 explain analyze
10 select mov_title from
11 (select dir_id from director where dir_fname='Woody' and dir_lname='Allen') as dir
12 inner join movie_direction m1 on dir.dir_id=m1.dir_id inner join movie m2 on m2.mov_id=m1.mov_id ;
```

Below the query editor is the 'Data Output' tab, which displays the 'EXPLAIN ANALYZE' results:

QUERY PLAN
1 Nested Loop (cost=0.58..19.55 rows=17 width=51) (actual time=0.027..0.372 rows=350 loops=1) 2 [...] > Nested Loop (cost=0.29..12.78 rows=17 width=4) (actual time=0.019..0.062 rows=350 loops=1) 3 [...] > Index Scan using kimoelboss4 on director (cost=0.00..8.02 rows=1 width=4) (actual time=0.007..0.007 rows=1 loops=1) 4 [...] Index Cond: (dir_lname = 'Allen'::bpchar) 5 [...] Filter: (dir_fname = 'Woody'::bpchar) 6 [...] > Index Only Scan using movie_direction_pkey on movie_direction m1 (cost=0.29..4.59 rows=17 width=8) (actual time=0.012..0.033 rows=35... 7 [...] Index Cond: (dir_id = director.dir_id) 8 [...] Heap Fetches: 0 9 [...] > Index Scan using kimoelboss on movie m2 (cost=0.29..0.40 rows=1 width=55) (actual time=0.001..0.001 rows=1 loops=350) 10 [...] Index Cond: (mov_id = m1.mov_id) 11 Planning Time: 0.559 ms 12 Execution Time: 0.410 ms

A green success message at the bottom right of the interface states: "Successfully run. Total query runtime: 113 msec. 12 rows affected."

Execution plan and costs:

The screenshot shows the pgAdmin interface with a query editor displaying a SQL script and its execution plan.

Query Editor Content:

```
1 set enable_seqscan=off;
2
3 create index kimoelboss on movie using btree(mov_id);
4 create index kimoelboss2 on movie_direction using btree(dir_id);
5 create index kimoelboss3 on director using hash(dir_fname);
6 create index kimoelboss4 on director using hash(dir_lname);
7
8
9 explain analyze
10 select mov_title from
11 (select dir_id from director where dir_fname='Woddy' and dir_lname='Allen') as dir
12 inner join movie_direction m1 on dir.dir_id=m1.dir_id inner join movie m2 on m2.mov_id=m1.mov_id ;
```

Execution Plan (Graphical View):

#	Node	Timings	Rows				
		Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Nested Loop Inner Join (cost=0.58..19.55 rows=17 width=51) (actual...)	0.592 ms	0.709 ms	↓ 20.59	350	17	1
2.	→ Nested Loop Inner Join (cost=0.29..12.78 rows=17 width=4) (...)	0.035 ms	0.116 ms	↓ 20.59	350	17	1
3.	→ Index Scan using kimoelboss4 on director as director (co... Filter: (dir_fname = 'Woddy'::bpchar) Index Cond: (dir_lname = 'Allen'::bpchar) Rows Removed by Filter: 0	0.019 ms	0.019 ms	↑ 1	1	1	1
4.	→ Index Only Scan using movie_direction_pkey on movie_dir... Index Cond: (dir_id = director.dir_id)	0.062 ms	0.062 ms	↓ 20.59	350	17	1
5.	→ Index Scan using kimoelboss on movie as m2 (cost=0.29..0.4 ... Index Cond: (mov_id = m1.mov_id)	0.001 ms	0.001 ms	↑ 1	1	1	350

Message Bar: Successfully run. Total query runtime: 41 msec. 1 rows affected.