

# Query 10 with no indecies

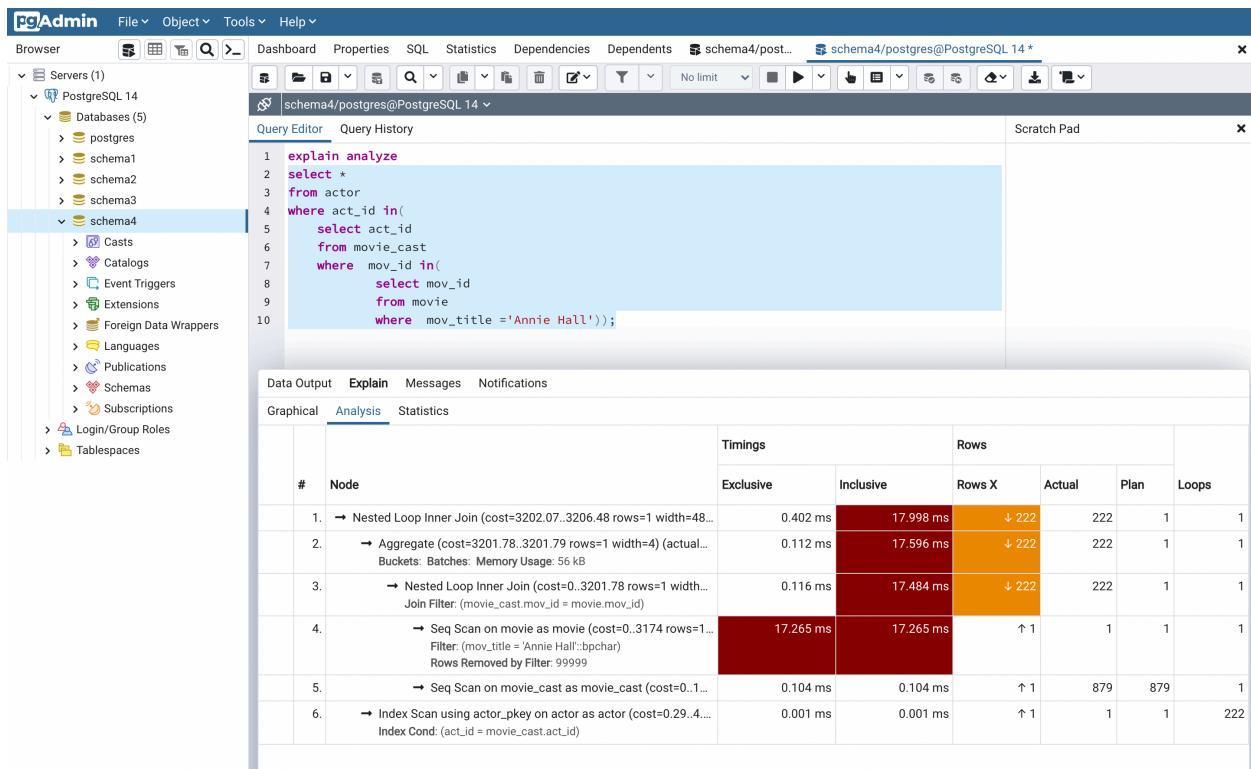
Planning and execution time :

The screenshot shows the PgAdmin 4 interface. On the left, the 'Servers' tree view shows a single server named 'PostgreSQL 14' with several databases listed under it, including 'postgres', 'schema1', 'schema2', 'schema3', and 'schema4'. The 'schema4' node is currently selected. The main window contains a 'Query Editor' tab with the following SQL code:

```
1 explain analyze
2 select *
3 from actor
4 where act_id in(
5     select act_id
6         from movie_cast
7         where mov_id in(
8             select mov_id
9                 from movie
10                where mov_title = 'Annie Hall'));
```

Below the query editor, there is a 'Data Output' tab which is currently inactive. To its right is an 'Explain' tab, which is active and displays the 'QUERY PLAN' for the executed query. The plan shows a nested loop join with a hash aggregate and various filters. At the bottom of the 'Explain' tab, a green message box indicates: 'Successfully run. Total query runtime: 45 msec. 15 rows affected.'

## Execution Plan and costs :



## Query 10 using B+ indices on : movie\_cast (mov\_id,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 its contents: Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas, and Subscriptions. The main area is the 'Query Editor' tab, which contains the following SQL code:

```
1 set enable_seqscan=off;
2 create index kimodelboss on movie_cast
3 using btree(mov_id,act_id);
4 create index kimodelboss2 on movie
5 using btree (mov_title);
6 explain analyze
7 select *
8 from actor
9 where act_id = 2
```

Below the Query Editor is the 'Data Output' tab, which displays the 'QUERY PLAN' for the executed query. The plan details the execution steps:

- Nested Loop (cost=17.03..21.44 rows=1 width=48) (actual time=0.065..0.209 rows=222 loops=1)
- [...] -> HashAggregate (cost=16.74..16.75 rows=1 width=4) (actual time=0.063..0.071 rows=222 loops=1)
- [...] Group Key: movie\_cast.act\_id
- [...] Batches: 1 Memory Usage: 56kB
- [...] -> Nested Loop (cost=0.69..16.74 rows=1 width=4) (actual time=0.017..0.040 rows=222 loops=1)
- [...] -> Index Scan using kimodelboss2 on movie (cost=0.42..8.44 rows=1 width=4) (actual time=0.010..0.010 rows=1 loops=1)
- [...] Index Cond: (mov\_title = 'Annie Hall')::bpchar
- [...] -> Index Only Scan using kimodelboss on movie\_cast (cost=0.28..8.29 rows=1 width=8) (actual time=0.005..0.020 rows=222 loops=1)
- [...] Index Cond: (mov\_id = movie.mov\_id)
- [...] Heap Fetches: 222
- [...] -> 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)
- [...] Index Cond: (act\_id = movie\_cast.act\_id)
- Planning Time: 0.270 ms
- Execution Time: 0.231 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:

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

```
1 set enable_seqscan=off;
2 create index kimelboss on movie_cast
3 using btree(mov_id,act_id);
4 create index kimelboss2 on movie
5 using btree (mov_title);
6 select *
7 from actor
8 where act_id in
```

The execution plan analysis table is displayed below the query editor. The table has the following columns:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=17.03..21.44 rows=1 width=48) (actual=0.365 ms rows=222)	0.365 ms	0.566 ms	↓ 222	222	1	1
2.	→ Aggregate (cost=16.74..16.75 rows=1 width=4) (actual=0.17..0.17)	0.097 ms	0.2 ms	↓ 222	222	1	1
3.	→ Nested Loop Inner Join (cost=0.69..16.74 rows=1 width=4)	0.025 ms	0.104 ms	↓ 222	222	1	1
4.	→ Index Scan using kimelboss on movie_cast (cost=0.69..16.74 rows=222 loops=1) Index Cond: (mov_id = actor.act_id)	0.017 ms	0.017 ms	↑ 1	1	1	1
5.	→ Index Only Scan using kimelboss on movie (cost=0.69..16.74 rows=222 loops=1) Index Cond: (mov_title = 'An Officer and a Gentleman')	0.062 ms	0.062 ms	↓ 222	222	1	1
6.	→ Index Scan using actor_pkey on actor as actor (cost=0.29..4.69 rows=1 width=48) Index Cond: (act_id = movie_cast.act_id)	0.001 ms	0.001 ms	↑ 1	1	1	222

**Query 10 using Hash 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)'. Under 'PostgreSQL 14', there are 'Databases (5)' including 'postgres', 'schema1', 'schema2', 'schema3', and 'schema4'. 'schema4' is selected, highlighted with a blue background. The main area contains a 'Query Editor' tab with the following SQL code:

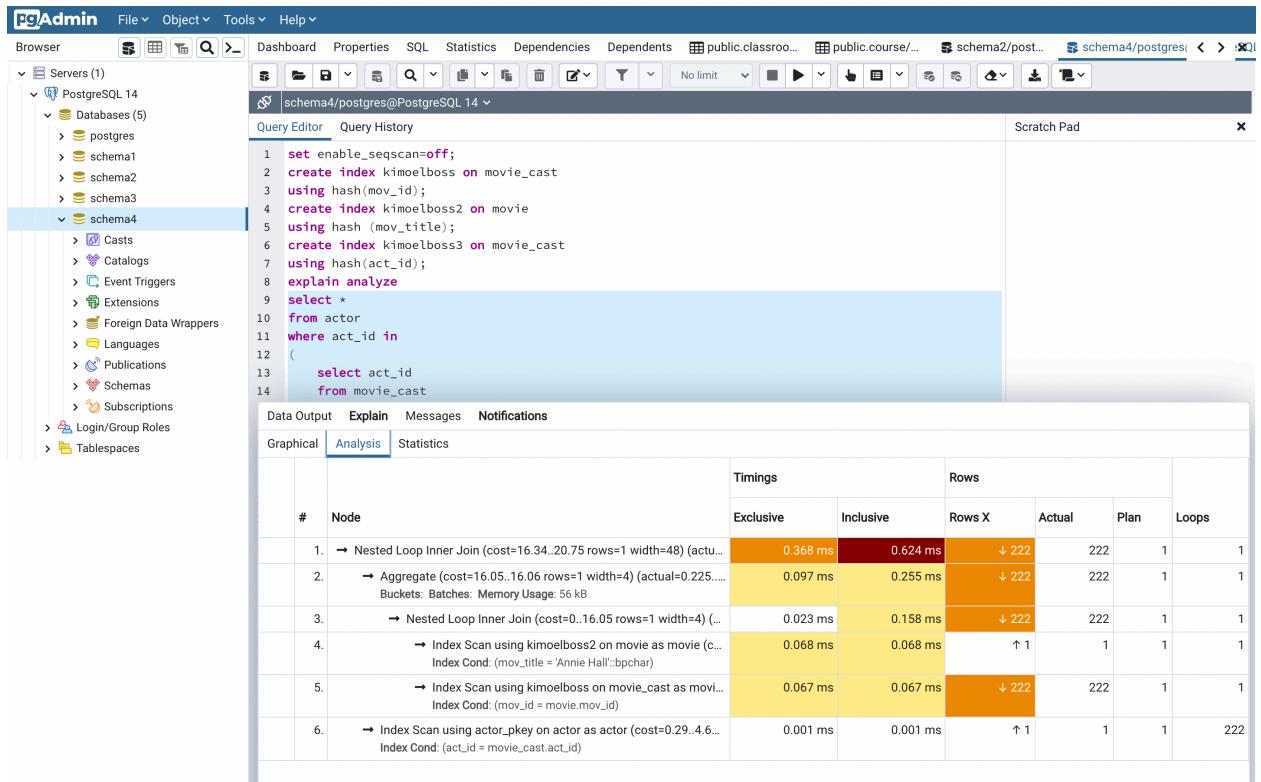
```
1 set enable_seqscan=off;
2 create index kimoelboss on movie_cast
3 using hash(mov_id);
4 create index kimoelboss2 on movie
5 using hash (mov_title);
6 create index kimoelboss3 on movie_cast
7 using hash(act_id);
8 explain analyze
9 select *
10 from actor
11 where act_id in
12 (
13     select act_id
14         from movie_cast
15     where mov_id in
16     (

