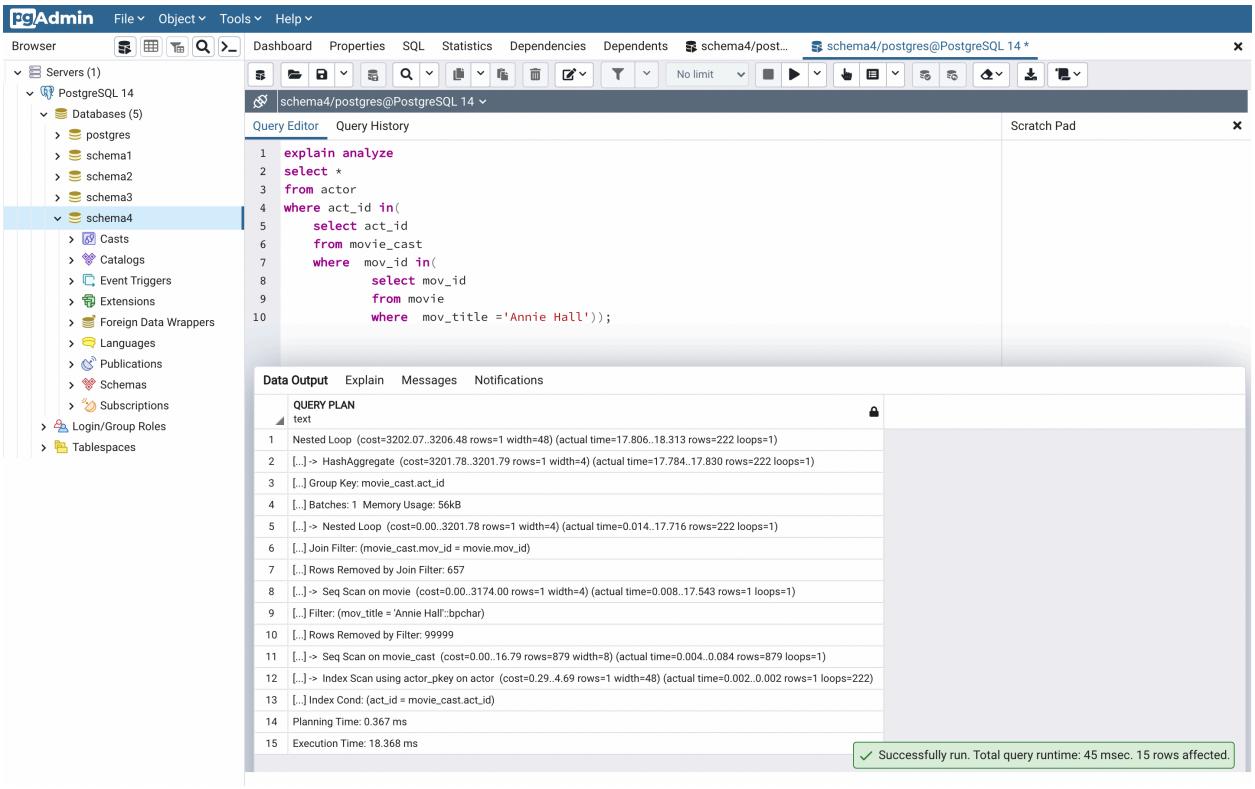
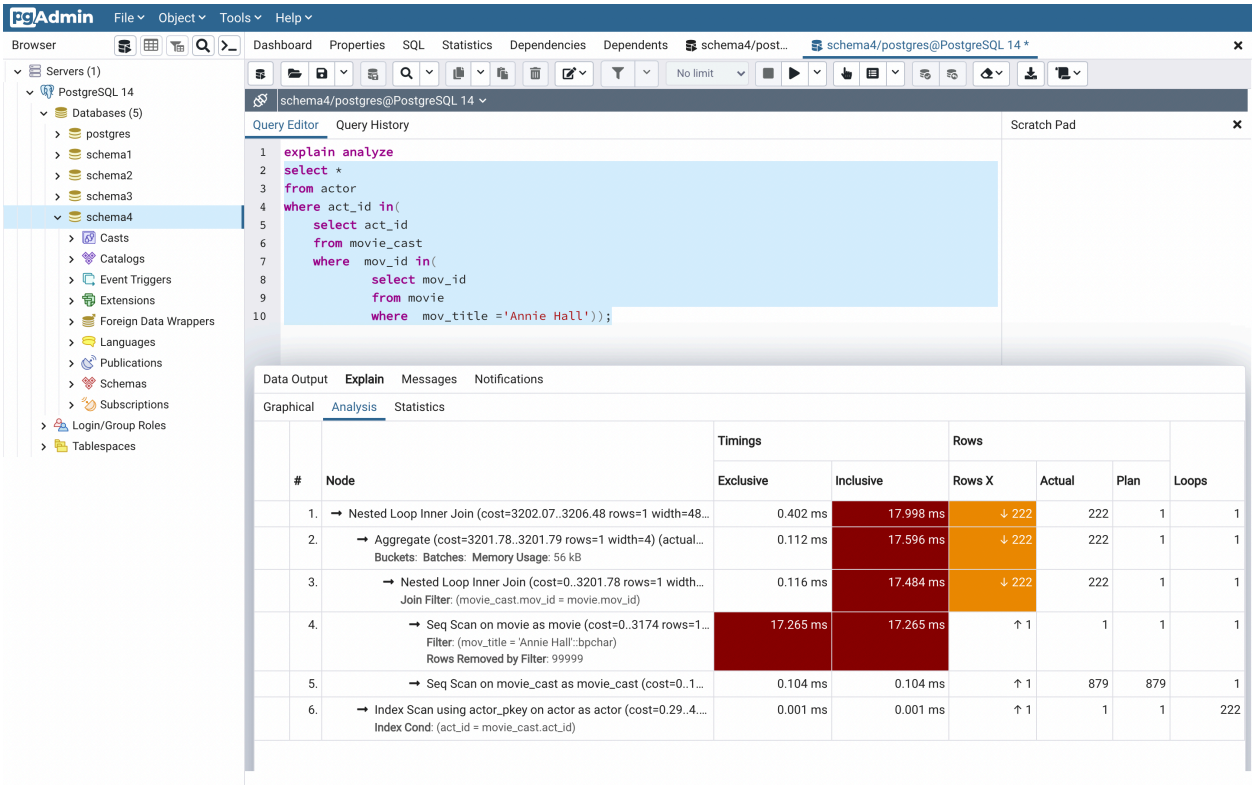
Query 10 with no indecies

Planning and execution time :



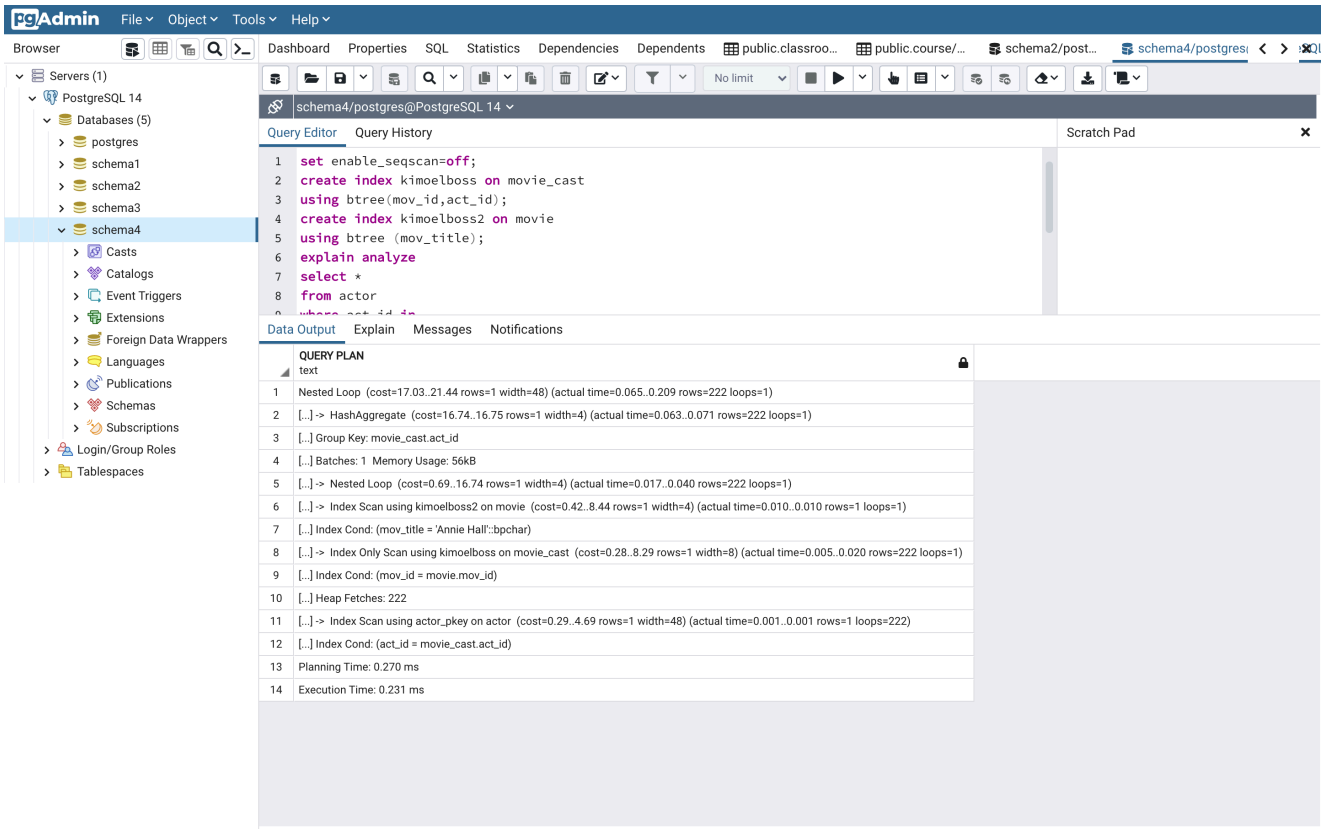
Execution Plan and costs :



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

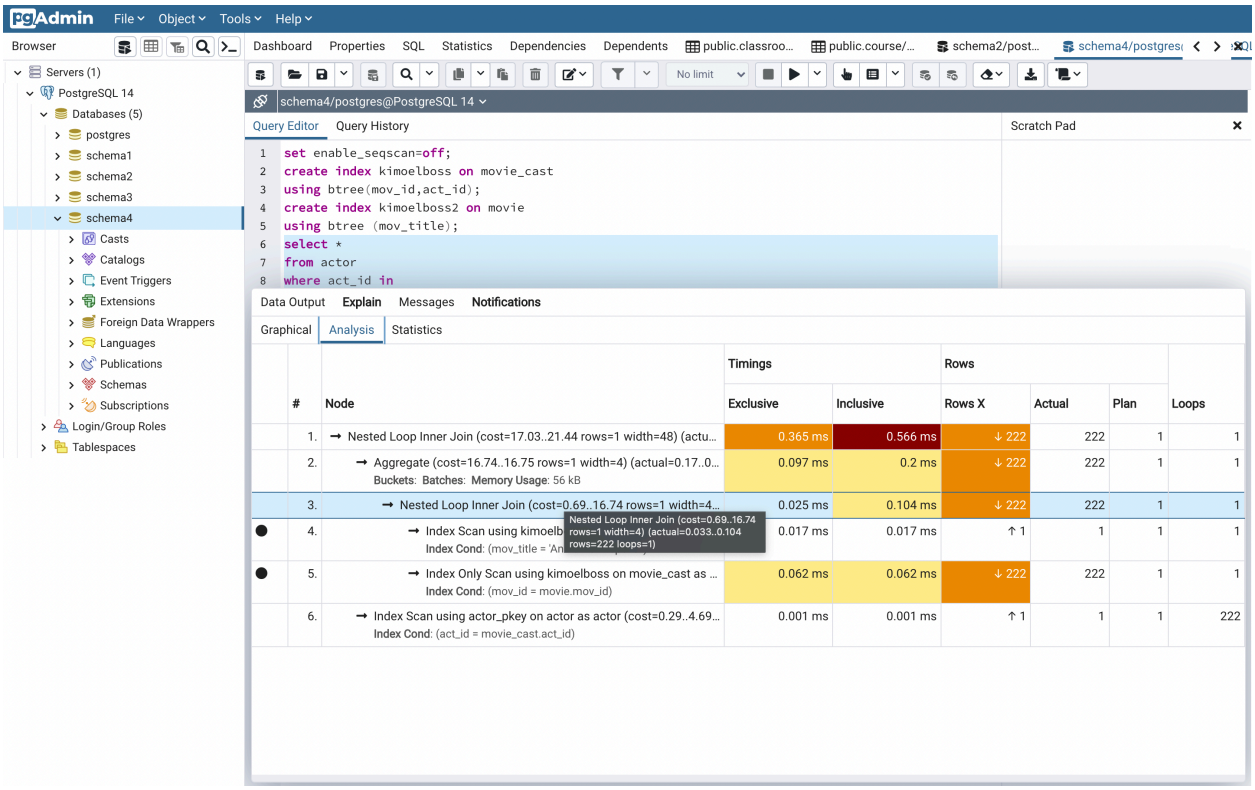
Disabled flags: seqscan

Planning and execution time :



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

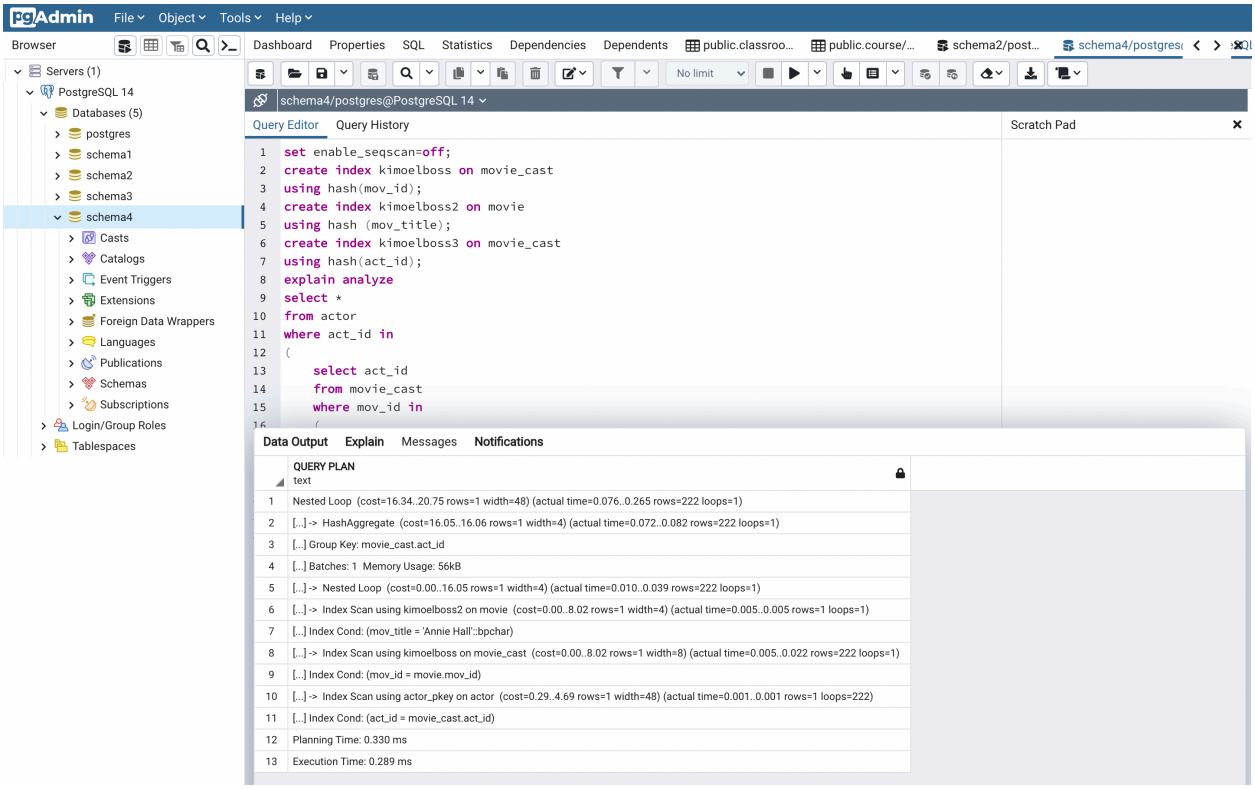
Execution plan and costs:



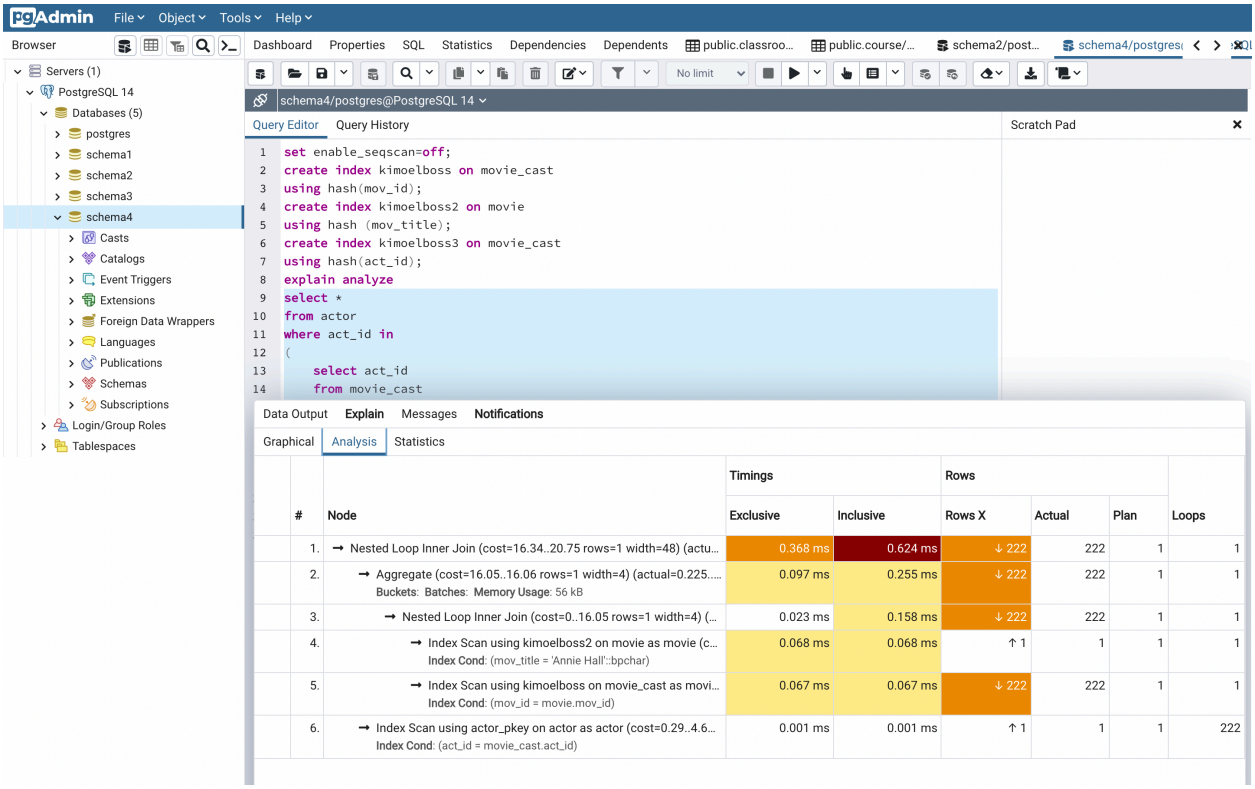
Query 10 using Hash index on : movie_cast (mov_id), , movie_cast(act_id)), movie (mov_title)

Disabled flags: seqscan

Planning and execution time :



Execution plan and costs:



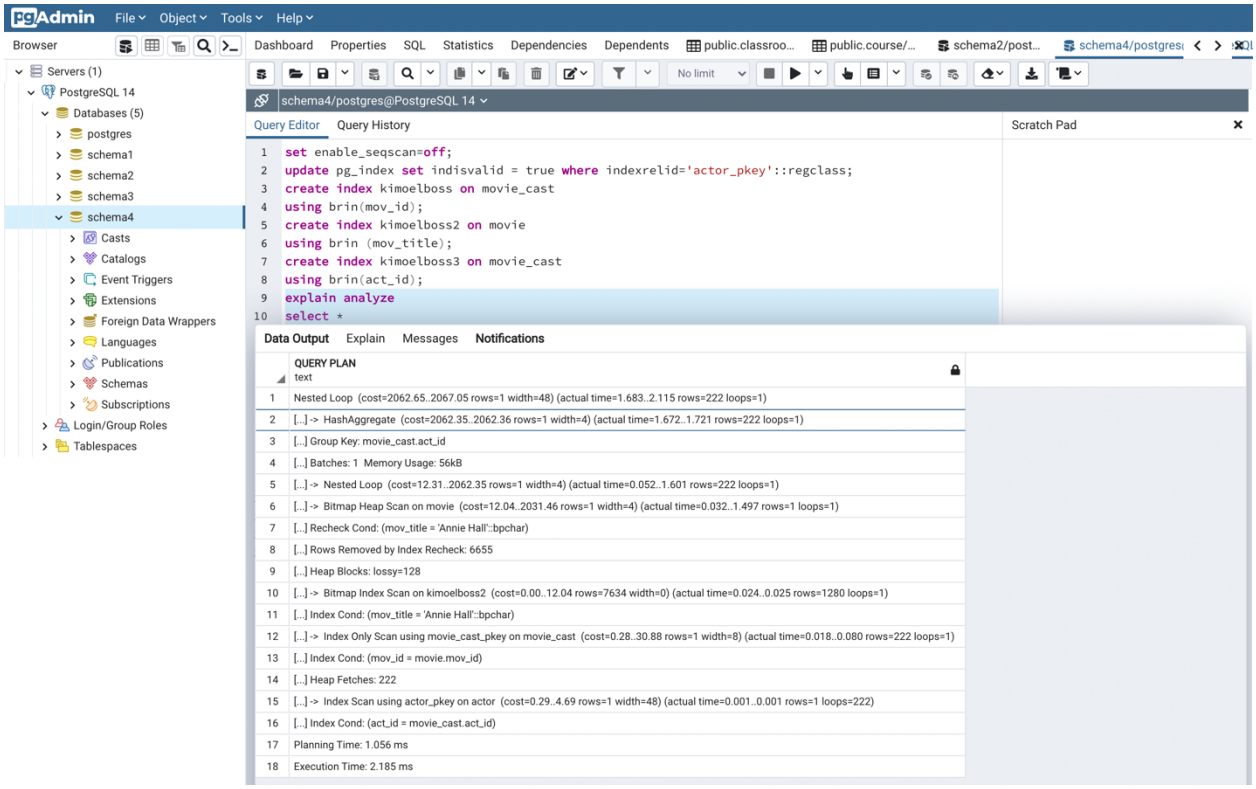
Here the hash index increased performance as it search in O(1) time so it helped in searching from mov_title=”Annie Hall” and mov_id and act_id

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

Disabled flags: seqscan

Planning and execution time :