```

Below the query editor, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Explain' tab is active, showing the 'QUERY PLAN' with the following text output:

```
1 Nested Loop (cost=16.34..20.75 rows=1 width=48) (actual time=0.076..0.265 rows=222 loops=1)
2 [...] > HashAggregate (cost=16.05..16.06 rows=1 width=4) (actual time=0.072..0.082 rows=222 loops=1)
3 [...] Group Key: movie_cast.act_id
4 [...] Batches: 1 Memory Usage: 56kB
5 [...] > Nested Loop (cost=0.00..16.05 rows=1 width=4) (actual time=0.010..0.039 rows=222 loops=1)
6 [...] > Index Scan using kimoelboss2 on movie (cost=0.00..8.02 rows=1 width=4) (actual time=0.005..0.005 rows=1)
7 [...] Index Cond: (mov_title = 'Annie Hall')::bpchar
8 [...] > Index Scan using kimoelboss on movie_cast (cost=0.00..8.02 rows=1 width=8) (actual time=0.005..0.022 rows=222 loops=1)
9 [...] Index Cond: (mov_id = movie.mov_id)
10 [...] > 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)
11 [...] Index Cond: (act_id = movie_cast.act_id)
12 Planning Time: 0.330 ms
13 Execution Time: 0.289 ms
```

## Execution plan and costs:



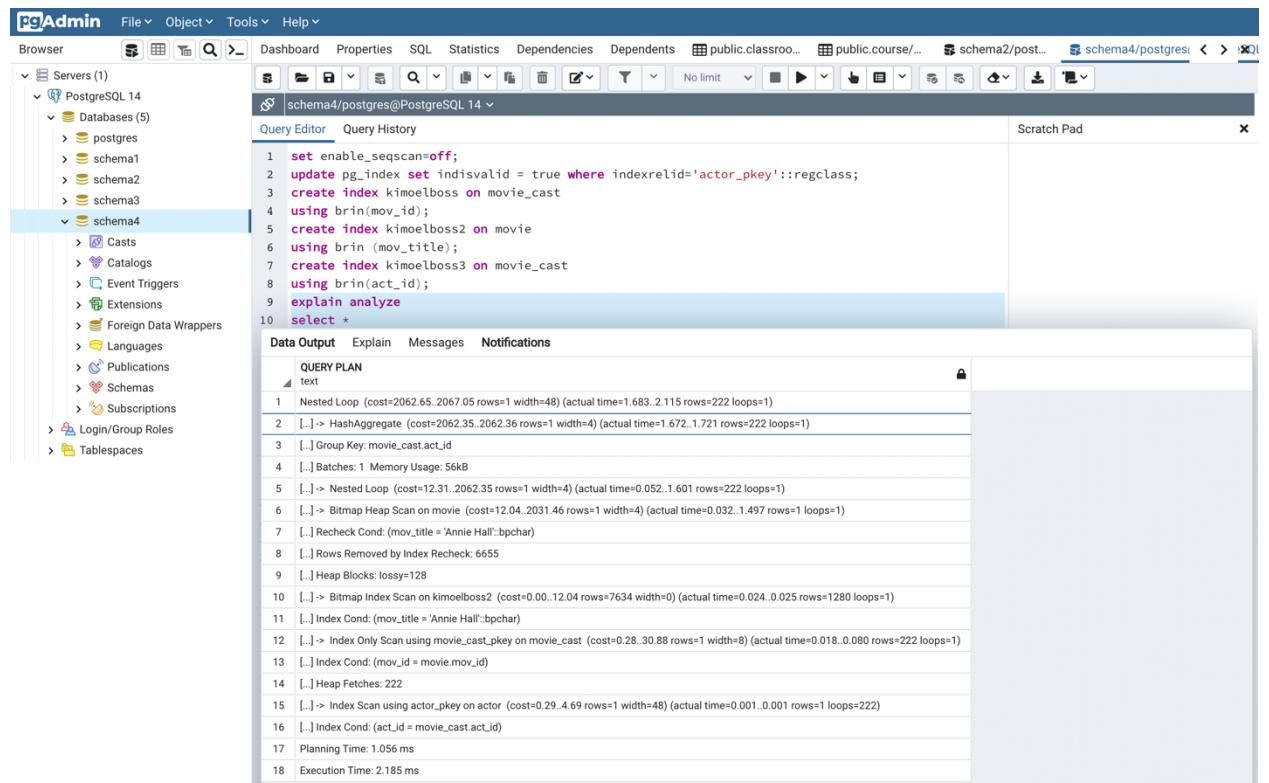
Here the hash index increased performance as it search in O(1) time so it helped in searching from mov\_title="Annie Hall" and mov\_id and act\_id

# Query 10 using BRIN index on : movie\_cast (mov\_id ), , movie\_cast(act\_id)), movie (mov\_title)

Disabled flags: seqscan

Planning and execution time :

## Without disabling the primary key



The screenshot shows the PgAdmin interface with the following details:

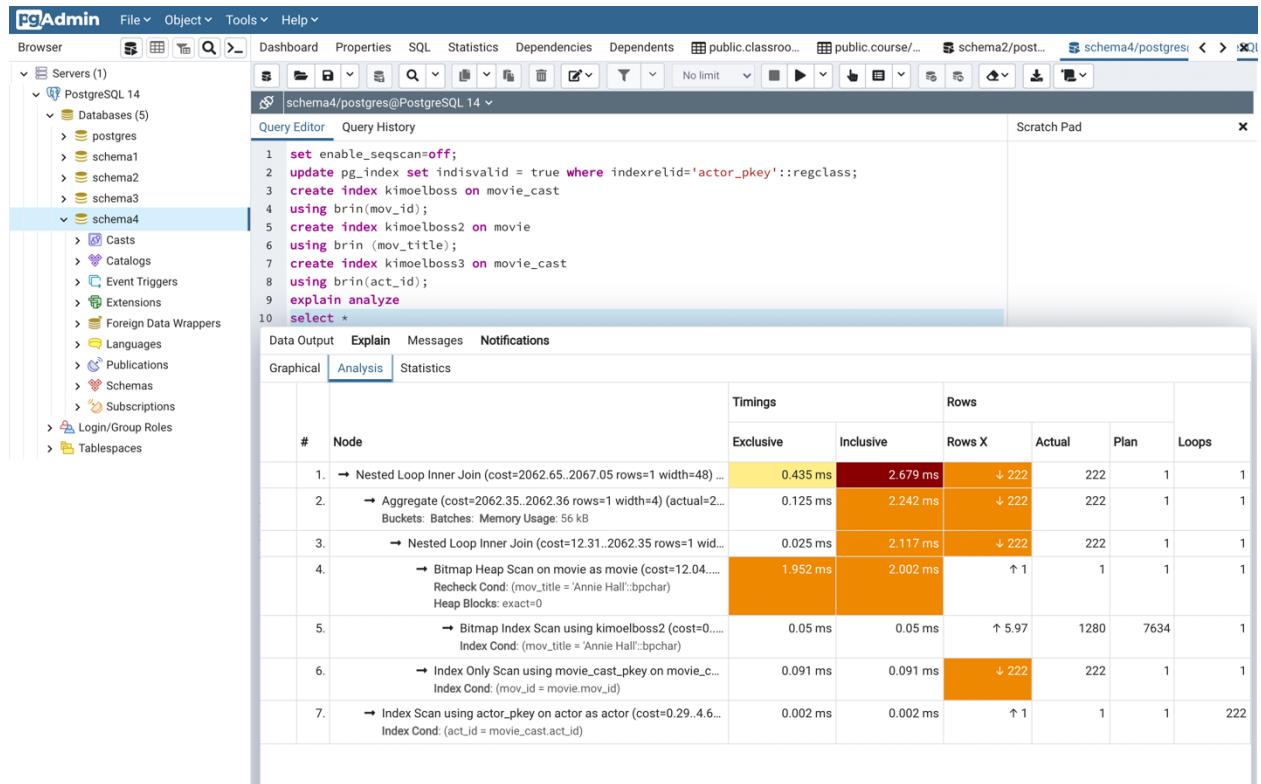
- Servers:** PostgreSQL 14
- Databases:** postgres, schema1, schema2, schema3, schema4 (selected)
- Query Editor:** Contains the following SQL code:

```
1 set enable_seqscan=off;
2 update pg_index set indisvalid = true where indexrelid='actor_pkey'::regclass;
3 create index kimodelboss on movie_cast
4 using brin(mov_id);
5 create index kimodelboss2 on movie
6 using brin (mov_title);
7 create index kimodelboss3 on movie_cast
8 using brin(act_id);
9 explain analyze
10 select *
```
- Data Output:** Shows the query plan with 18 numbered steps, including:
  - Step 1: Nested Loop (cost=2062.65..2067.05 rows=1 width=48) (actual time=1.683..2.115 rows=222 loops=1)
  - Step 2: HashAggregate (cost=2062.35..2062.36 rows=1 width=4) (actual time=1.672..1.721 rows=222 loops=1)
  - Step 3: Group Key: movie\_cast.act\_id
  - Step 4: Batches: 1 Memory Usage: 56kB
  - Step 5: Nested Loop (cost=12.31..2062.35 rows=1 width=4) (actual time=0.052..1.601 rows=222 loops=1)
  - Step 6: Bitmap Heap Scan on movie (cost=12.04..2031.46 rows=1 width=4) (actual time=0.032..1.497 rows=1 loops=1)
  - Step 7: Recheck Cond: (mov\_title = 'Annie Hall')::bpchar
  - Step 8: Rows Removed by Index Recheck: 6655
  - Step 9: Heap Blocks: lossy=128
  - Step 10: Bitmap Index Scan on kimodelboss2 (cost=0.00..12.04 rows=7634 width=0) (actual time=0.024..0.025 rows=1280 loops=1)
  - Step 11: Index Cond: (mov\_title = 'Annie Hall')::bpchar
  - Step 12: Index Only Scan using movie\_cast\_pkey on movie\_cast (cost=0.28..30.88 rows=1 width=8) (actual time=0.018..0.080 rows=222 loops=1)
  - Step 13: Index Cond: (mov\_id = movie.mov\_id)
  - Step 14: Heap Fetches: 222
  - Step 15: 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)
  - Step 16: Index Cond: (act\_id = movie\_cast.act\_id)
  - Step 17: Planning Time: 1.056 ms
  - Step 18: Execution Time: 2.185 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:

### Without disabling the primary key



**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 interface with a query editor containing the following SQL code:

```
1 set enable_seqscan=off;
2 create index kimelboss on movie_cast
3 using btree(mov_id);
4 create index kimelboss2 on movie
5 using hash (mov_title);
6 create index kimelboss3 on movie_cast
7 using btree(act_id);
8 explain analyze
9 select *
10 from actor
```

The "Data Output" tab shows the query plan:

QUERY PLAN
1 Nested Loop (cost=16.62..21.02 rows=1 width=48) (actual time=0.067..0.207 rows=222 loops=1) 2 [...] > HashAggregate (cost=16.32..16.33 rows=1 width=4) (actual time=0.053..0.060 rows=222 loops=1) 3 [...] Group Key: movie_cast.act_id 4 [...] Batches: 1 Memory Usage: 56kB 5 [...] > Nested Loop (cost=0.28..16.32 rows=1 width=4) (actual time=0.008..0.027 rows=222 loops=1) 6 [...] > Index Scan using kimelboss2 on movie (cost=0.00..8.02 rows=1 width=4) (actual time=0.005..0.006 rows=1 loops=1) 7 [...] Index Cond: (mov_title = 'Annie Hall') 8 [...] > Index Scan using kimelboss on movie_cast (cost=0.28..8.29 rows=1 width=8) (actual time=0.002..0.012 rows=222 loops=1) 9 [...] Index Cond: (mov_id = movie.mov_id) 10 [...] > 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) 11 [...] Index Cond: (act_id = movie_cast.act_id) 12 Planning Time: 0.364 ms 13 Execution Time: 0.240 ms

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:

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

The query in the Query Editor is:

```

1 set enable_seqscan=off;
2 create index kimelboss on movie_cast
3 using btree(mov_id);
4 create index kimelboss2 on movie
5 using hash (mov_title);
6 create index kimelboss3 on movie_cast
7 using btree(act_id);
8 explain analyze
9 select *
10 from actor

```

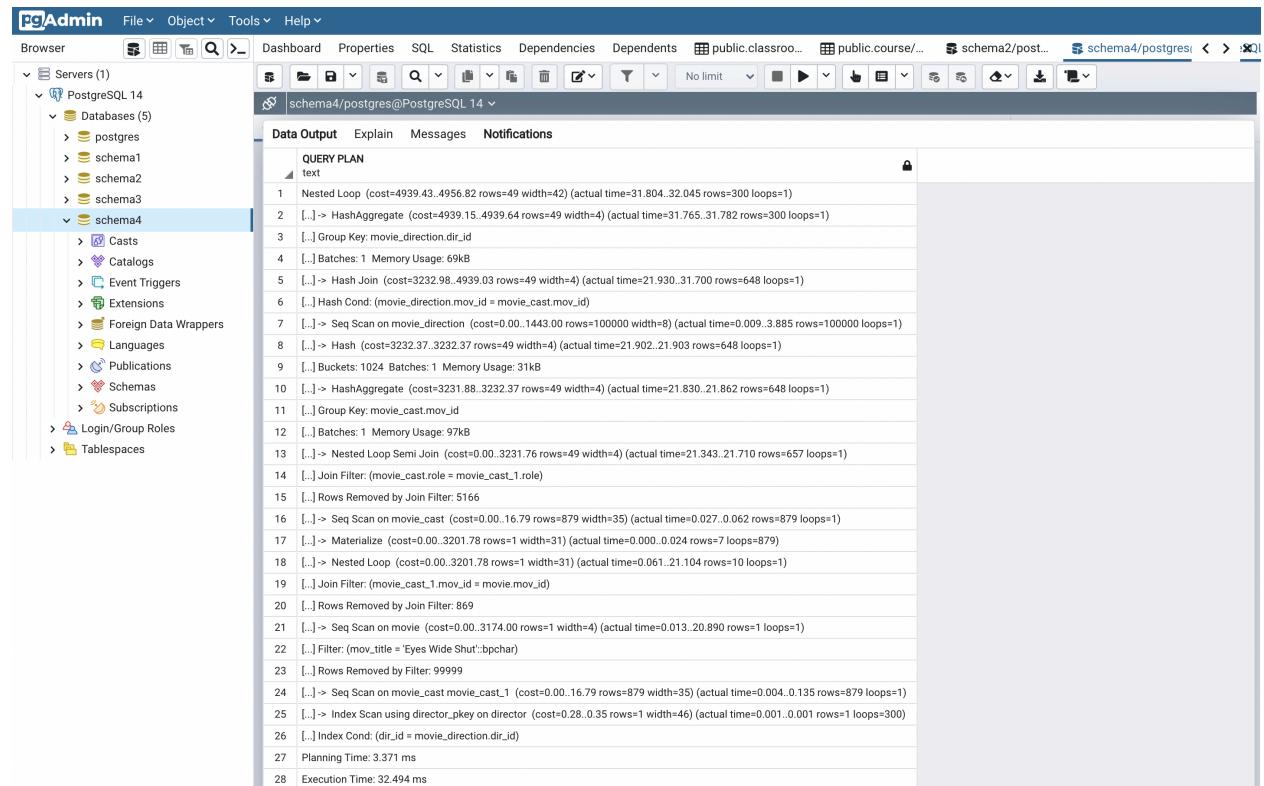
The Explain Analysis tab shows the execution plan:

#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=16.62..21.02 rows=1 width=48) (actual=1 rows=1)	0.405 ms	0.584 ms	↓ 222	222	1	1
2.	→ Aggregate (cost=16.32..16.33 rows=1 width=4) (actual=0.147... Batches: Memory Usage: 56 kB)	0.099 ms	0.178 ms	↓ 222	222	1	1
3.	→ Nested Loop Inner Join (cost=0.28..16.32 rows=1 width=48) (actual=1 rows=1)	0.029 ms	0.079 ms	↓ 222	222	1	1
4.	→ Index Scan using kimelboss2 on movie as movie (cost=0.29..4.6... Index Cond: (mov_title = 'Annie Hall'))	0.014 ms	0.014 ms	↑ 1	1	1	1
5.	→ Index Scan using kimelboss on movie_cast as movie_cast (cost=0.29..4.6... Index Cond: (mov_id = movie.mov_id))	0.037 ms	0.037 ms	↓ 222	222	1	1
6.	→ Index Scan using actor_pkey on actor as actor (cost=0.29..4.6... Index Cond: (act_id = movie_cast.act_id))	0.001 ms	0.001 ms	↑ 1	1	1	222

A message at the bottom right indicates: ✓ Successfully run. Total query runtime: 37 msec. 1 rows affected.

# Query 11 with no indices

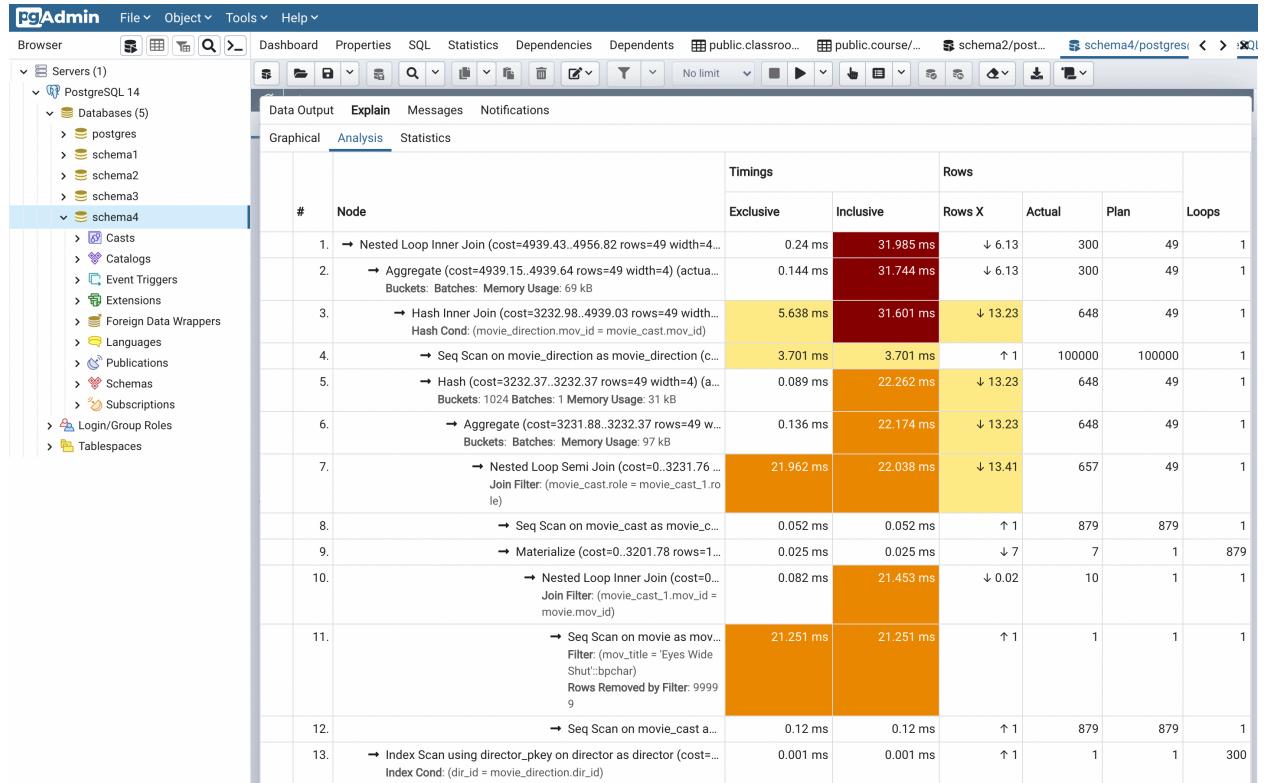
Planning and execution time :



The screenshot shows the PgAdmin 4 interface. On the left, the object browser displays a tree structure of servers, databases, and schema. The current schema selected is 'schema4'. In the main pane, the 'Data Output' tab is active, showing the 'QUERY PLAN' for a query. The plan consists of 28 numbered steps, starting with a Nested Loop and ending with an execution time of 32.494 ms. The steps include various operations like Hash Aggregate, Hash Join, Seq Scan, and Index Scan, many of which are marked with ellipses indicating they were not indexed.