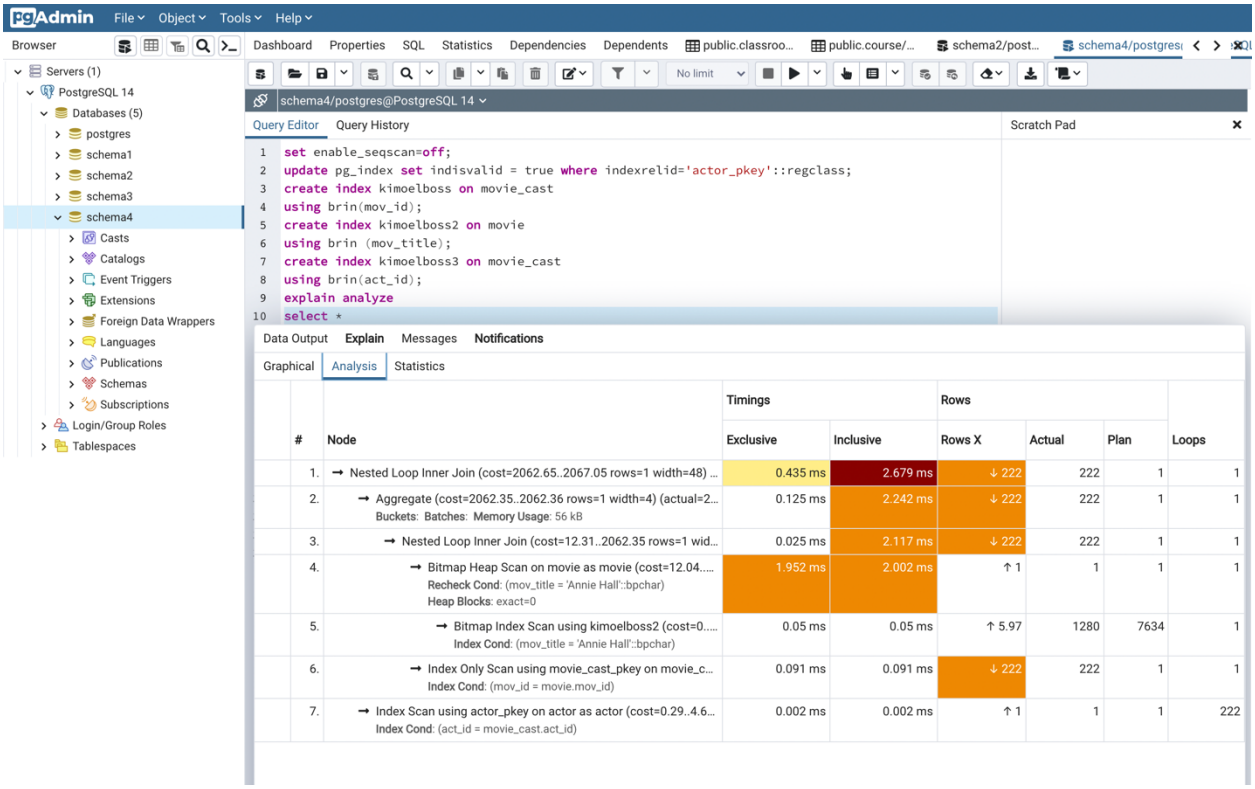
Without disabling the primary key



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

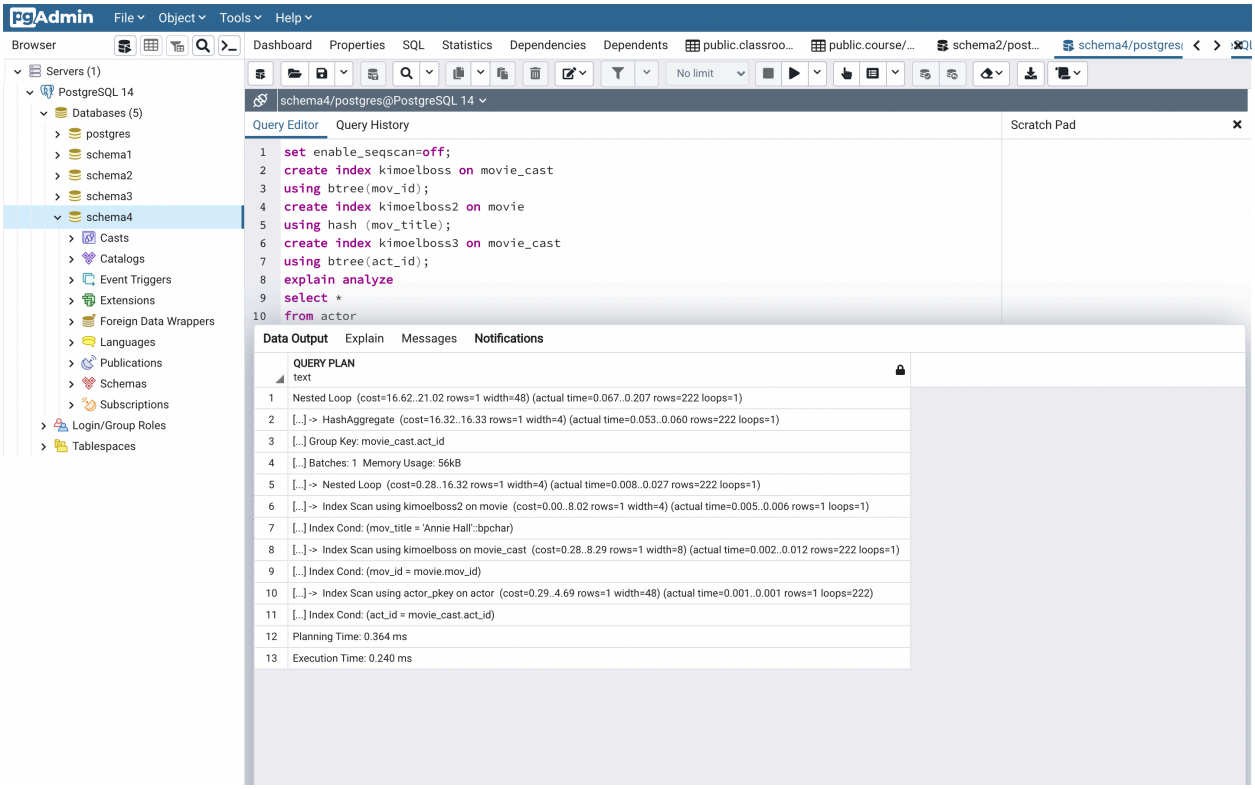


Query 10 using MIX index on : Btree-> movie_cast (mov_id), , movie_cast(act_id)),

Hash -> movie (mov_title)

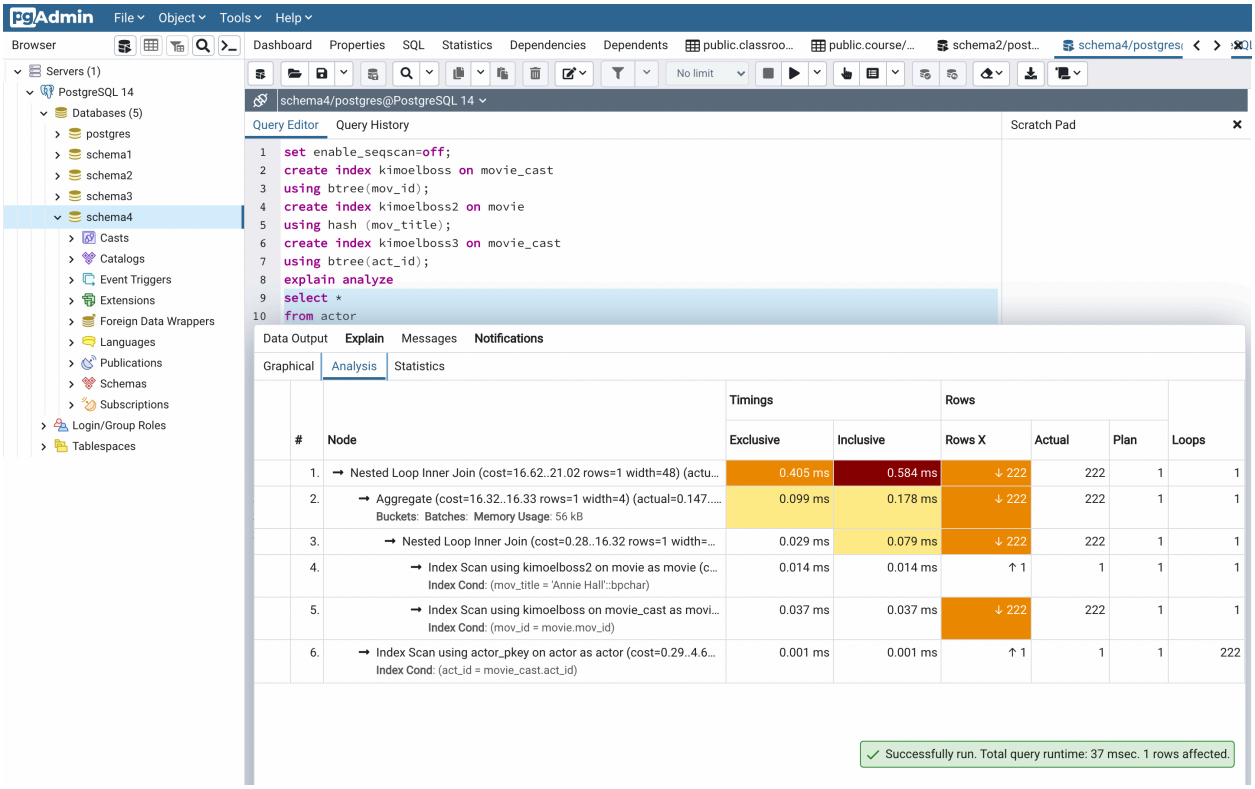
Disabled flags: seqscan

Planning and execution time :



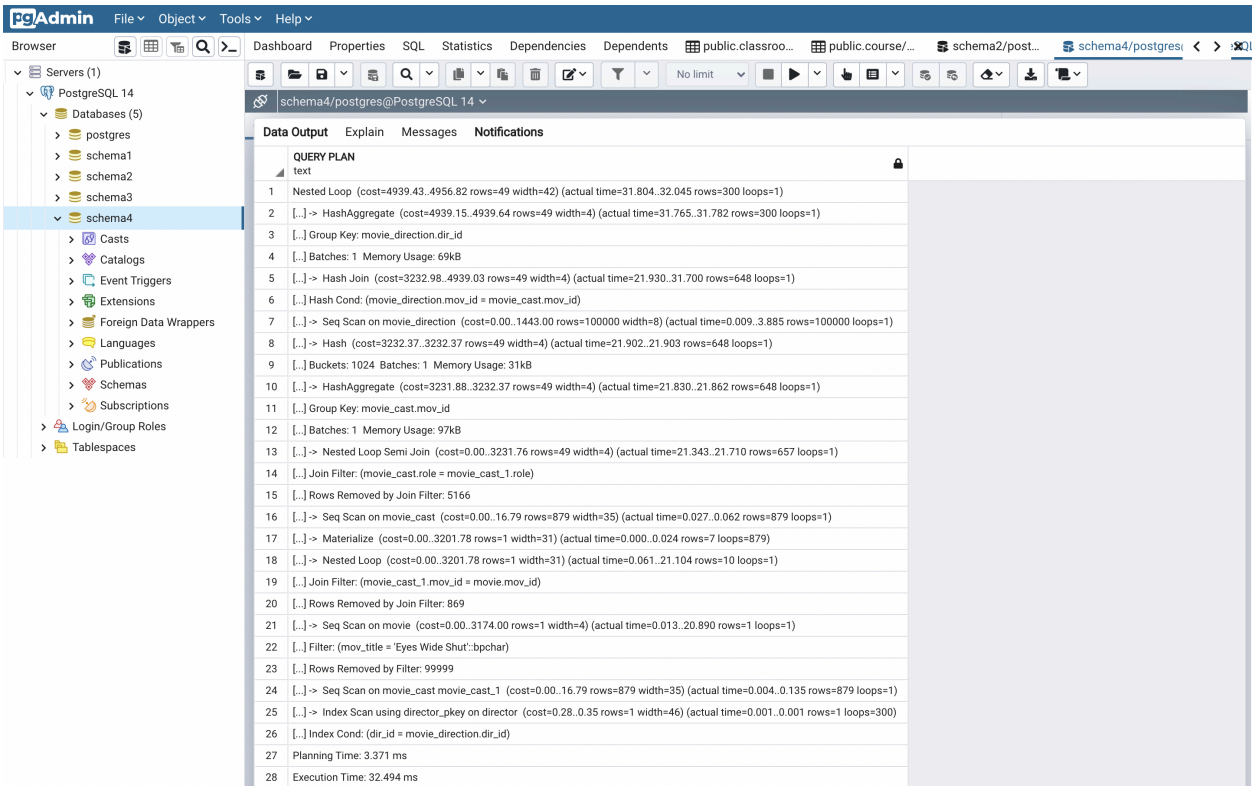
Here the hash index increased performance and less cost as it search in O(1) time so it helped in searching from mov_title=”Annie Hall” and B+ tree optimized the searched for the ids of movie and actor in O(logn) so overall much better performance

Execution plan and costs:

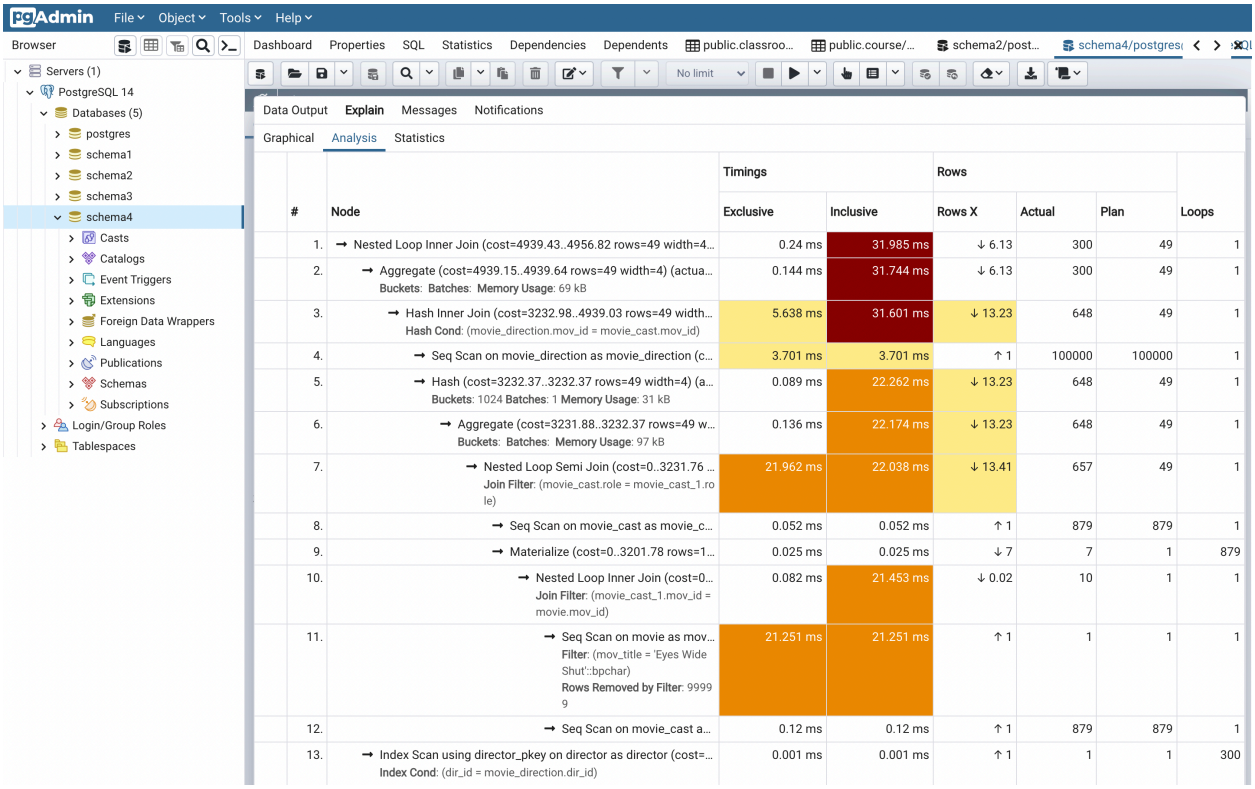


Query 11 with no indices

Planning and execution time :



Execution plan and costs:

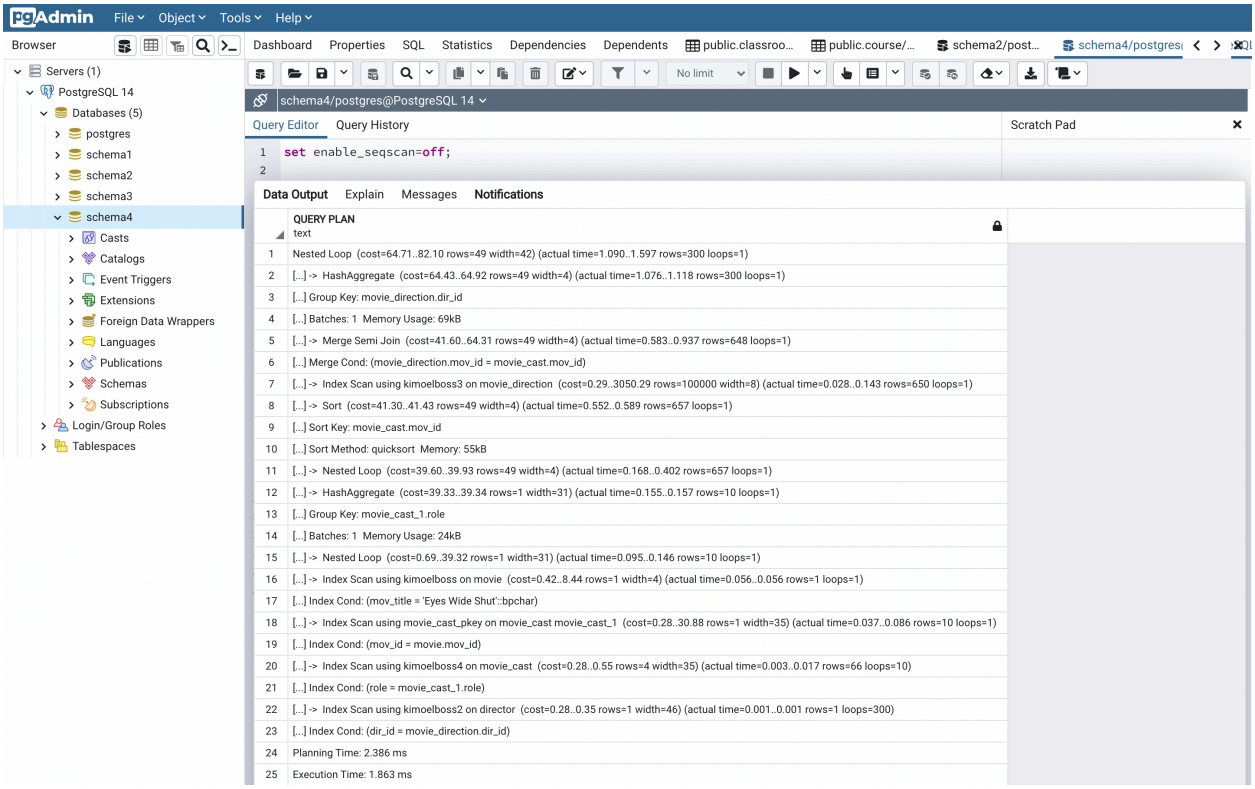


#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Nested Loop Inner Join (cost=4939.43..4956.82 rows=49 width=4...	0.24 ms	31.985 ms	↓ 6.13	300	49	1
2.	→ Aggregate (cost=4939.15..4939.64 rows=49 width=4) (actua... Buckets: Batches: Memory Usage: 69 kB	0.144 ms	31.744 ms	↓ 6.13	300	49	1
3.	→ Hash Inner Join (cost=3232.98..4939.03 rows=49 width... Hash Cond: (movie_direction.mov_id = movie_cast.mov_id)	5.638 ms	31.601 ms	↓ 13.23	648	49	1
4.	→ Seq Scan on movie_direction as movie_direction (c...	3.701 ms	3.701 ms	↑ 1	100000	100000	1
5.	→ Hash (cost=3232.37..3232.37 rows=49 width=4) (a... Buckets: 1024 Batches: 1 Memory Usage: 31 kB	0.089 ms	22.262 ms	↓ 13.23	648	49	1
6.	→ Aggregate (cost=3231.88..3232.37 rows=49 w... Buckets: Batches: Memory Usage: 97 kB	0.136 ms	22.174 ms	↓ 13.23	648	49	1
7.	→ Nested Loop Semi Join (cost=0..3231.76 ... Join Filter: (movie_cast.role = movie_cast_1.role)	21.962 ms	22.038 ms	↓ 13.41	657	49	1
8.	→ Seq Scan on movie_cast as movie_c...	0.052 ms	0.052 ms	↑ 1	879	879	1
9.	→ Materialize (cost=0..3201.78 rows=1...	0.025 ms	0.025 ms	↓ 7	7	1	879
10.	→ Nested Loop Inner Join (cost=0... Join Filter: (movie_cast_1.mov_id = movie.mov_id)	0.082 ms	21.453 ms	↓ 0.02	10	1	1
11.	→ Seq Scan on movie as mov... Filter: (mov_title = 'Eyes Wide Shut'::bpchar) Rows Removed by Filter: 99999	21.251 ms	21.251 ms	↑ 1	1	1	1
12.	→ Seq Scan on movie_cast a...	0.12 ms	0.12 ms	↑ 1	879	879	1
13.	→ Index Scan using director_pkey on director as director (cost=... Index Cond: (dir_id = movie_direction.dir_id)	0.001 ms	0.001 ms	↑ 1	1	1	300

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

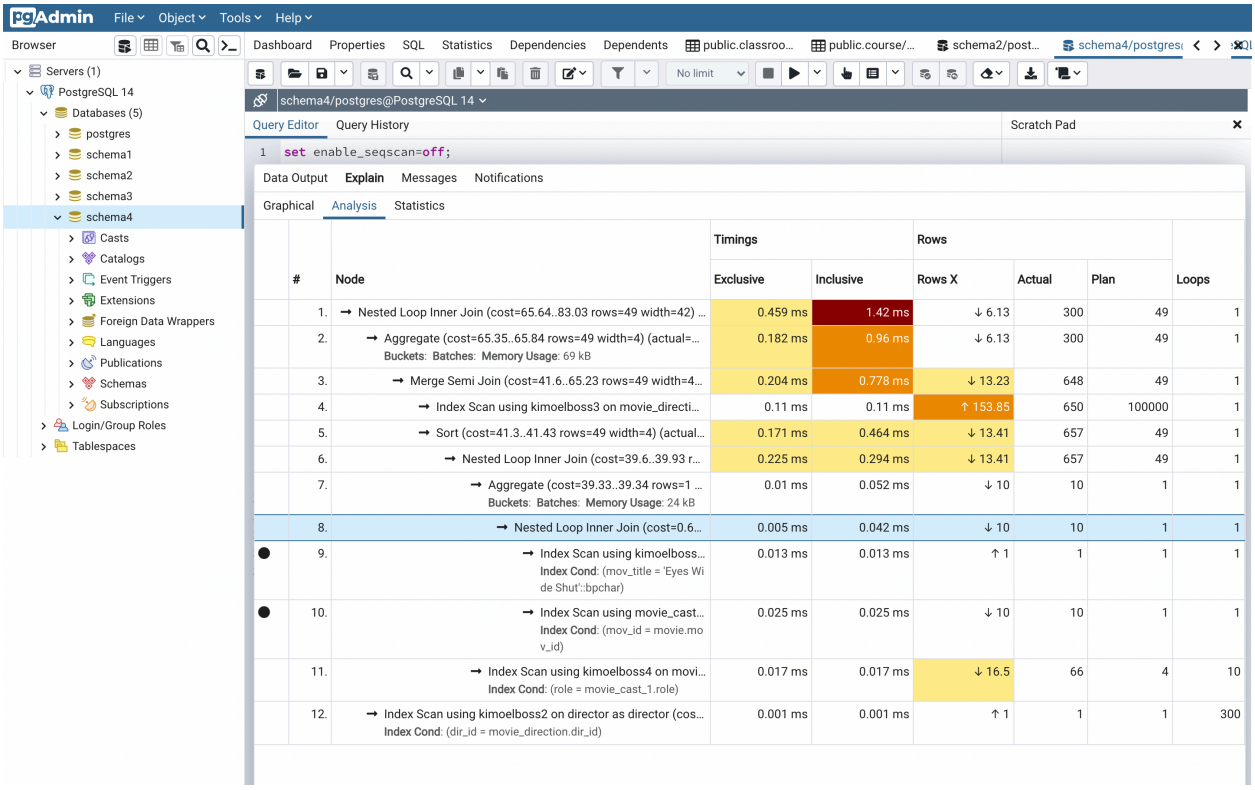
Disabled flags: seqscan

Planning and execution time :



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

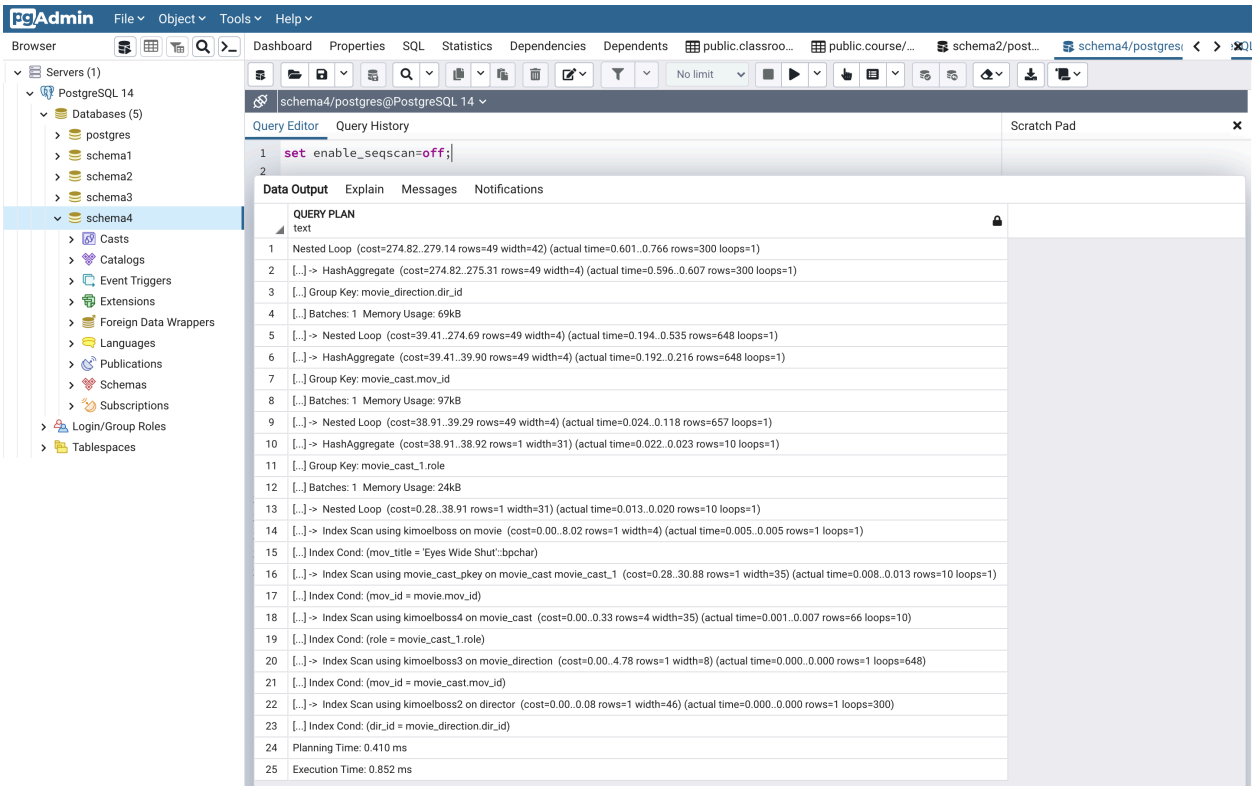
Execution plan and costs:



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

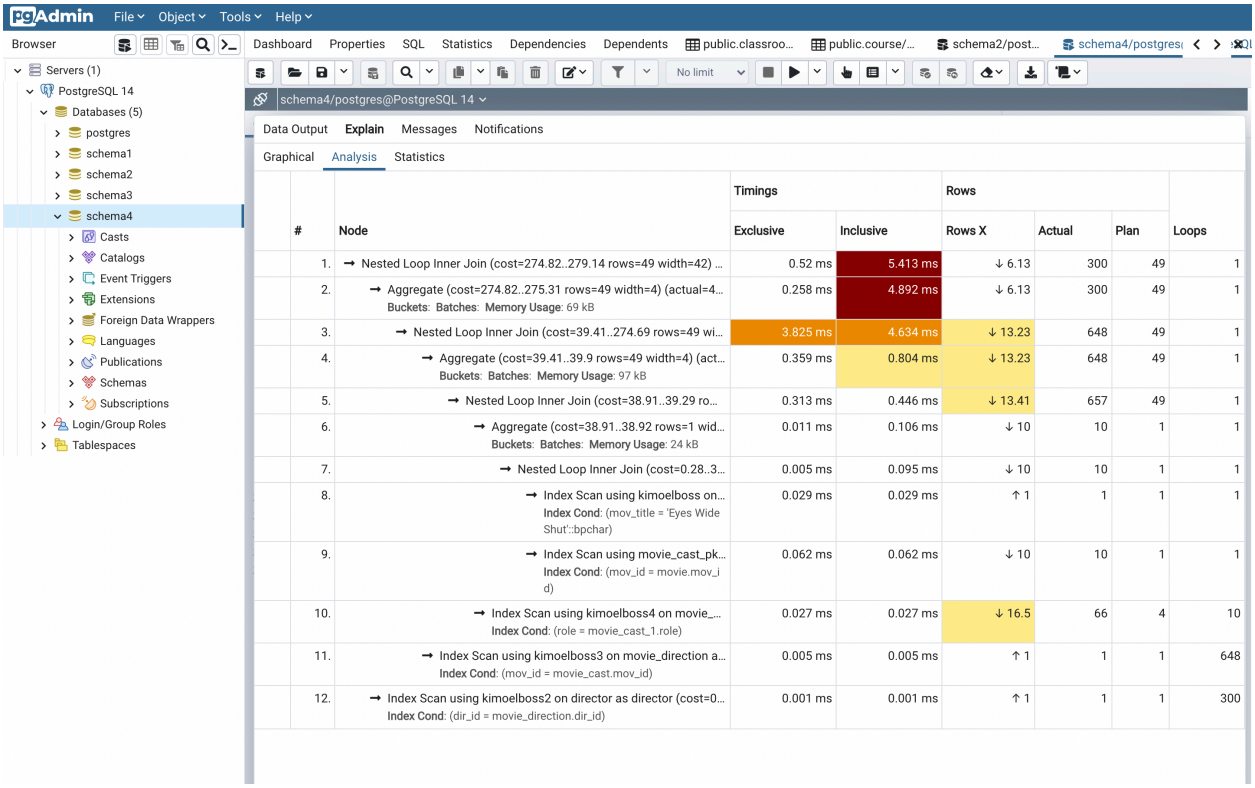
Disabled flags: seqscan

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:



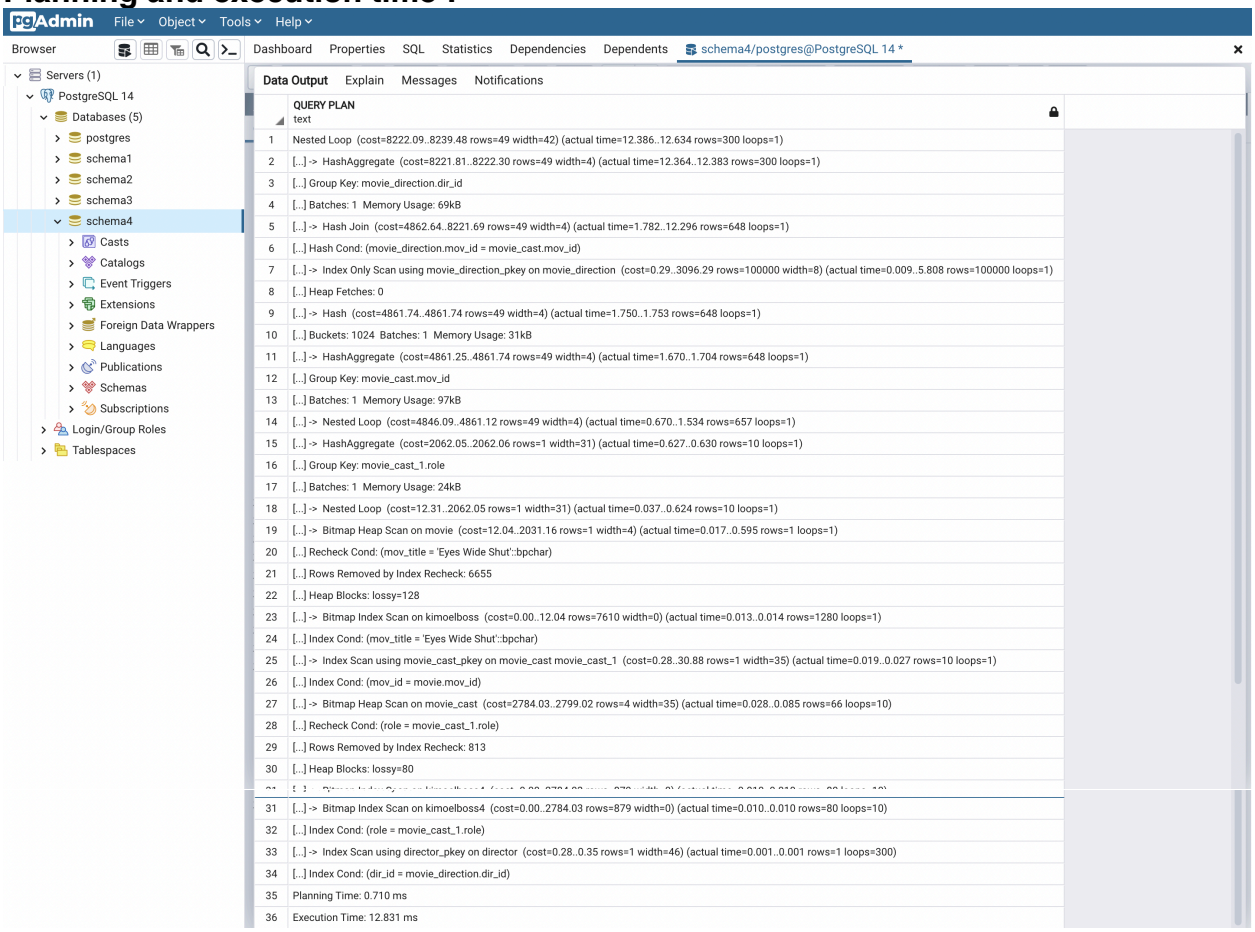
#	Node	Timings: Exclusive	Timings: Inclusive	Rows: Rows X	Rows: Actual	Rows: Plan	Loops
1.	→ Nested Loop Inner Join (cost=274.82..279.14 rows=49 width=42) ...	0.52 ms	5.413 ms	↓ 6.13	300	49	1
2.	→ Aggregate (cost=274.82..275.31 rows=49 width=4) (actual=4... Buckets: Batches: Memory Usage: 69 kB	0.258 ms	4.892 ms	↓ 6.13	300	49	1
3.	→ Nested Loop Inner Join (cost=39.41..274.69 rows=49 wi...	3.825 ms	4.634 ms	↓ 13.23	648	49	1
4.	→ Aggregate (cost=39.41..39.9 rows=49 width=4) (act... Buckets: Batches: Memory Usage: 97 kB	0.359 ms	0.804 ms	↓ 13.23	648	49	1
5.	→ Nested Loop Inner Join (cost=38.91..39.29 ro...	0.313 ms	0.446 ms	↓ 13.41	657	49	1
6.	→ Aggregate (cost=38.91..38.92 rows=1 wid... Buckets: Batches: Memory Usage: 24 kB	0.011 ms	0.106 ms	↓ 10	10	1	1
7.	→ Nested Loop Inner Join (cost=0.28..3...	0.005 ms	0.095 ms	↓ 10	10	1	1
8.	→ Index Scan using kimoelboss on... Index Cond: (mov_title = 'Eyes Wide Shut'::bpchar)	0.029 ms	0.029 ms	↑ 1	1	1	1
9.	→ Index Scan using movie_cast_pk... Index Cond: (mov_id = movie.mov_id)	0.062 ms	0.062 ms	↓ 10	10	1	1
10.	→ Index Scan using kimoelboss4 on movie_... Index Cond: (role = movie_cast_1.role)	0.027 ms	0.027 ms	↓ 16.5	66	4	10
11.	→ Index Scan using kimoelboss3 on movie_direction a... Index Cond: (mov_id = movie_cast.mov_id)	0.005 ms	0.005 ms	↑ 1	1	1	648
12.	→ Index Scan using kimoelboss2 on director as director (cost=0... Index Cond: (dir_id = movie_direction.dir_id)	0.001 ms	0.001 ms	↑ 1	1	1	300

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

Disabled flags: seqscan

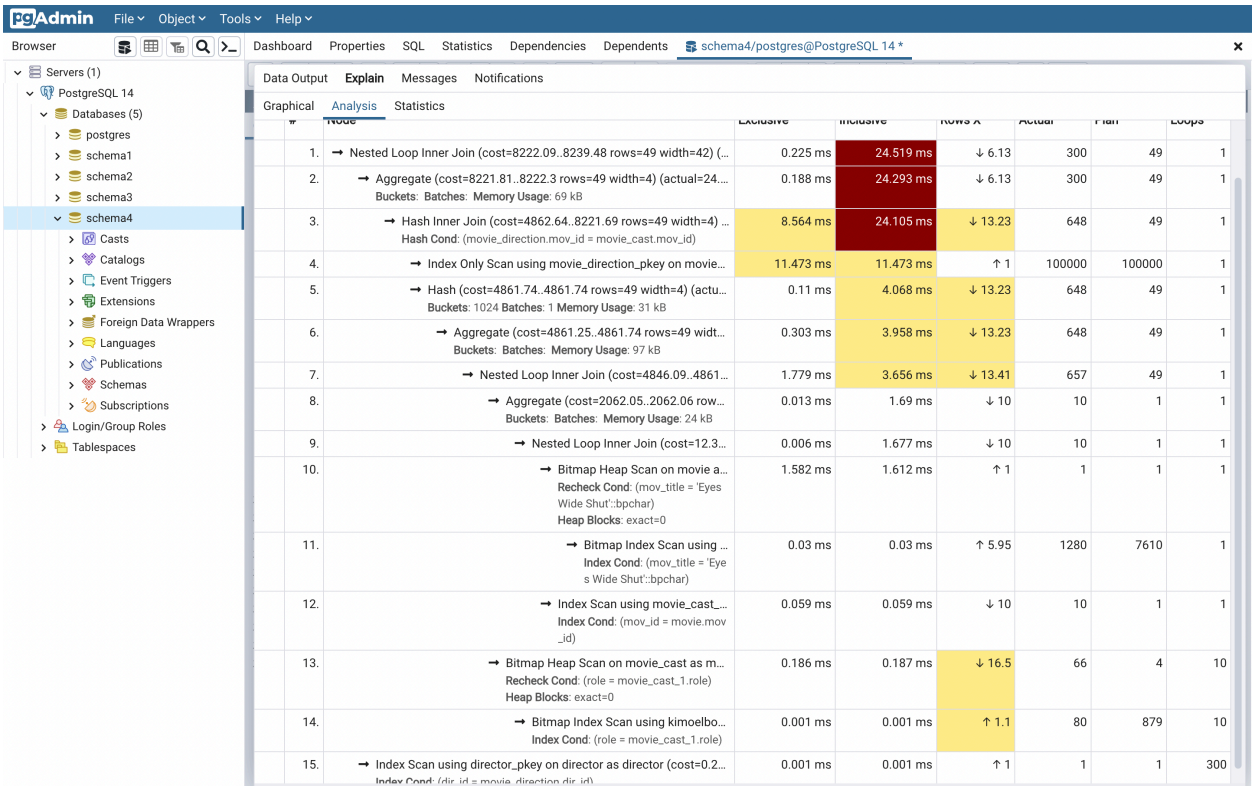
PRIMARY KEY ON

Planning and execution time :