Step	Operation	Cost	Rows	Width	Time
1	Nested Loop	(cost=4939.43..4956.82)	rows=49	width=42	(actual time=31.804..32.045)
2	[...] > HashAggregate	(cost=4939.15..4939.64)	rows=49	width=4	(actual time=31.765..31.782)
3	[...] Group Key: movie_direction.dir_id				
4	[...] Batches: 1 Memory Usage: 69kB				
5	[...] > Hash Join	(cost=3232.98..4939.03)	rows=49	width=4	(actual time=21.930..31.700)
6	[...] Hash Cond: (movie_direction.movie_id = movie_cast.movie_id)				
7	[...] > Seq Scan on movie_direction	(cost=0.00..1443.00)	rows=100000	width=8	(actual time=0.009..3.885)
8	[...] > Hash	(cost=3232.37..3232.37)	rows=49	width=4	(actual time=21.902..21.903)
9	[...] Buckets: 1024 Batches: 1 Memory Usage: 31kB				
10	[...] > HashAggregate	(cost=3231.88..3232.37)	rows=49	width=4	(actual time=21.830..21.862)
11	[...] Group Key: movie_cast.movie_id				
12	[...] Batches: 1 Memory Usage: 97kB				
13	[...] > Nested Loop Semi Join	(cost=0.00..3231.76)	rows=49	width=4	(actual time=21.343..21.710)
14	[...] Join Filter: (movie_cast.role = movie_cast_1.role)				
15	[...] Rows Removed by Join Filter: 5166				
16	[...] > Seq Scan on movie_cast	(cost=0.00..16.79)	rows=879	width=35	(actual time=0.027..0.062)
17	[...] > Materialize	(cost=0.00..3201.78)	rows=1	width=31	(actual time=0.000..0.024)
18	[...] > Nested Loop	(cost=0.00..3201.78)	rows=1	width=31	(actual time=0.061..21.104)
19	[...] Join Filter: (movie_cast_1.movie_id = movie.movie_id)				
20	[...] Rows Removed by Join Filter: 869				
21	[...] > Seq Scan on movie	(cost=0.00..3174.00)	rows=1	width=4	(actual time=0.013..20.890)
22	[...] Filter: (mov.title = 'Eyes Wide Shut':bpchar)				
23	[...] Rows Removed by Filter: 99999				
24	[...] > Seq Scan on movie_cast movie_cast_1	(cost=0.00..16.79)	rows=879	width=35	(actual time=0.004..0.135)
25	[...] > Index Scan using director_pkey on director	(cost=0.28..0.35)	rows=1	width=46	(actual time=0.001..0.001)
26	[...] Index Cond: (dir_id = movie_direction.dir_id)				
27	Planning Time: 3.371 ms				
28	Execution Time: 32.494 ms				

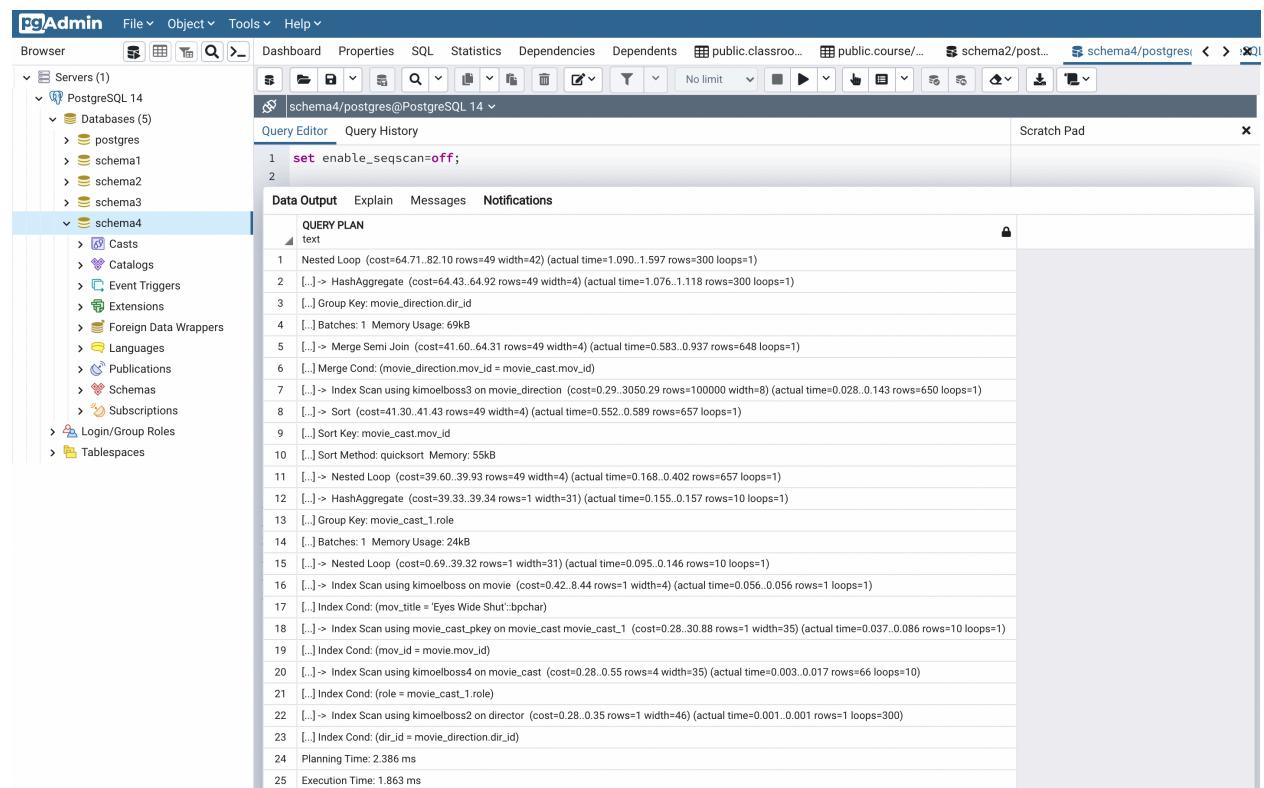
## Execution plan and costs:



## 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. 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. The main area is the 'Query Editor' tab, which contains the following SQL command:

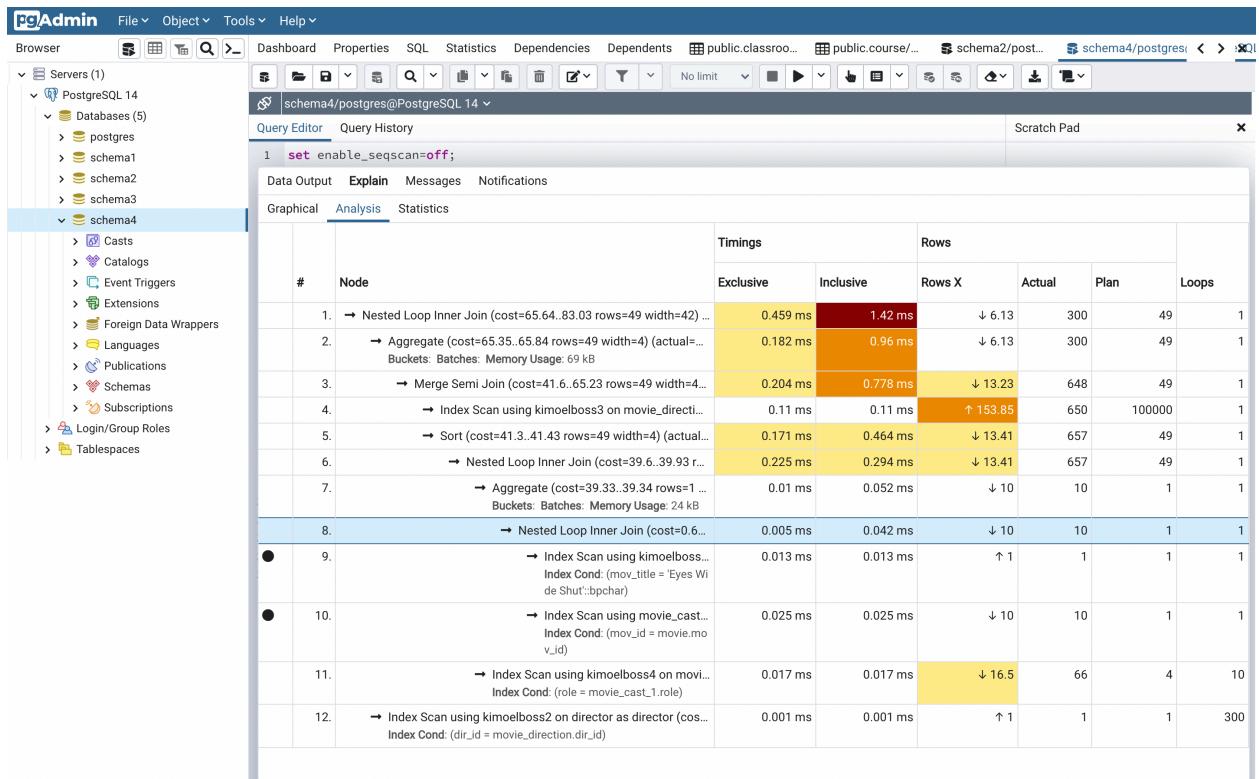
```
1 set enable_seqscan=off;
```

Below the query, the 'Data Output' tab is active, displaying the 'QUERY PLAN' for the executed query. The plan shows a nested loop join with the following steps:

- 1 Nested Loop (cost=64.71..82.10 rows=49 width=42) (actual time=1.090..1.597 rows=300 loops=1)
  - 2 [...] > HashAggregate (cost=64.43..64.92 rows=49 width=4) (actual time=1.076..1.118 rows=300 loops=1)
    - 3 [...] Group Key: movie\_direction.dir\_id
    - 4 [...] Batches: 1 Memory Usage: 69kB
    - 5 [...] > Merge Semi Join (cost=41.60..64.31 rows=49 width=4) (actual time=0.583..0.937 rows=648 loops=1)
    - 6 [...] Merge Cond: (movie\_direction.mov\_id = movie\_cast.mov\_id)
    - 7 [...] > Index Scan using kimodelboss3 on movie\_direction (cost=0.29..3050.29 rows=100000 width=8) (actual time=0.028..0.143 rows=650 loops=1)
    - 8 [...] > Sort (cost=41.30..41.43 rows=49 width=4) (actual time=0.552..0.589 rows=657 loops=1)
    - 9 [...] Sort Key: movie\_cast.mov\_id
    - 10 [...] Sort Method: quicksort Memory: 55kB
    - 11 [...] > Nested Loop (cost=39.60..39.93 rows=49 width=4) (actual time=0.168..0.402 rows=657 loops=1)
    - 12 [...] > HashAggregate (cost=39.33..39.34 rows=1 width=31) (actual time=0.155..0.157 rows=10 loops=1)
      - 13 [...] Group Key: movie\_cast.role
      - 14 [...] Batches: 1 Memory Usage: 24kB
      - 15 [...] > Nested Loop (cost=0.69..39.32 rows=1 width=31) (actual time=0.095..0.146 rows=10 loops=1)
      - 16 [...] > Index Scan using kimodelboss on movie (cost=0.42..8.44 rows=1 width=4) (actual time=0.056..0.056 rows=1 loops=1)
      - 17 [...] Index Cond: (mov\_title = 'Eyes Wide Shut'.bpchar)
      - 18 [...] > Index Scan using movie\_cast\_pkkey on movie\_cast movie\_cast\_1 (cost=0.28..30.88 rows=1 width=35) (actual time=0.037..0.086 rows=10 loops=1)
        - 19 [...] Index Cond: (mov\_id = movie.mov\_id)
      - 20 [...] > Index Scan using kimodelboss4 on movie\_cast (cost=0.28..0.55 rows=4 width=35) (actual time=0.003..0.017 rows=66 loops=10)
      - 21 [...] Index Cond: (role = movie\_cast.role)
      - 22 [...] > Index Scan using kimodelboss2 on director (cost=0.28..0.35 rows=1 width=46) (actual time=0.001..0.001 rows=1 loops=300)
      - 23 [...] Index Cond: (dir\_id = movie\_direction.dir\_id)
    - 24 Planning Time: 2.386 ms
    - 25 Execution Time: 1.863 ms

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(logn) time so it is better than seqscan

## Execution plan and costs:



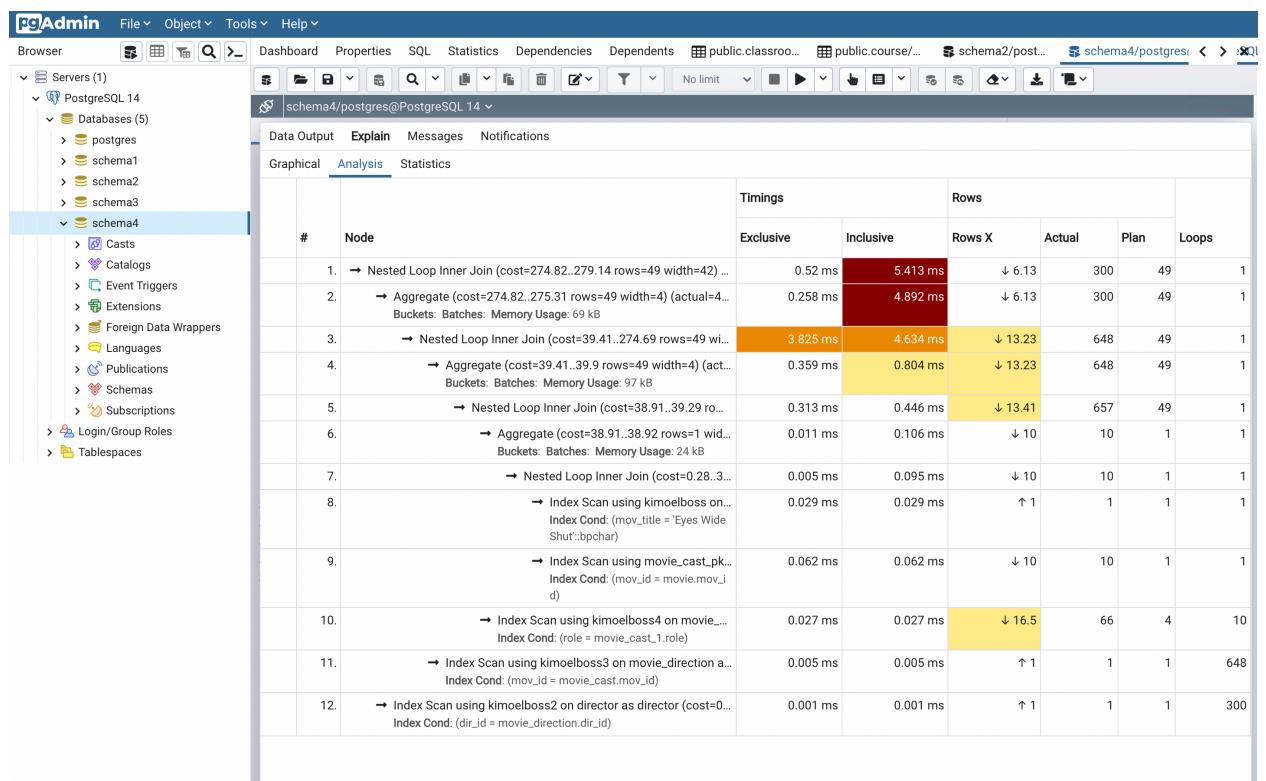
**Query 11 using Hash index on movie (mov\_title) , director (dir\_id), movie\_direction (mov\_id) , movie\_cast (role)**

## Disabled flags: seqscan

### **Planning and execution time :**

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 :**

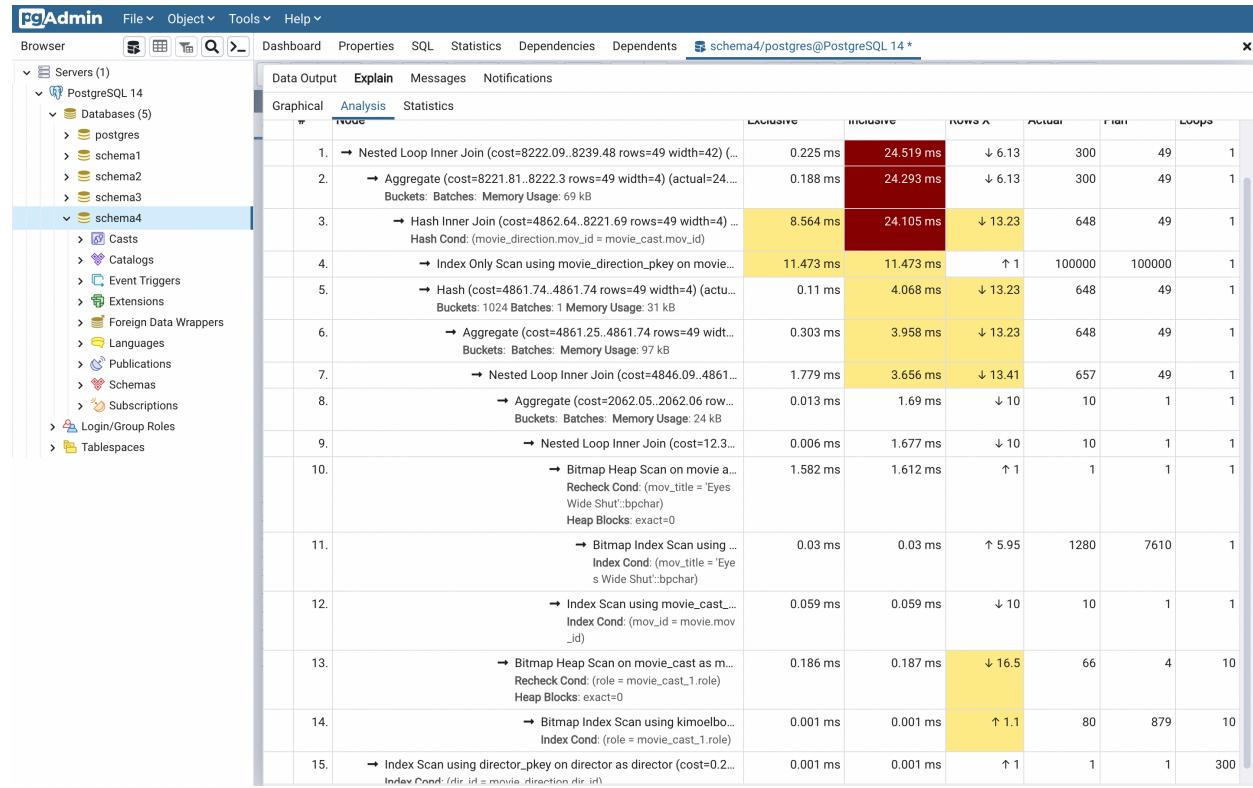
The screenshot shows the pgAdmin 4 interface with the following details:

- Left Panel (Servers):** Shows a tree view of database objects. The 'schema4' node is selected.
- Top Bar:** Includes File, Object, Tools, Help, Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a connection tab for schema4/postgres@PostgreSQL 14\*.
- Central Area:** A large window titled "Data Output" displays the "QUERY PLAN" for a query. The plan consists of 36 numbered steps, starting with a Nested Loop and ending with a Planning Time of 0.710 ms and an Execution Time of 12.831 ms.

```
1 Nested Loop (cost=8222.09..8239.48 rows=49 width=42) (actual time=12.386..12.634 rows=300 loops=1)
  2 [...] > HashAggregate (cost=8221.81..8222.30 rows=49 width=4) (actual time=12.364..12.383 rows=300 loops=1)
  3 [...] Group Key: movie_direction.dir_id
  4 [...] Batches: 1 Memory Usage: 69kB
  5 [...] > Hash Join (cost=4862.64..8221.69 rows=49 width=4) (actual time=1.782..12.296 rows=648 loops=1)
  6 [...] Hash Cond: (movie_direction.movie_id = movie_cast.movie_id)
  7 [...] > Index Only Scan using movie_direction_pkey on movie_direction (cost=0.29..3096.29 rows=100000 width=8) (actual time=0.009..5.808 rows=100000 loops=1)
  8 [...] Heap Fetches: 0
  9 [...] > Hash (cost=4861.74..4861.74 rows=49 width=4) (actual time=1.750..1.753 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=1.670..1.704 rows=648 loops=1)
 12 [...] Group Key: movie_cast.movie_id
 13 [...] Batches: 1 Memory Usage: 97kB
 14 [...] > Nested Loop (cost=4846.09..4861.12 rows=49 width=4) (actual time=0.670..1.534 rows=657 loops=1)
 15 [...] > HashAggregate (cost=2062.05..2062.06 rows=1 width=31) (actual time=0.627..0.630 rows=10 loops=1)
 16 [...] Group Key: movie_cast_1.role
 17 [...] Batches: 1 Memory Usage: 24kB
 18 [...] > Nested Loop (cost=12.31..2062.05 rows=1 width=31) (actual time=0.037..0.624 rows=10 loops=1)
 19 [...] > Bitmap Heap Scan on movie (cost=12.04..2031.16 rows=1 width=4) (actual time=0.017..0.595 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 kimelboss (cost=0.00..12.04 rows=7610 width=0) (actual time=0.013..0.014 rows=1280 loops=1)
 24 [...] Index Cond: (mov_title = 'Eyes Wide Shut':bpchar)
 25 [...] > Index Scan using movie_cast_pkey on movie_cast movie_cast_1 (cost=0.28..30.88 rows=1 width=35) (actual time=0.019..0.027 rows=1 loops=1)
 26 [...] Index Cond: (mov_id = movie.movie_id)
 27 [...] > Bitmap Heap Scan on movie_cast (cost=2784.03..2799.02 rows=4 width=35) (actual time=0.028..0.085 rows=66 loops=10)
 28 [...] Recheck Cond: (role = movie_cast_1.role)
 29 [...] Rows Removed by Index Recheck: 813
 30 [...] Heap Blocks: lossy=80
 31 [...] > Bitmap Index Scan on kimelboss4 (cost=0.00..2784.03 rows=879 width=0) (actual time=0.010..0.010 rows=80 loops=10)
 32 [...] Index Cond: (role = movie_cast_1.role)
 33 [...] > Index Scan using director_pkey on director (cost=0.28..0.35 rows=1 width=46) (actual time=0.001..0.001 rows=1 loops=300)
 34 [...] Index Cond: (dir_id = movie_direction.dir_id)
 35 Planning Time: 0.710 ms
 36 Execution Time: 12.831 ms
```

Here the BRIN increased the cost significantly as it is most suitable in small range queries in big ranges. However that was not the case and the performance is much bad now

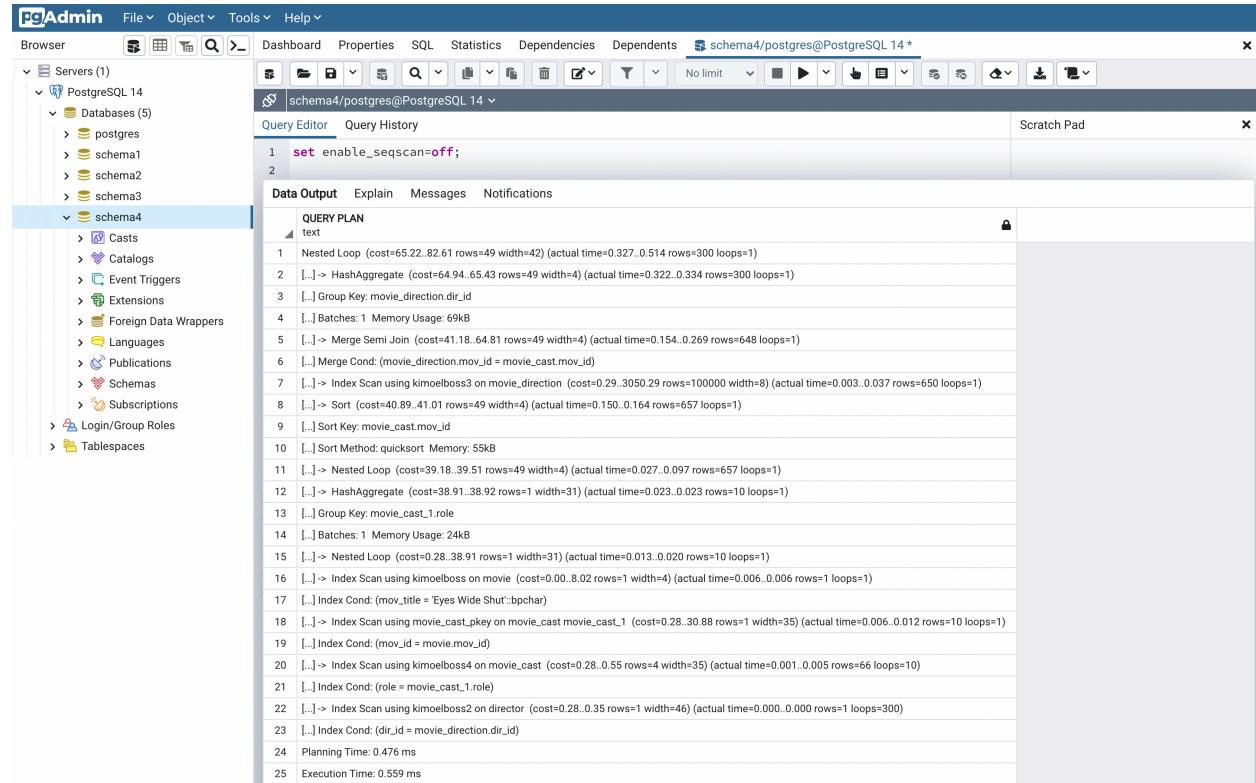
## Execution plan and costs:



# 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 :



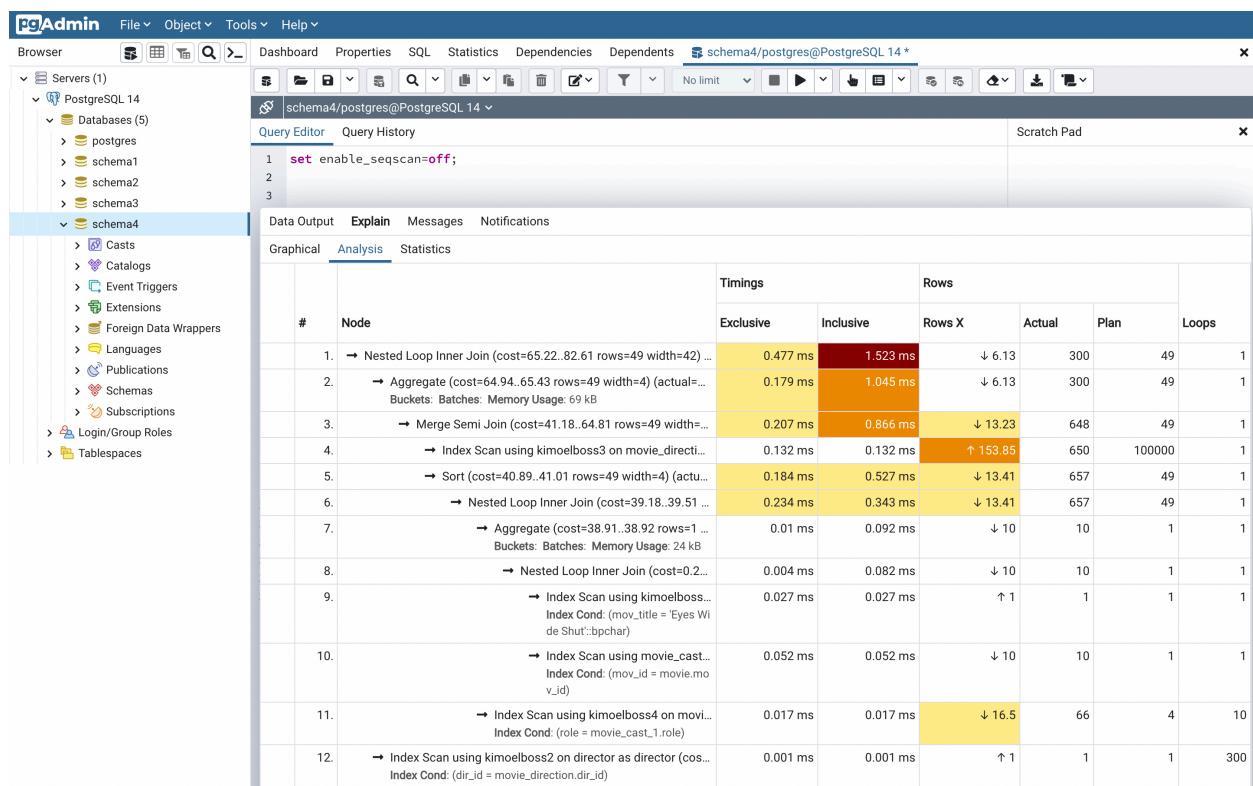
```
set enable_seqscan=off;
```

The screenshot shows the PgAdmin 4 interface. In the left sidebar, under 'Servers (1)', there is one server entry: 'PostgreSQL 14'. Under 'Databases (5)', there are five databases: 'postgres', 'schema1', 'schema2', 'schema3', and 'schema4'. 'schema4' is currently selected. The main area is the 'Query Editor' tab, which contains the command 'set enable\_seqscan=off;'. Below the editor is the 'Data Output' tab, which displays the 'EXPLAIN PLAN' for the query. The plan shows a detailed breakdown of the query's execution, including nested loops, hash aggregates, and index scans, all utilizing B+ trees and hash indexes.

Step	Operation	Cost	Rows	Width	Time
1	Nested Loop	65.22	82 61	49	actual time=0.327..0.514
2	[...] > HashAggregate	64.94	65 43	49	actual time=0.322..0.334
3	[...] Group Key: movie_direction.dir_id				
4	[...] Batches: 1				Memory Usage: 69kB
5	[...] > Merge Semi Join	41.18	64.81	49	actual time=0.154..0.269
6	[...] Merge Cond: (movie_direction.mov_id = movie_cast.mov_id)				
7	[...] > Index Scan using kimelboss3 on movie_direction	0.29	3050.29	8	actual time=0.003..0.037
8	[...] > Sort	40.89	41.01	49	actual time=0.150..0.164
9	[...] Sort Key: movie_cast.mov_id				
10	[...] Sort Method: quicksort				Memory: 55kB
11	[...] > Nested Loop	39.18	39.51	49	actual time=0.027..0.097
12	[...] > HashAggregate	38.91	38.92	31	actual time=0.023..0.023
13	[...] Group Key: movie_cast_1.role				
14	[...] Batches: 1				Memory Usage: 24kB
15	[...] > Nested Loop	0.28	38.91	31	actual time=0.013..0.020
16	[...] > Index Scan using kimelboss on movie	0.00	8.02	1	width=4
17	[...] Index Cond: (mov_title = 'Eyes Wide Shut':bpchar)				
18	[...] > Index Scan using movie_cast_pkey on movie_cast movie_cast_1	0.28	30.88	35	actual time=0.006..0.012
19	[...] Index Cond: (mov_id = movie.mov_id)				
20	[...] > Index Scan using kimelboss4 on movie_cast	0.28	0.55	4	width=35
21	[...] Index Cond: (role = movie_cast_1.role)				
22	[...] > Index Scan using kimelboss2 on director	0.28	0.35	46	actual time=0.000..0.000
23	[...] Index Cond: (dir_id = movie_direction.dir_id)				
24	Planning Time:	0.476	ms		
25	Execution Time:	0.559	ms		

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

## Execution plan and costs:



## Query 12 no index

Planning and execution time :

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

- Browser:** Shows a tree view of the database structure under "PostgreSQL 14". The "schema4" node is selected.
- Query Editor:** Contains the following SQL code:

```
1 set enable_seqscan=on;
2
3
4 explain analyze
5 select mov_title
6   from movie
7  where mov_id in (
8    select mov_id
9      from movie_direction
10     where dir_id=
11       (select dir_id
12         from director
13           where dir_fname = 'Woddy' and dir_lname='Allen'));
```
- Data Output:** Displays the query plan in text format:

```
1 Nested Loop (cost=147.59..292.86 rows=17 width=51) (actual time=1.539..2.108 rows=350 loops=1)
2 [...] InitPlan 1 (returns $0)
3 [...] > Seq Scan on director (cost=0.00..147.00 rows=1 width=4) (actual time=0.023..1.514 rows=1 loops=1)
4 [...] Filter: ((dir_fname = 'Woddy') AND (dir_lname = 'Allen'))
5 [...] Rows Removed by Filter: 5999
6 [...] > Index Only Scan using movie_direction_pkey on movie_direction (cost=0.29..4.59 rows=17 width=4) (actual time=1.532..1.574 rows=350 loops=1)
7 [...] Index Cond: (dir_id = $0)
8 [...] Heap Fetches: 0
9 [...] > Index Scan using movie_pkey on movie (cost=0.29..8.31 rows=1 width=55) (actual time=0.001..0.001 rows=1 loops=350)
10 [...] Index Cond: (mov_id = movie_direction.mov_id)
11 Planning Time: 0.416 ms
12 Execution Time: 2.157 ms
```

## Execution plan and costs:

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. The main area has tabs for 'Query Editor' (selected) and 'Query History'. The 'Query Editor' tab contains the following SQL code:

```
1 set enable_seqscan=on;
2
3
4 explain analyze
5 select mov_title
6   from movie
7 where mov_id in (
8       select mov_id
9         from movie_direction
10        where dir_id=
11            (select dir_id
12              from director
13             where dir_fname = 'Woddy' and dir_lname='Allen'));
```

Below the code, there are tabs for 'Data Output', 'Explain' (selected), 'Messages', and 'Notifications'. Under 'Explain', there are three tabs: 'Graphical' (selected), 'Analysis' (selected), and 'Statistics'. The 'Analysis' tab displays the execution plan in a table:

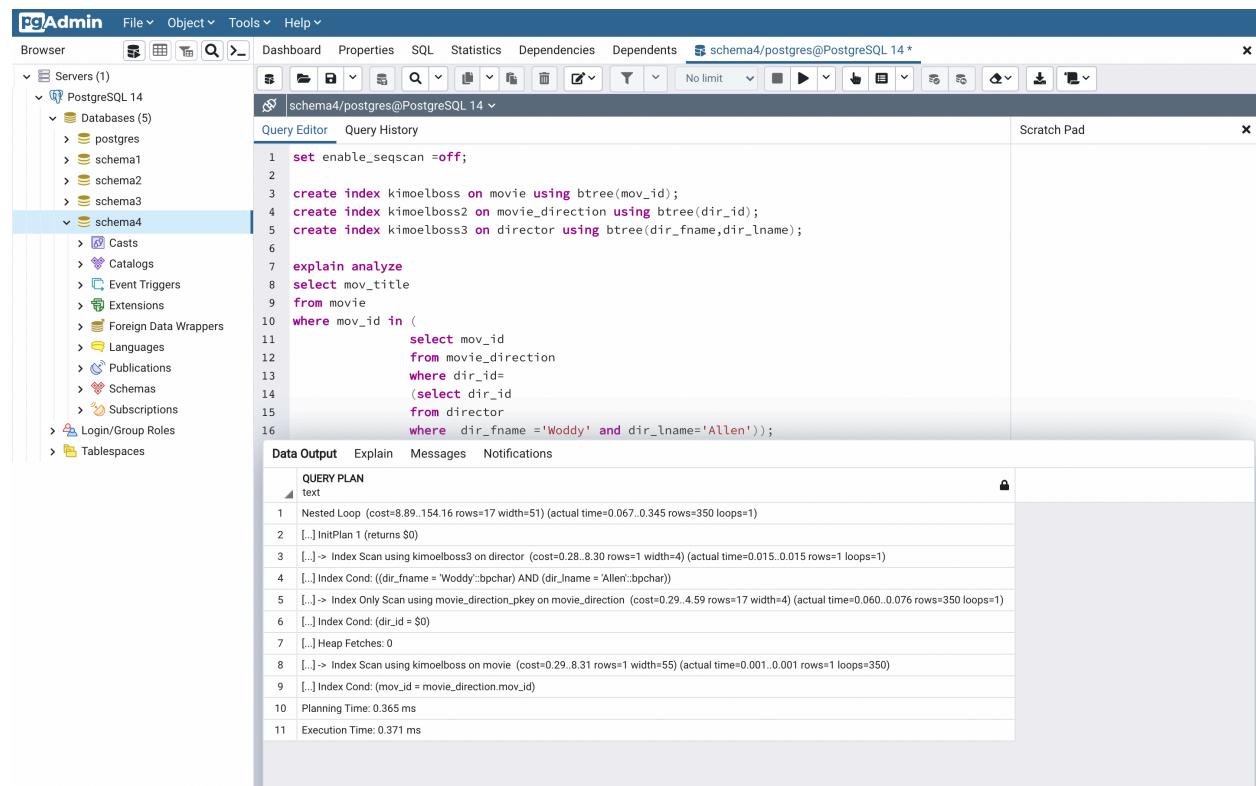
#	Node	Timings			Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan		
1.	→ Nested Loop Inner Join (cost=147.59..292.86 rows=17 width=51) (...) 1. → Seq Scan on director as director (cost=0..147 rows=1 width=4... Filter: ((dir_fname = 'Woddy')::bpchar) AND (dir_lname = 'Allen')::bpcha... Rows Removed by Filter: 5999	0.876 ms	2.381 ms	↓ 20.59	350	17	1	
2.	→ Index Only Scan using movie_direction_pkey on movie_direction... Index Cond: (dir_id = \$0)	1.503 ms	1.503 ms	↑ 1	1	1	1	
3.	→ Index Scan using movie_pkey on movie as movie (cost=0.29.... Index Cond: (mov_id = movie_direction.mov_id)	1.695 ms	1.695 ms	↓ 20.59	350	17	1	
4.	→ Index Scan using movie_pkey on movie as movie (cost=0.29.... Index Cond: (mov_id = movie_direction.mov_id)	0.002 ms	0.002 ms	↑ 1	1	1	350	

A green success message at the bottom right says: 'Successfully run. Total query runtime: 43 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 with the following details:

- Browser:** Shows a tree view of servers, databases, and schema. Schema4 is selected.
- Query Editor:** Contains 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
9 from movie
10 where mov_id in (
11     select mov_id
12         from movie_direction
13         where dir_id=
14             (select dir_id
15                 from director
16                 where dir_fname = 'Woddy' and dir_lname='Allen'));
```
- Data Output:** Shows the **QUERY PLAN** section with the following plan:

```
1 Nested Loop (cost=8.89..154.16 rows=17 width=51) (actual time=0.067..0.345 rows=350 loops=1)
2 [...] InitPlan 1 returns $0
3 [...] > Index Scan using kimoelboss3 on director (cost=0.28..8.30 rows=1 width=4) (actual time=0.015..0.015 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 (cost=0.29..4.59 rows=17 width=4) (actual time=0.060..0.076 rows=350 loops=1)
6 [...] Index Cond: (dir_id = $0)
7 [...] Heap Fetches: 0
8 [...] > Index Scan using kimoelboss on movie (cost=0.29..8.31 rows=1 width=55) (actual time=0.001..0.001 rows=1 loops=350)
9 [...] Index Cond: (mov_id = movie_direction.mov_id)
10 Planning Time: 0.365 ms
11 Execution Time: 0.371 ms
```

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.

The query in the Query Editor is:

```

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
9   from movie
10 where mov_id in (
11       select mov_id
12         from movie_direction
13       where dir_id=
14           (select dir_id
15             from director
16           where dir_fname = 'Woddy' and dir_lname='Allen'));

```

The Explain tab shows the execution plan:

#	Node	Timings	Rows				
		Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Nested Loop Inner Join (cost=8.89..154.16 rows=17 width=51) (act...	0.684 ms	0.932 ms	↓ 20.59	350	17	1
2.	→ Index Scan using kimoelboss3 on director as director (cost=0... Index Cond: ((dir_fname = 'Woddy')::bpchar) AND (dir_lname = 'Allen')::bpchar)	0.039 ms	0.039 ms	↑ 1	1	1	1
3.	→ Index Only Scan using movie_direction_pkey on movie_direction... Index Cond: (dir_id = \$0)	0.207 ms	0.207 ms	↓ 20.59	350	17	1
4.	→ Index Scan using kimoelboss on movie as movie (cost=0.29..8... Index Cond: (mov_id = movie_direction.mov_id)	0.002 ms	0.002 ms	↑ 1	1	1	350

## 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. The left sidebar displays the database structure under 'Servers (1) > PostgreSQL 14 > schema4'. The 'Query Editor' tab is active, containing the following SQL code:

```
1 set enable_seqscan =off;
2
3 create index kimoelboss on movie using hash(mov_id);
4 create index kimoelboss2 on movie_direction using hash(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
11   from movie
12 where mov_id in (
13       select mov_id
14         from movie_direction
15       where dir_id=
16           (select dir_id
```

The 'Explain' tab is selected, showing the following query plan:

QUERY PLAN
1 Nested Loop (cost=8.31..148.91 rows=17 width=51) (actual time=0.038..0.370 rows=350 loops=1) 2 [...] InitPlan 1 (returns \$0) 3 [...] > Index Scan using kimoelboss4 on director (cost=0.00..8.02 rows=1 width=4) (actual time=0.006..0.007 rows=1 loops=1) 4 [...] Index Cond: (dir_lname = 'Allen') 5 [...] Filter: (dir_fname = 'Woody') 6 [...] > Index Only Scan using movie_direction_pkey on movie_direction (cost=0.29..4.59 rows=17 width=4) (actual time=0.034..0.046 rows=350 loops=1) 7 [...] Index Cond: (dir_id = \$0) 8 [...] Heap Fetches: 0 9 [...] > Index Scan using kimoelboss on movie (cost=0.00..8.02 rows=1 width=55) (actual time=0.001..0.001 rows=1 loops=350) 10 [...] Index Cond: (mov_id = movie_direction.mov_id) 11 Planning Time: 0.312 ms 12 Execution Time: 0.419 ms

A green success message at the bottom right of the explain plan area reads: "Successfully run. Total query runtime: 341 msec. 12 rows affected."

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:

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 hash(mov_id);
4 create index kimoelboss2 on movie_direction using hash(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
11 from movie
12 where mov_id in (
13     select mov_id
14         from movie_direction
15         where dir_id=
16             (select dir_id
    
```

**Explain Plan:**

#	Node	Timings	Rows				
		Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Nested Loop Inner Join (cost=8.31..148.91 rows=17 width=51) (actual= 350 rows=17)	0.809 ms	0.966 ms	↓ 20.59	350	17	1
2.	→ Index Scan using kimoelboss4 on director as director (cost=0.00..0.00 rows=1 width=51) (actual= 1 rows=1)	0.023 ms	0.023 ms	↑ 1	1	1	1
3.	→ Index Only Scan using movie_direction_pkey on movie_direction (cost=0.00..0.00 rows=1 width=51) (actual= 1 rows=1)	0.132 ms	0.132 ms	↓ 20.59	350	17	1
4.	→ Index Scan using kimoelboss on movie as movie (cost=0..8.02 rows=17 width=51) (actual= 350 rows=17)	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 :

```
set enable_seqscan =off;
create index kimoelboss on director using brin(dir_fname);
create index kimoelboss4 on director using brin(dir_lname);
explain analyze
select mov_title
from movie
where mov_id < 100;
```

DATA OUTPUT Explain Messages Notifications

QUERY PLAN

```
text
1 Nested Loop (cost=159.62..304.89 rows=17 width=51) (actual time=1.184..1.751 rows=350 loops=1)
2 [...] InitPlan 1 (returns $0)
3 [...] > Bitmap Heap Scan on director (cost=12.03..159.03 rows=1 width=4) (actual time=0.025..1.080 rows=1 loops=1)
4 [...] Recheck Cond: (dir_lname = 'Allen'\:bpchar)
5 [...] Rows Removed by Index Recheck: 5999
6 [...] Filter: (dir_fname = 'Woddy'\:bpchar)
7 [...] Heap Blocks: lossy=57
8 [...] > Bitmap Index Scan on kimoelboss4 (cost=0.00..12.03 rows=6000 width=0) (actual time=0.017..0.017 rows=570 loops=1)
9 [...] Index Cond: (dir_lname = 'Allen'\:bpchar)
10 [...] > Index Only Scan using movie_direction_pkey on movie_direction (cost=0.29..4.59 rows=17 width=4) (actual time=1.162..1.204 rows=350 loops=1)
11 [...] Index Cond: (dir_id = $0)
12 [...] Heap Fetches: 0
13 [...] > Index Scan using movie_pkey on movie (cost=0.29..8.31 rows=1 width=55) (actual time=0.001..0.001 rows=1 loops=350)
14 [...] Index Cond: (mov_id = movie_direction.mov_id)
15 Planning Time: 0.528 ms
16 Execution Time: 1.808 ms
```

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 and costs for a query.

The query in the Query Editor is:

```

1 set enable_seqscan =off;
2
3
4
5 create index kimodelboss3 on director using brin(dir_fname);
6 create index kimodelboss4 on director using brin(dir_lname);
7
8
9 explain analyze
10 select mov_title
11 from movie
12 where mov_id = 1
  
```

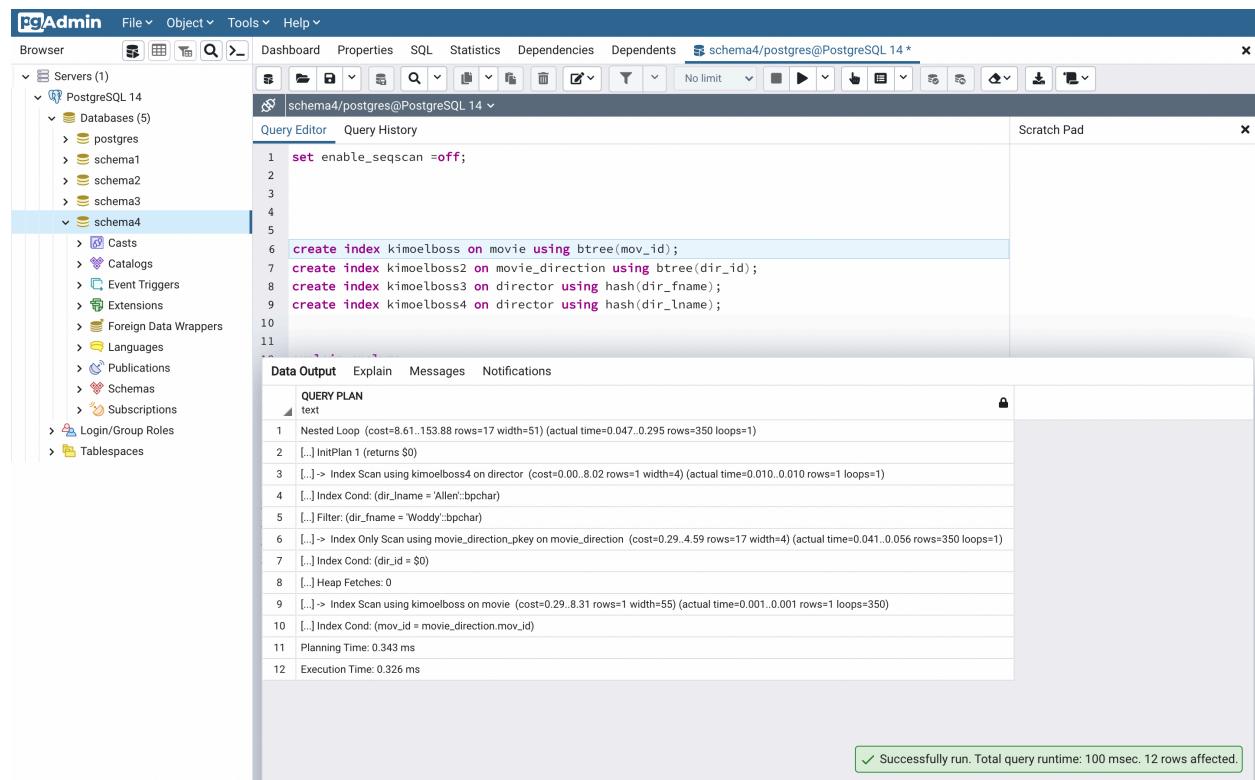
The Explain Analysis tab shows the execution plan:

#	Node	Timings	Rows				
		Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Nested Loop Inner Join (cost=159.62..304.89 rows=17 width=51) (...)	0.623 ms	1.662 ms	↓ 20.59	350	17	1
2.	→ Bitmap Heap Scan on director as director (cost=12.03..159.03...) (...) Filter: (dir_fname = 'Woody'\:bpchar) Rows Removed by Filter: 0 Recheck Cond: (dir_lname = 'Allen'\:bpchar) Heap Blocks: exact=0	1.026 ms	1.038 ms	↑ 1	1	1	1
3.	→ Bitmap Index Scan using kimodelboss4 (cost=0..12.03 rows=1 width=1) (...) Index Cond: (dir_lname = 'Allen'\:bpchar)	0.012 ms	0.012 ms	↑ 10.53	570	6000	1
4.	→ Index Only Scan using movie_direction_pkey on movie_director... (...) Index Cond: (dir_id = \$0)	1.113 ms	1.113 ms	↓ 20.59	350	17	1
5.	→ Index Scan using movie_pkey on movie as movie (cost=0.29.... (...) Index Cond: (mov_id = movie_direction.mov_id)	0.001 ms	0.001 ms	↑ 1	1	1	350

## 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. On the left, the server tree shows 'PostgreSQL 14' with 'Databases (5)' containing 'postgres', 'schema1', 'schema2', 'schema3', and 'schema4'. The 'schema4' node is selected. In the center, the 'Query Editor' tab is active, displaying the following SQL code:

```
1 set enable_seqscan =off;
2
3
4
5
6 create index kimoelboss on movie using btree(mov_id);
7 create index kimoelboss2 on movie_direction using btree(dir_id);
8 create index kimoelboss3 on director using hash(dir_fname);
9 create index kimoelboss4 on director using hash(dir_lname);
10
11
```

Below the query editor, the 'Data Output' tab is selected, showing the 'QUERY PLAN' for the executed query. The plan details a nested loop join with specific index scans and heap fetches. A message at the bottom right indicates the query was successfully run with a total runtime of 100 msec and 12 rows affected.

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

## Execution plan and costs:

pgAdmin 4.18 - PostgreSQL 14

File Object Tools Help

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

Query Editor Query History

```

1 set enable_seqscan =off;
2
3
4
5
6 create index kimoelboss on movie using btree(mov_id);
7 create index kimoelboss2 on movie_direction using btree(dir_id);
8 create index kimoelboss3 on director using hash(dir_fname);
9 create index kimoelboss4 on director using hash(dir_lname);
10
11
12 explain analyze
13 select mov_title
  
```

Data Output Explain Messages Notifications

Graphical Analysis Statistics

#	Node	Timings		Rows			
		Exclusive	Inclusive	Rows X	Actual	Plan	Loops
1.	→ Nested Loop Inner Join (cost=8.61..153.88 rows=17 width=51) (actual= 350 rows=17 loops=1)	0.549 ms	0.725 ms	↓ 20.59	350	17	1
2.	→ Index Scan using kimoelboss4 on director as director (cost=0.00..0.03 rows=1 width=51) Filter: (dir_fname = 'Woody'\:bpchar) Index Cond: (dir_lname = 'Allen'\:bpchar) Rows Removed by Filter: 0	0.031 ms	0.031 ms	↑ 1	1	1	1
3.	→ Index Only Scan using movie_direction_pkey on movie_direction (cost=0.29..0.30 rows=1 width=51) Index Cond: (dir_id = \$0)	0.144 ms	0.144 ms	↓ 20.59	350	17	1
4.	→ Index Scan using kimoelboss on movie as movie (cost=0.29..0.30 rows=1 width=51) Index Cond: (mov_id = movie_direction.mov_id)	0.001 ms	0.001 ms	↑ 1	1	1	350

Successfully run. Total query runtime: 43 msec. 1 rows affected.