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

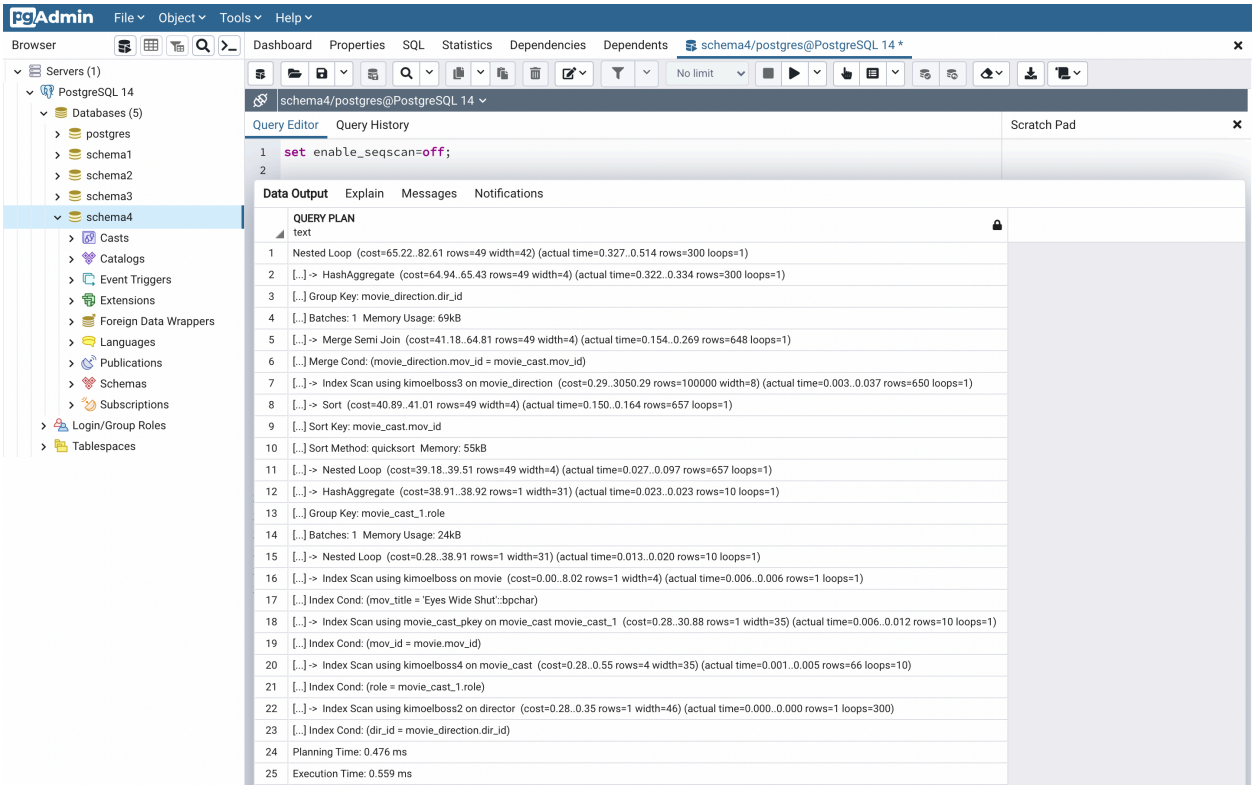
Execution plan and costs:



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

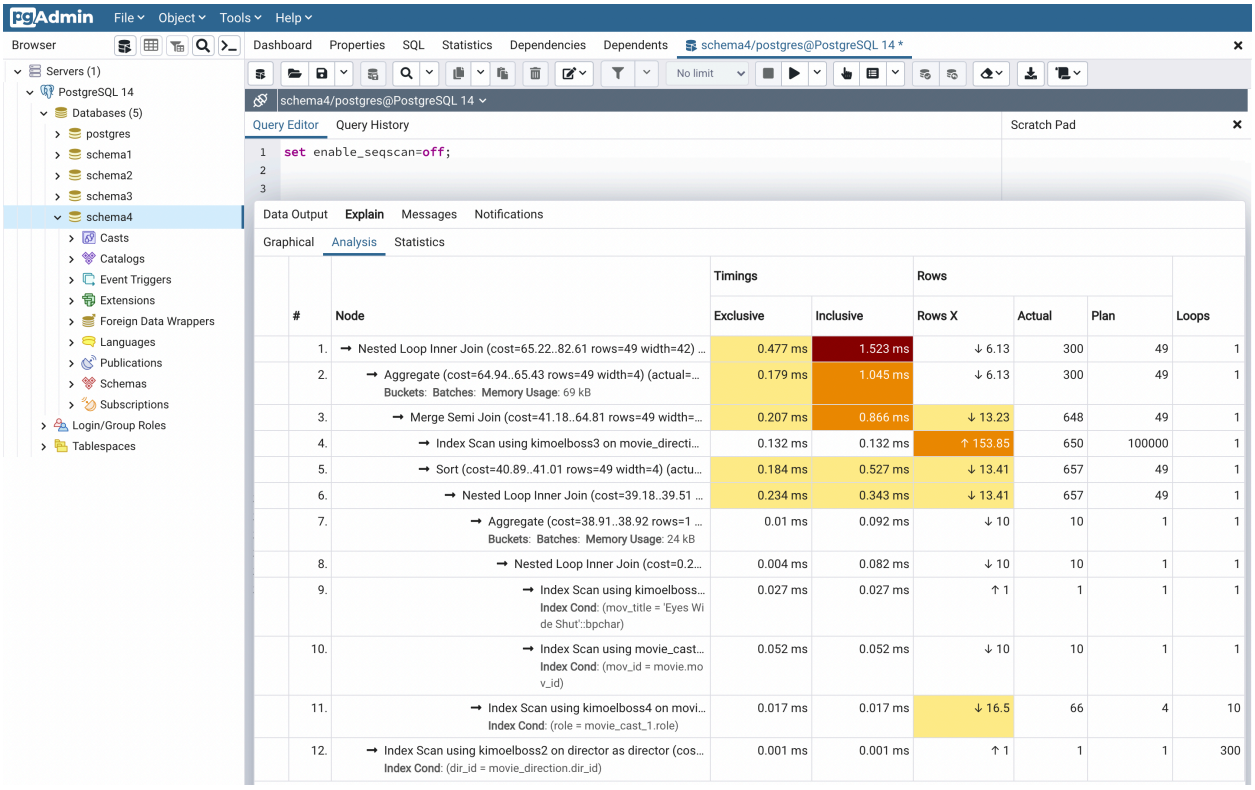
Disabled flags: seqscan

Planning and execution time :



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

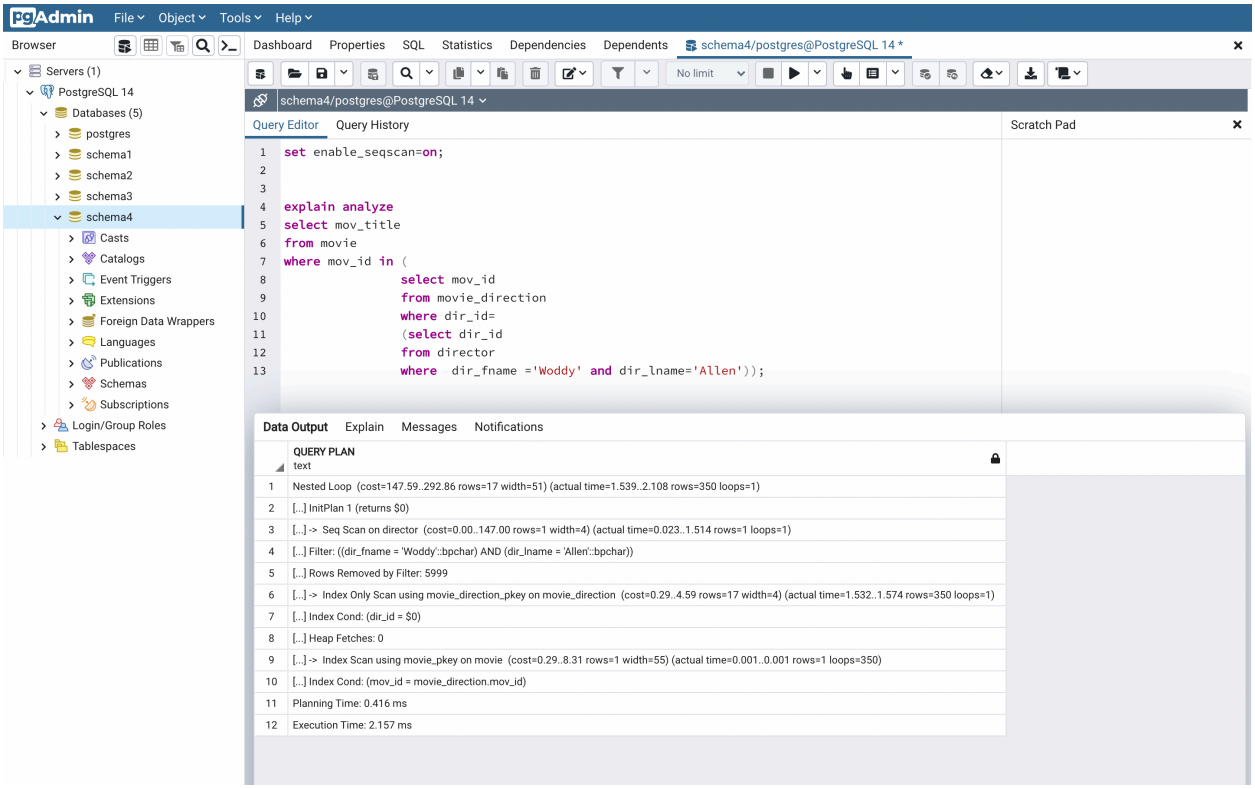
Execution plan and costs:



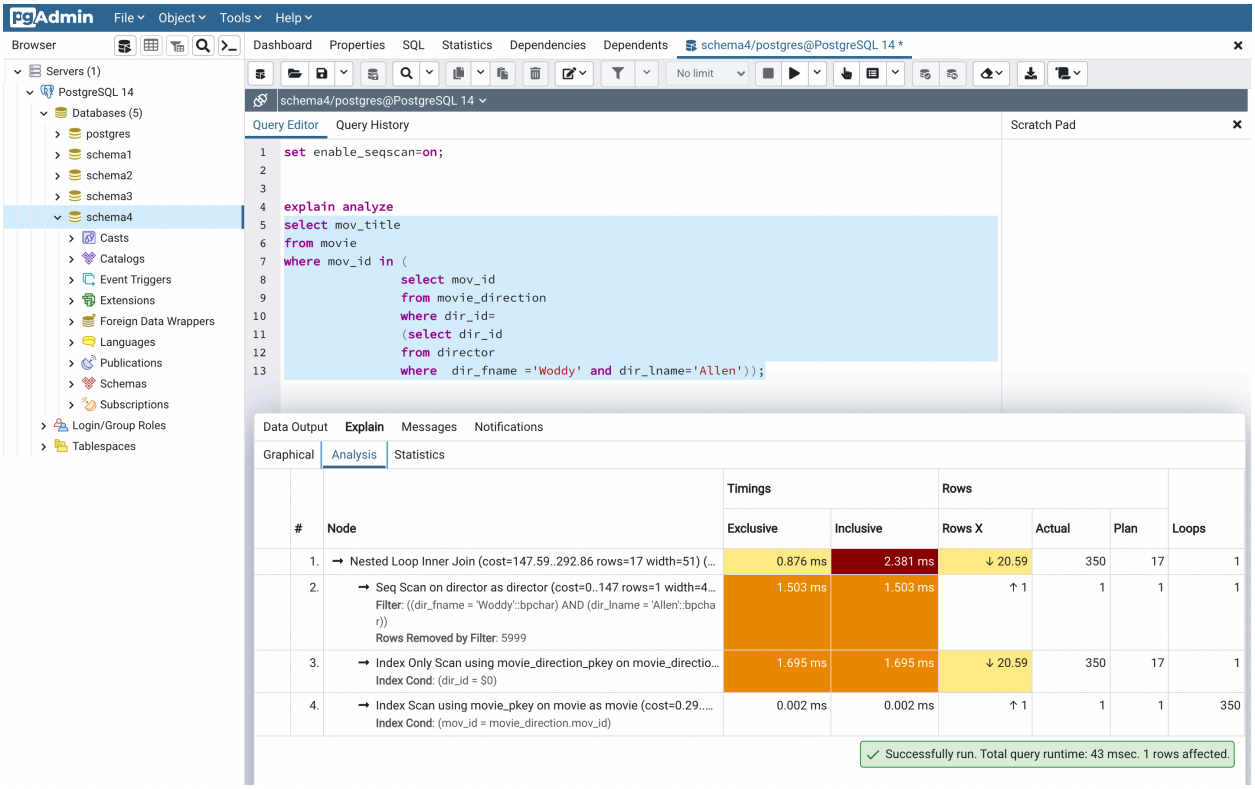
	#	Node	Timings		Rows			Loops
			Exclusive	Inclusive	Rows X	Actual	Plan	
	1.	→ Nested Loop Inner Join (cost=65.22..82.61 rows=49 width=42) ...	0.477 ms	1.523 ms	↓ 6.13	300	49	1
	2.	→ Aggregate (cost=64.94..65.43 rows=49 width=4) (actual=... Buckets: Batches: Memory Usage: 69 kB	0.179 ms	1.045 ms	↓ 6.13	300	49	1
	3.	→ Merge Semi Join (cost=41.18..64.81 rows=49 width=...	0.207 ms	0.866 ms	↓ 13.23	648	49	1
	4.	→ Index Scan using kimoelboss3 on movie_directi...	0.132 ms	0.132 ms	↑ 153.85	650	100000	1
	5.	→ Sort (cost=40.89..41.01 rows=49 width=4) (actu...	0.184 ms	0.527 ms	↓ 13.41	657	49	1
	6.	→ Nested Loop Inner Join (cost=39.18..39.51 ...	0.234 ms	0.343 ms	↓ 13.41	657	49	1
	7.	→ Aggregate (cost=38.91..38.92 rows=1 ... Buckets: Batches: Memory Usage: 24 kB	0.01 ms	0.092 ms	↓ 10	10	1	1
	8.	→ Nested Loop Inner Join (cost=0.2...	0.004 ms	0.082 ms	↓ 10	10	1	1
	9.	→ Index Scan using kimoelboss... Index Cond: (mov_title = 'Eyes Wide Shut'::bpchar)	0.027 ms	0.027 ms	↑ 1	1	1	1
	10.	→ Index Scan using movie_cast... Index Cond: (mov_id = movie.mov_id)	0.052 ms	0.052 ms	↓ 10	10	1	1
	11.	→ Index Scan using kimoelboss4 on movi... Index Cond: (role = movie_cast_1.role)	0.017 ms	0.017 ms	↓ 16.5	66	4	10
	12.	→ Index Scan using kimoelboss2 on director as director (cos... Index Cond: (dir_id = movie_direction.dir_id)	0.001 ms	0.001 ms	↑ 1	1	1	300

Query 12 no index

Planning and execution time :



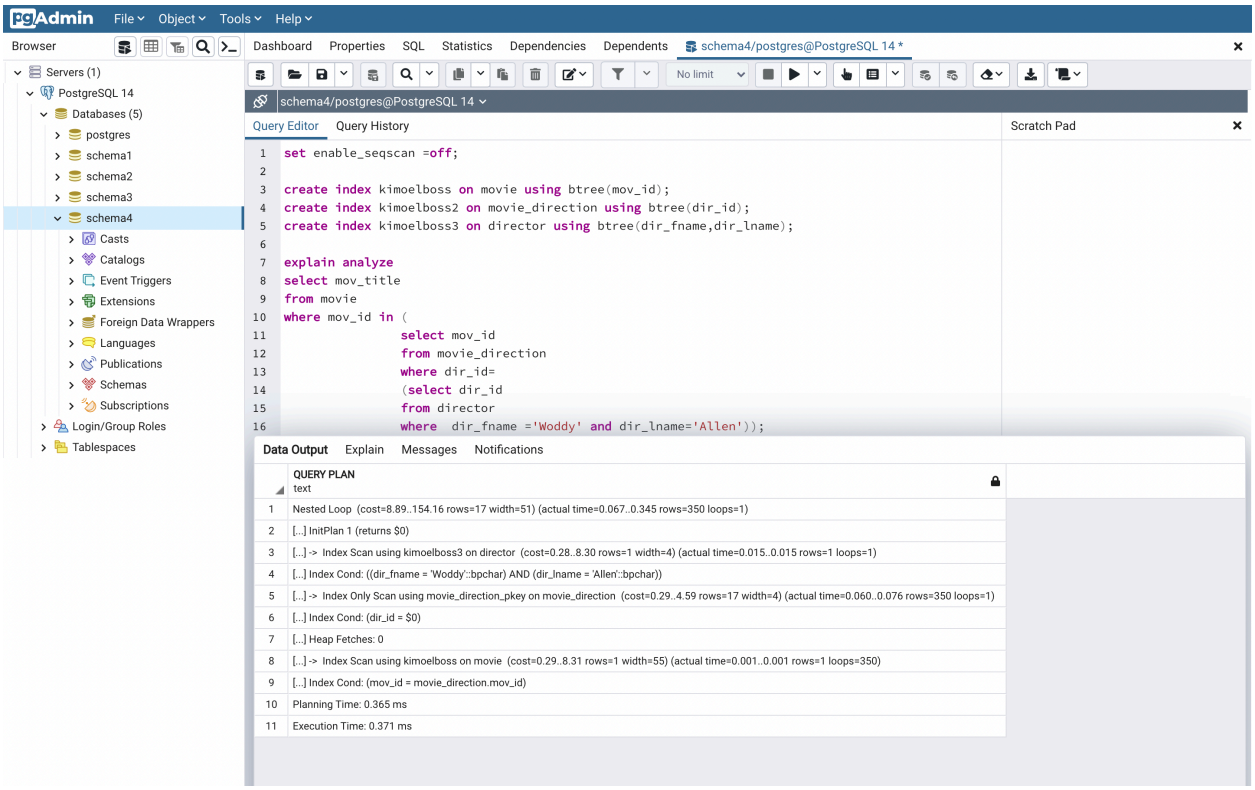
Execution plan and costs:



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

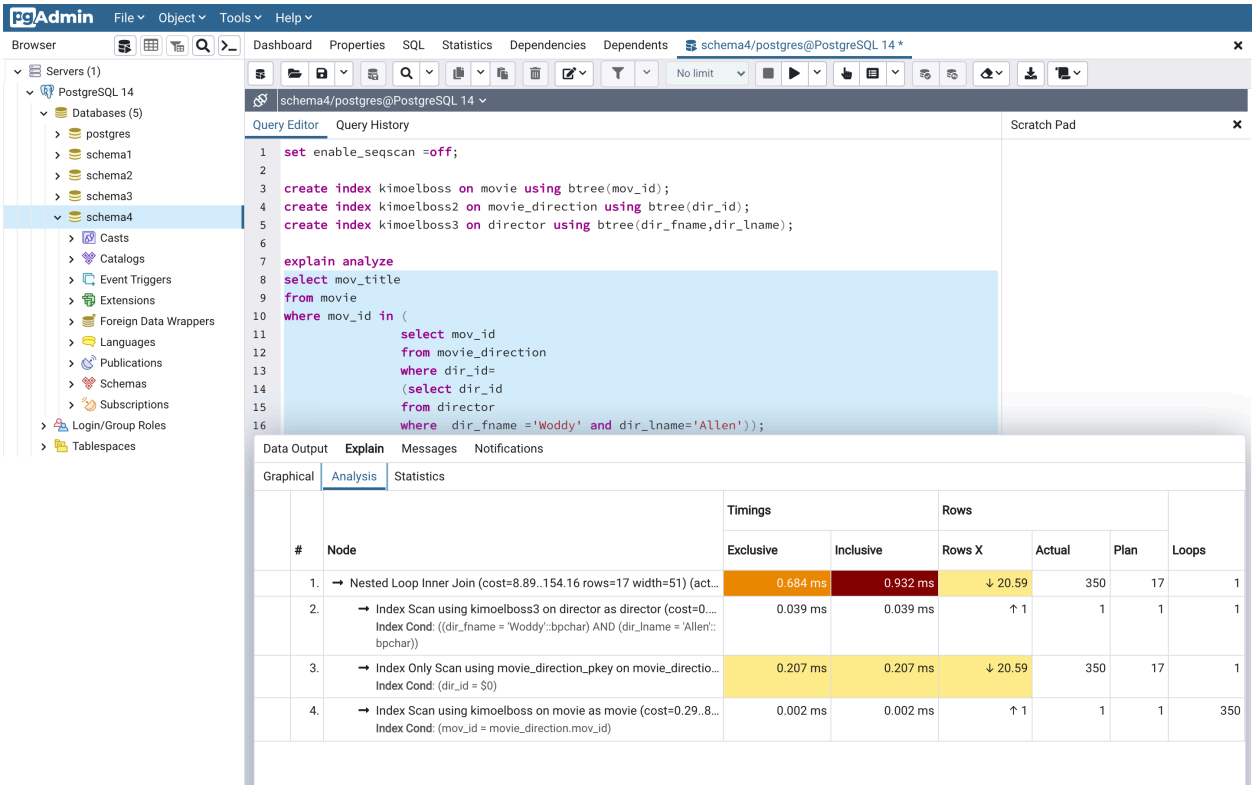
Disabled flags: seqscan

Planning and execution time :



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

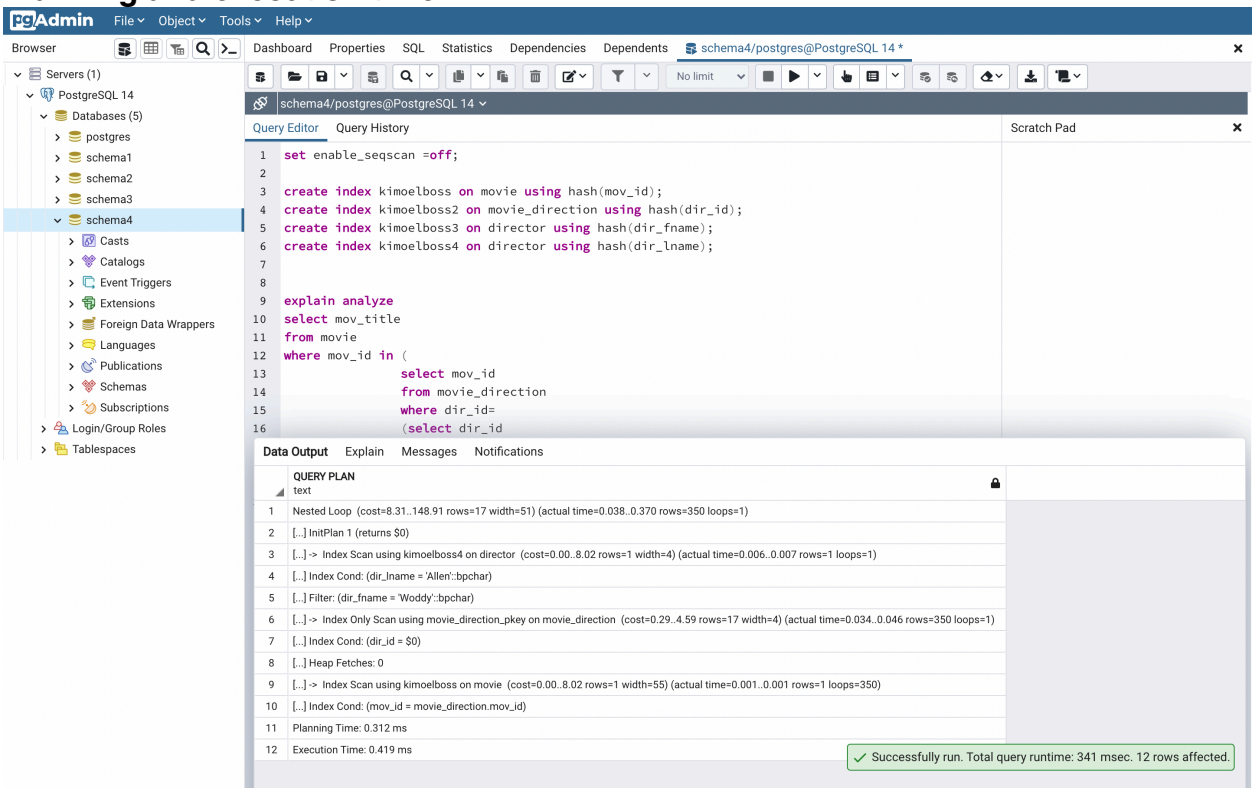
Execution plan and costs:



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

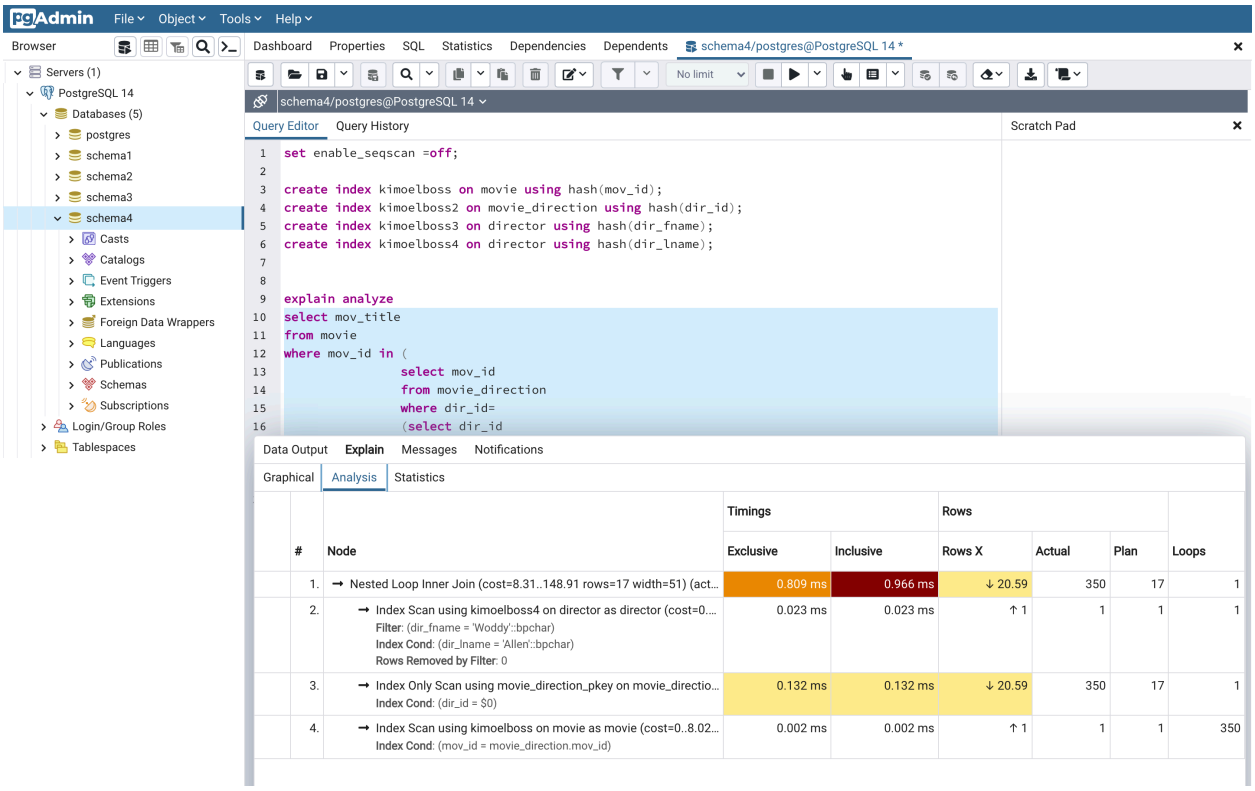
Disabled flags: seqscan

Planning and execution time :



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)

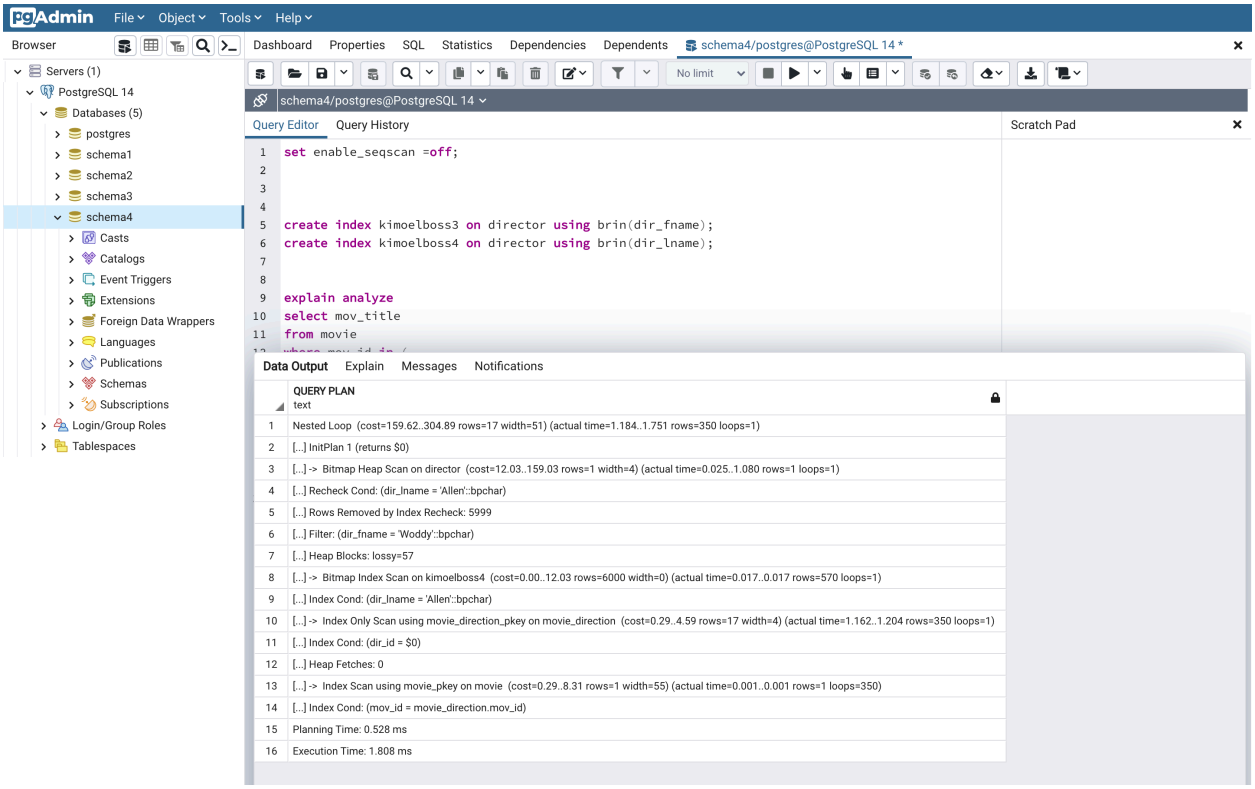
Execution plan and costs:



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

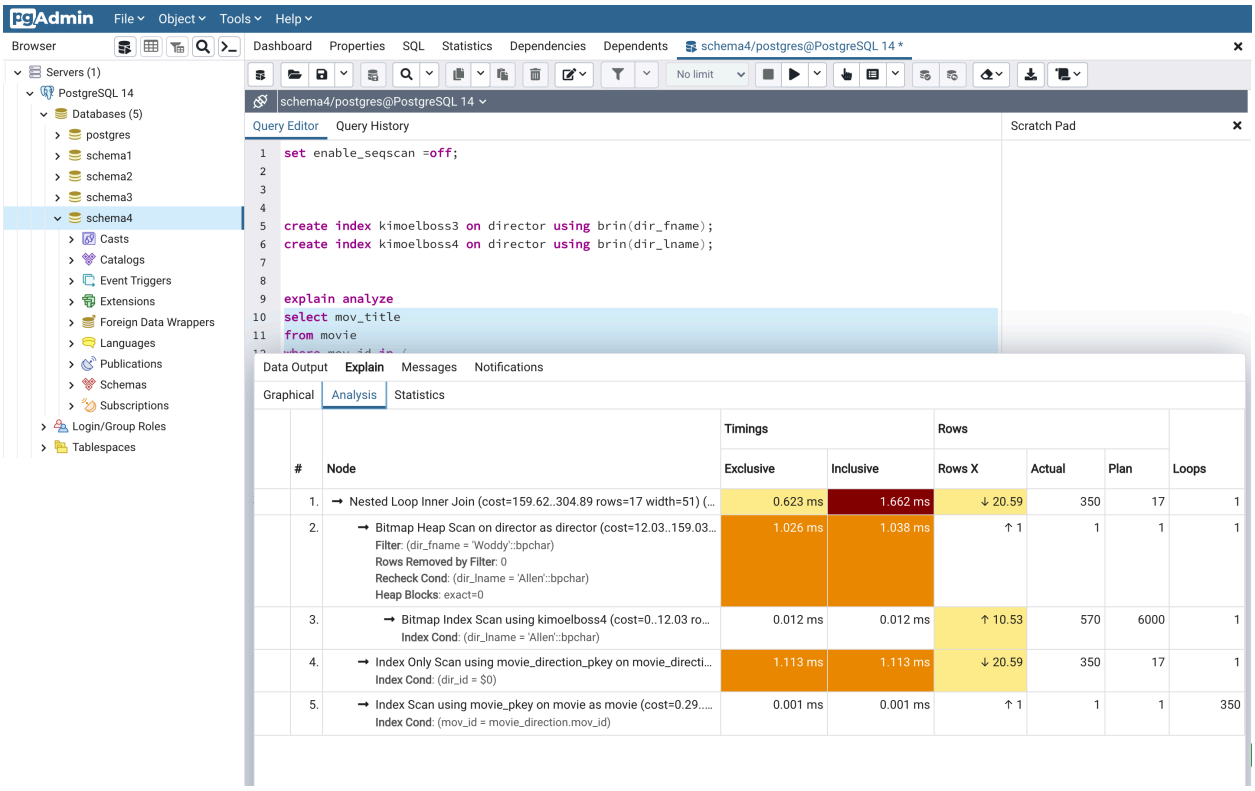
Disabled flags: seqscan

Planning and execution time :



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

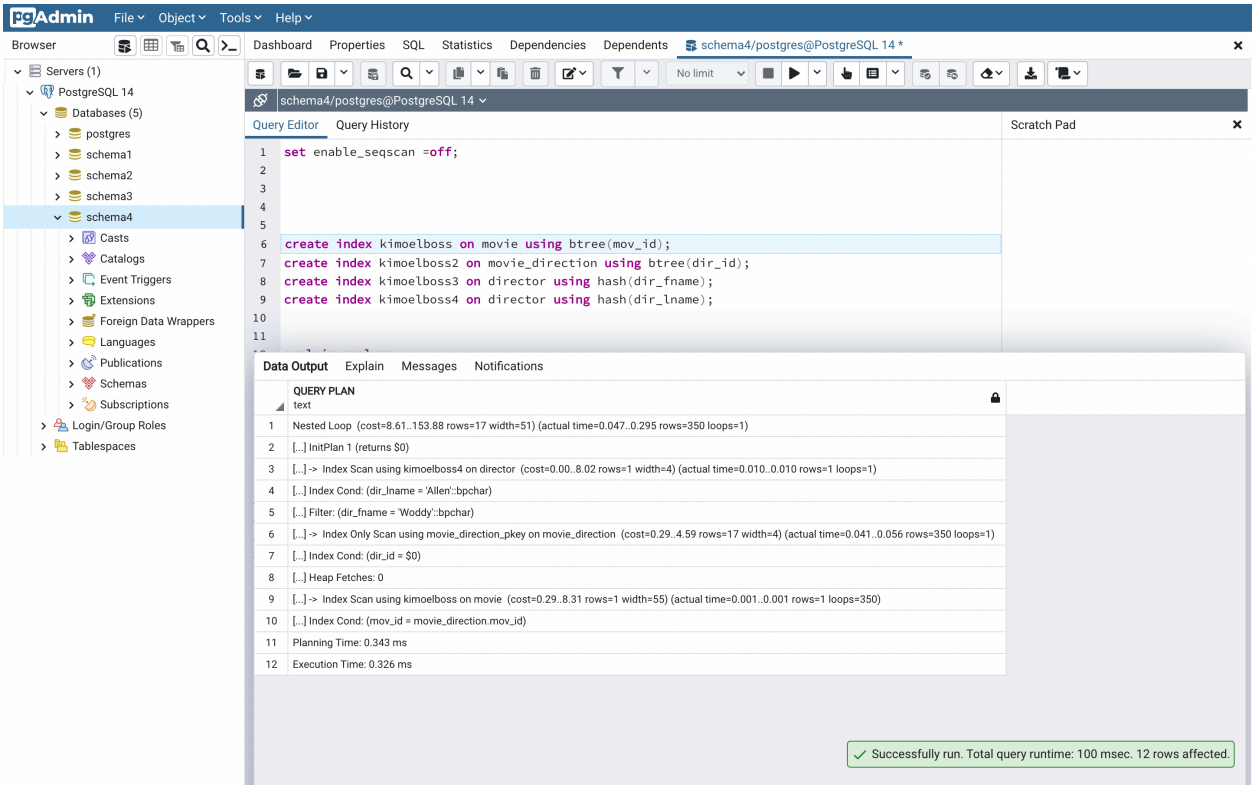
Execution plan and costs:



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 :



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

Execution plan and costs: