Universidade de São Paulo
Escola de Artes, Ciências e Humanidades
Disciplina: Laboratório de Banco de Dados
Profª Dra. Fátima Nunes.

Administração de Condomínio

Parte III - Artefato C

Fernando K. G. de Amorim – 10387644
João Guilherme da Costa Seike – 9784634
Lucas Pereira Castelo Branco – 10258772
Victor Gomes de O. M. Nicola – 9844881

São Paulo
2019

Artefato C

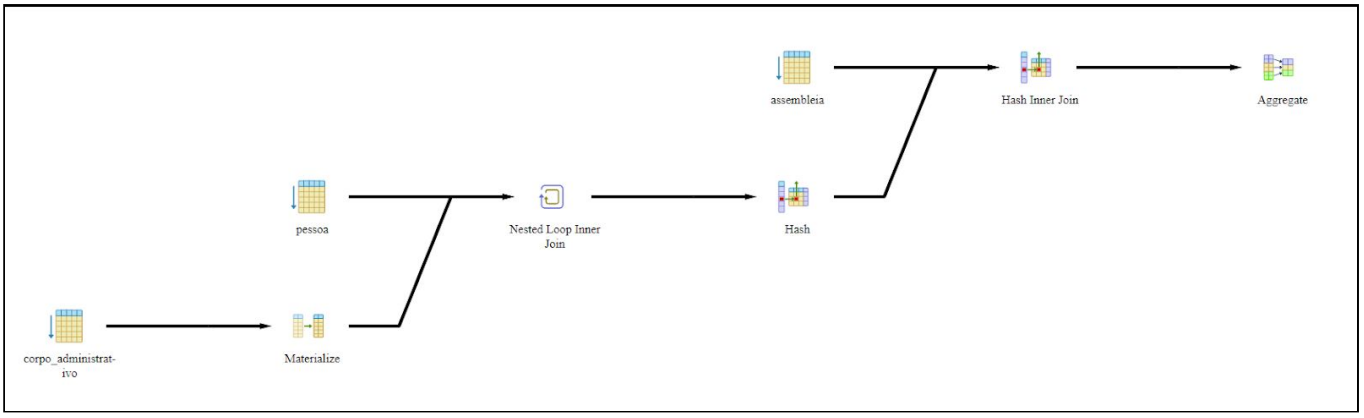
Todas as representações gráficas abaixo foram retiradas da ferramenta EXPLAIN disponível no software PGAdmin.

DISCLAIMER: Os tempos de planejamento e execução aqui demonstrados estão sob a política de *caching* do PostgreSQL, o que provoca uma mudança relativa nos tempos de execução se comparada a outras etapas demonstradas aqui em outros artefatos. Mais detalhes sobre a política do PostgreSQL podem ser encontraddas aqui: https://www.slideshare.net/uptimeforce/postgresql-query-cache-pgc

Query I

Original

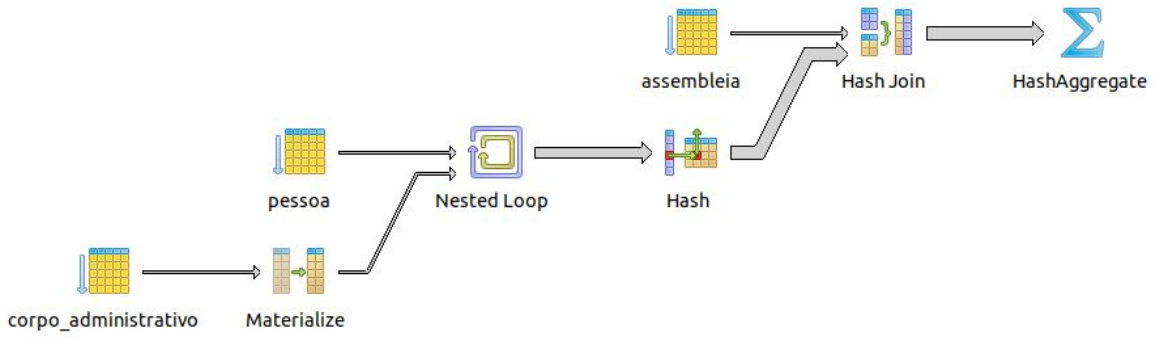
```
SELECT Pessoa.nome,
                MAX(assembleia.data) AS ultima_assembleia
        FROM adm_condominio.Pessoa AS Pessoa
        JOIN adm_condominio.Corpo_Administrativo AS Corpo_Administrativo
          ON Pessoa.id_pessoa IN (Corpo_Administrativo.id_sindico, Corpo_Administrativo.id_subsindico,
                                  Corpo_Administrativo.id_conselheiro_1,
Corpo_Administrativo.id_conselheiro_2,
                                  Corpo_Administrativo.id_conselheiro_3)
        JOIN adm_condominio.Assembleia AS Assembleia
          ON Corpo_Administrativo.id_corpo = Assembleia.fk_id_corpo_admin
        WHERE data_eleicao <= CAST(NOW() AS DATE) - interval '2 years'
        GROUP BY 1;
```



	QUERY PLAN text
1	HashAggregate (cost=172.41..173.41 rows=100 width=17) (actual time=1.645..1.652 rows=39 loops=1)
2	Group Key: pessoa.nome
3	-> Hash Join (cost=148.56..168.30 rows=822 width=17) (actual time=1.568..1.599 rows=125 loops=1)
4	Hash Cond: (assembleia.fk_id_corpo_admin = corpo_administrativo.id_corpo)
5	-> Seq Scan on assembleia (cost=0.00..13.00 rows=300 width=8) (actual time=0.024..0.027 rows=40 loops=1)
6	-> Hash (cost=145.14..145.14 rows=274 width=17) (actual time=1.525..1.525 rows=280 loops=1)
7	Buckets: 1024 Batches: 1 Memory Usage: 22kB
8	-> Nested Loop (cost=0.00..145.14 rows=274 width=17) (actual time=0.055..1.471 rows=280 loops=1)
9	Join Filter: ((pessoa.id_pessoa = corpo_administrativo.id_sindico) OR (pessoa.id_pessoa = corpo_administrativo.id_subsindico) OR (pess...
10	Rows Removed by Join Filter: 5320
11	-> Seq Scan on pessoa (cost=0.00..2.00 rows=100 width=17) (actual time=0.014..0.022 rows=100 loops=1)
12	-> Materialize (cost=0.00..3.28 rows=56 width=24) (actual time=0.000..0.003 rows=56 loops=100)
13	-> Seq Scan on corpo_administrativo (cost=0.00..3.00 rows=56 width=24) (actual time=0.017..0.061 rows=56 loops=1)
14	Filter: (data_eleicao <= ((now())::date - '2 years'::interval))
15	Rows Removed by Filter: 44
16	Planning Time: 0.340 ms
17	Execution Time: 1.725 ms

a)

```
SELECT Pessoa.nome,
        MAX(assembleia.data) AS ultima_assembleia
        FROM adm_condominio.Pessoa AS Pessoa
        JOIN adm_condominio.Corpo_Administrativo AS Corpo_Administrativo
        ON Pessoa.id_pessoa IN (Corpo_Administrativo.id_sindico, Corpo_Administrativo.id_subsindico,
                Corpo_Administrativo.id_conselheiro_1, Corpo_Administrativo.id_conselheiro_2,
                Corpo_Administrativo.id_conselheiro_3)
        JOIN adm_condominio.Assembleia AS Assembleia
        ON Corpo_Administrativo.id_corpo = Assembleia.fk_id_corpo_admin
        GROUP BY 1;
```



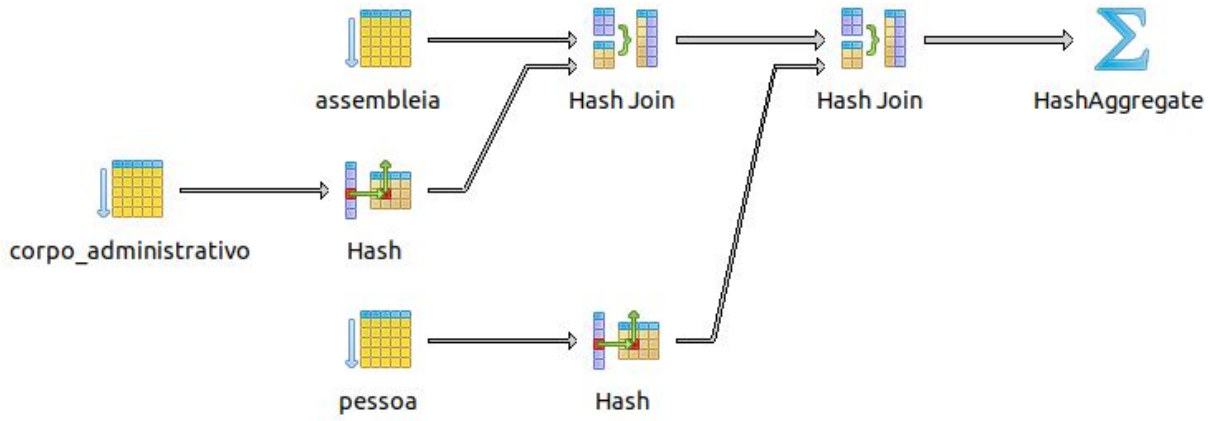
	QUERY PLAN text
1	HashAggregate (cost=290.70..291.70 rows=100 width=17) (actual time=2.944..2.955 rows=45 loops=1)
2	Group Key: pessoa.nome
3	-> Hash Join (cost=260.38..283.35 rows=1470 width=17) (actual time=2.801..2.859 rows=200 loops=1)
4	Hash Cond: (assembleia.fk_id_corpo_admin = corpo_administrativo.id_corpo)
5	-> Seq Scan on assembleia (cost=0.00..13.00 rows=300 width=8) (actual time=0.015..0.019 rows=40 loops=1)
6	-> Hash (cost=254.25..254.25 rows=490 width=17) (actual time=2.779..2.779 rows=500 loops=1)
7	Buckets: 1024 Batches: 1 Memory Usage: 33kB
8	-> Nested Loop (cost=0.00..254.25 rows=490 width=17) (actual time=0.026..2.668 rows=500 loops=1)
9	Join Filter: ((pessoa.id_pessoa = corpo_administrativo.id_sindico) OR (pessoa.id_pessoa = corpo_administrativo.id_subsindico) OR (pessoa.id_pessoa = corpo_administrat...
10	Rows Removed by Join Filter: 9500
11	-> Seq Scan on pessoa (cost=0.00..2.00 rows=100 width=17) (actual time=0.007..0.024 rows=100 loops=1)
12	-> Materialize (cost=0.00..2.50 rows=100 width=24) (actual time=0.000..0.008 rows=100 loops=100)
13	-> Seq Scan on corpo_administrativo (cost=0.00..2.00 rows=100 width=24) (actual time=0.007..0.022 rows=100 loops=1)
14	Planning time: 0.234 ms
15	Execution time: 3.019 ms

Mudanças realizadas: removemos a cláusula WHERE.

modificações que ocorreram no plano de execução: A remoção do WHERE causou uma baixa do Planning time, porém houve um aumento significativo no tempo de execução. O motivo disso é que quando retiramos o WHERE, o escopo executado é maior, e portanto, o tempo aumenta.

b)

```
SELECT Pessoa.nome,
        MAX(assembleia.data) AS ultima_assembleia
        FROM adm_condominio.Pessoa AS Pessoa
        JOIN adm_condominio.Corpo_Administrativo AS Corpo_Administrativo
        ON Pessoa.id_pessoa = Corpo_Administrativo.id_sindico
        JOIN adm_condominio.Assembleia AS Assembleia
        ON Corpo_Administrativo.id_corpo = Assembleia.fk_id_corpo_admin
        WHERE data_eleicao <= CAST(NOW() AS DATE) - interval '2 years'
        GROUP BY 1;
```



	QUERY PLAN text
1	HashAggregate (cost=22.06..23.06 rows=100 width=17) (actual time=0.209..0.212 rows=9 loops=1)
2	Group Key: pessoa.nome
3	-> Hash Join (cost=6.95..21.22 rows=168 width=17) (actual time=0.159..0.188 rows=25 loops=1)
4	Hash Cond: (corpo administrativo.id sindico = pessoa.id pessoa)
5	-> Hash Join (cost=3.70..17.51 rows=168 width=8) (actual time=0.094..0.114 rows=25 loops=1)
6	Hash Cond: (assembleia.fk id corpo admin = corpo administrativo.id corpo)
7	-> Seq Scan on assembleia (cost=0.00..13.00 rows=300 width=8) (actual time=0.007..0.012 rows=40 loops=1)
8	-> Hash (cost=3.00..3.00 rows=56 width=8) (actual time=0.077..0.077 rows=56 loops=1)
9	Buckets: 1024 Batches: 1 Memory Usage: 11kB
10	-> Seq Scan on corpo administrativo (cost=0.00..3.00 rows=56 width=8) (actual time=0.014..0.063 rows=56 loops=1)
11	Filter: (data eleicao <= ((now())::date - '2 years'::interval))
12	Rows Removed by Filter: 44
13	-> Hash (cost=2.00..2.00 rows=100 width=17) (actual time=0.057..0.057 rows=100 loops=1)
14	Buckets: 1024 Batches: 1 Memory Usage: 13kB
15	-> Seq Scan on pessoa (cost=0.00..2.00 rows=100 width=17) (actual time=0.012..0.030 rows=100 loops=1)
16	Planning time: 0.414 ms
17	Execution time: 0.275 ms

Mudanças realizadas: Alteramos o primeiro “JOIN”

modificações que ocorreram no plano de execução: A modificação que fizemos no JOIN foi uma simplificação dele, ao fazermos uma mera igualdade ao invés de uma comparação de listas. O resultado foi uma redução drástica no modelo. Isso prova que o SGBD não é otimizado para fazer múltiplas verificações.

Query II

Original

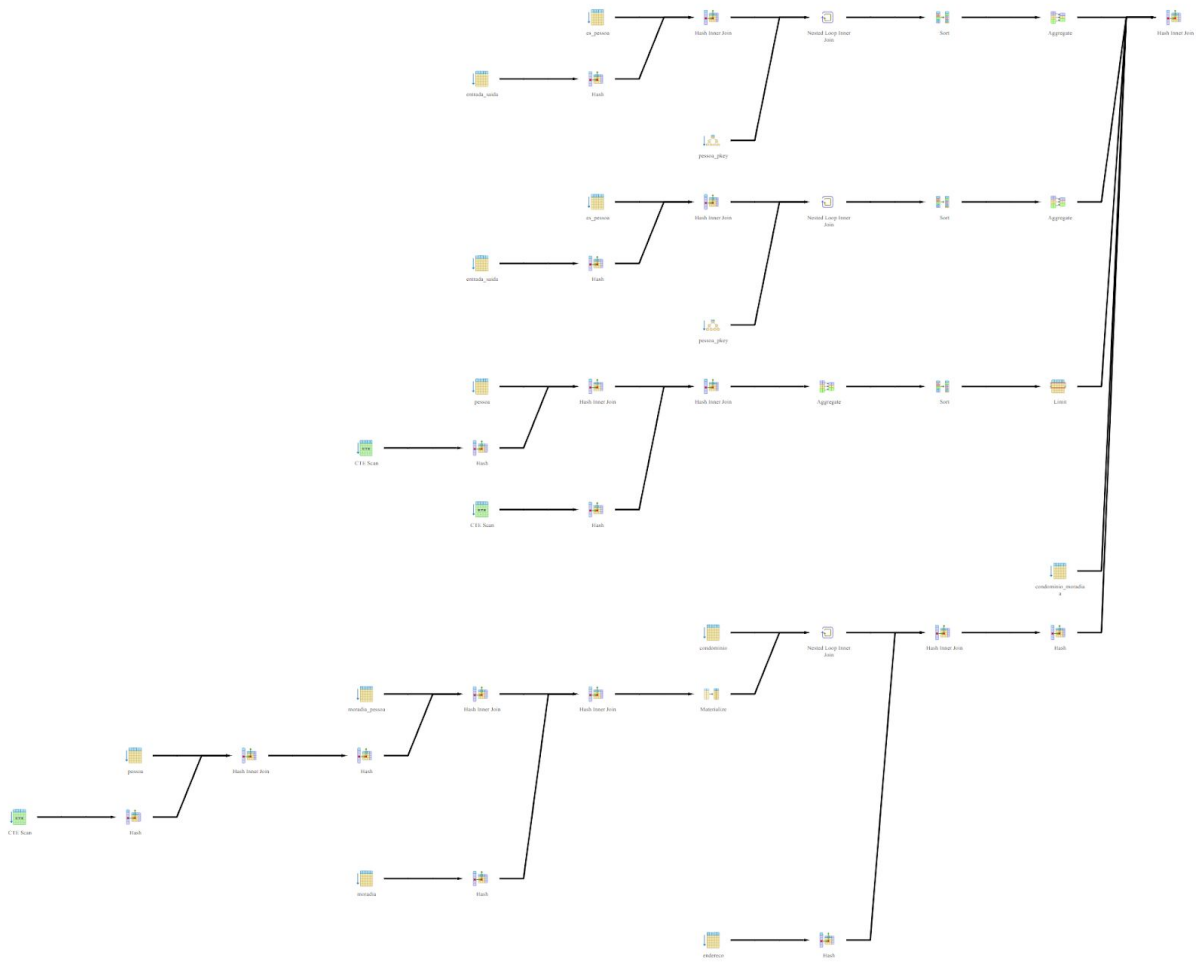
```
WITH sum_entradas AS (
```

```
		SELECT pessoa.id_pessoa AS id_pessoa, SUM(DATE_PART('epoch', Entrada_Saida.data_hora))
AS soma
		FROM adm_condominio.Pessoa AS Pessoa
		JOIN adm_condominio.Es_Pessoa AS Es_Pessoa ON Pessoa.id_pessoa =
Es_Pessoa.fk_id_pessoa
		JOIN adm_condominio.Entrada_Saida AS Entrada_Saida ON Entrada_Saida.id_es =
Es_Pessoa.fk_id_es
		WHERE Entrada_Saida.acao = 'e'
		GROUP BY 1
),

sum_saidas AS (
		SELECT pessoa.id_pessoa AS id_pessoa, SUM(DATE_PART('epoch', Entrada_Saida.data_hora
)) AS soma
		FROM adm_condominio.Pessoa AS Pessoa
		JOIN adm_condominio.Es_Pessoa AS Es_Pessoa ON Pessoa.id_pessoa =
Es_Pessoa.fk_id_pessoa
		JOIN adm_condominio.Entrada_Saida AS Entrada_Saida ON Entrada_Saida.id_es =
Es_Pessoa.fk_id_es
		WHERE Entrada_Saida.acao = 's'
		GROUP BY 1
),

top_five AS (
	SELECT Pessoa.id_pessoa, SUM(sum_entradas.soma - sum_saidas.soma)
		FROM adm_condominio.Pessoa AS Pessoa
		JOIN sum_entradas ON sum_entradas.id_pessoa = Pessoa.id_pessoa
		JOIN sum_saidas ON sum_saidas.id_pessoa = Pessoa.id_pessoa
		GROUP BY 1 ORDER BY 2 DESC
		LIMIT 5
)

SELECT Pessoa.nome, Endereco.cidade
	FROM adm_condominio.Pessoa AS Pessoa
	JOIN adm_condominio.Moradia_Pessoa AS Moradia_Pessoa
	  ON Moradia_Pessoa.fk_id_pessoa = Pessoa.id_pessoa
	JOIN adm_condominio.Moradia AS Moradia
	  ON Moradia.id_moradia = Moradia_Pessoa.fk_id_moradia
	JOIN adm_condominio.Condominio_Moradia AS Condominio_Moradia
	JOIN adm_condominio.Condominio AS Condominio
	  ON Condominio.id_condominio = Condominio_Moradia.fk_id_condominio
	  ON Condominio_Moradia.fk_id_condominio = Condominio.id_condominio
	JOIN adm_condominio.Endereco AS Endereco
	  ON Endereco.id_endereco = Condominio.fk_id_endereco
	JOIN top_five
	  ON top_five.id_pessoa = Pessoa.id_pessoa;
```


Hash Inner Join
Nested Loop Inner Join
Sort
Aggregate
Hash Inner Join
es_pessoa
Hash
entrada_saida
pessoa_pkey
Nested Loop Inner Join
Sort
Aggregate
es_pessoa
Hash
entrada_saida
Hash
pessoa_pkey
Hash Inner Join
Hash Inner Join
Aggregate
Sort
Limit
pessoa
Hash
CTE Scan
CTE Scan
Hash
condominio_moradia
s
Nested Loop Inner Join
Hash Inner Join
Hash
condominio
Hash Inner Join
Hash Inner Join
Materialize
moradia_pessoa
Hash
pessoa
Hash Inner Join
CTE Scan
Hash
moradia
Hash
Hash
redereco

	QUERY PLAN text
1	Hash Join (cost=278.78..918.19 rows=126560 width=24) (actual time=0.887..0.887 rows=0 loops=1)
2	Hash Cond: (condominio_moradia.fk_id_condominio = condominio.id_condominio)
3	CTE sum_entradas
4	-> GroupAggregate (cost=56.37..56.55 rows=9 width=12) (actual time=0.194..0.202 rows=6 loops=1)
5	Group Key: pessoa_1.id_pessoa
6	-> Sort (cost=56.37..56.39 rows=9 width=12) (actual time=0.181..0.182 rows=6 loops=1)
7	Sort Key: pessoa_1.id_pessoa
8	Sort Method: quicksort Memory: 25kB
9	-> Nested Loop (cost=16.29..56.23 rows=9 width=12) (actual time=0.126..0.169 rows=6 loops=1)
10	-> Hash Join (cost=16.15..54.74 rows=9 width=12) (actual time=0.098..0.106 rows=6 loops=1)
11	Hash Cond: (es_pessoa.fk_id_es = entrada_saida.id_es)
12	-> Seq Scan on es_pessoa (cost=0.00..32.60 rows=2260 width=8) (actual time=0.020..0.022 rows=10 loops=1)
13	-> Hash (cost=16.13..16.13 rows=2 width=12) (actual time=0.051..0.052 rows=11 loops=1)
14	Buckets: 1024 Batches: 1 Memory Usage: 9kB
15	-> Seq Scan on entrada_saida (cost=0.00..16.13 rows=2 width=12) (actual time=0.022..0.027 rows=11 loops=1)
16	Filter: (acao = 'e'::bpchar)
17	Rows Removed by Filter: 9
18	-> Index Only Scan using pessoa_pkey on pessoa pessoa_1 (cost=0.14..0.17 rows=1 width=4) (actual time=0.009..0.009 rows=1 loops=6)
19	Index Cond: (id_pessoa = es_pessoa.fk_id_pessoa)
20	Heap Fetches: 6
21	CTE sum_saidas
22	-> GroupAggregate (cost=56.37..56.55 rows=9 width=12) (actual time=0.111..0.115 rows=4 loops=1)
23	Group Key: pessoa_2.id_pessoa
24	-> Sort (cost=56.37..56.39 rows=9 width=12) (actual time=0.107..0.108 rows=4 loops=1)
25	Sort Key: pessoa_2.id_pessoa
26	Sort Method: quicksort Memory: 25kB
27	-> Nested Loop (cost=16.29..56.23 rows=9 width=12) (actual time=0.083..0.098 rows=4 loops=1)
28	-> Hash Join (cost=16.15..54.74 rows=9 width=12) (actual time=0.069..0.075 rows=4 loops=1)
29	Hash Cond: (es_pessoa_1.fk_id_es = entrada_saida_1.id_es)
30	-> Seq Scan on es_pessoa es_pessoa_1 (cost=0.00..32.60 rows=2260 width=8) (actual time=0.016..0.018 rows=10 loops=1)
31	-> Hash (cost=16.13..16.13 rows=2 width=12) (actual time=0.032..0.032 rows=9 loops=1)
32	Buckets: 1024 Batches: 1 Memory Usage: 9kB
33	-> Seq Scan on entrada_saida entrada_saida_1 (cost=0.00..16.13 rows=2 width=12) (actual time=0.017..0.023 rows=9 loops=1)
34	Filter: (acao = 's'::bpchar)
35	Rows Removed by Filter: 11
36	-> Index Only Scan using pessoa_pkey on pessoa pessoa_2 (cost=0.14..0.17 rows=1 width=4) (actual time=0.005..0.005 rows=1 loops=4)
37	Index Cond: (id_pessoa = es_pessoa_1.fk_id_pessoa)
38	Heap Fetches: 4
39	CTE top_five
40	-> Limit (cost=3.39..3.41 rows=5 width=12) (actual time=0.447..0.447 rows=0 loops=1)
41	-> Sort (cost=3.39..3.42 rows=9 width=12) (actual time=0.446..0.446 rows=0 loops=1)
42	Sort Key: (sum((sum_entradas.soma - sum_saidas.soma))) DESC
43	Sort Method: quicksort Memory: 25kB
44	-> HashAggregate (cost=3.16..3.25 rows=9 width=12) (actual time=0.435..0.435 rows=0 loops=1)
45	Group Key: pessoa_3.id_pessoa
46	-> Hash Join (cost=0.58..3.09 rows=9 width=20) (actual time=0.434..0.434 rows=0 loops=1)
47	Hash Cond: (pessoa_3.id_pessoa = sum_entradas.id_pessoa)
48	-> Hash Join (cost=0.29..2.76 rows=9 width=16) (actual time=0.178..0.200 rows=4 loops=1)
49	Hash Cond: (pessoa_3.id_pessoa = sum_saidas.id_pessoa)
50	-> Seq Scan on pessoa pessoa_3 (cost=0.00..2.00 rows=100 width=4) (actual time=0.025..0.036 rows=100 loops=1)

```
51                      ->  Hash  (cost=0.18..0.18 rows=9 width=12) (actual time=0.131..0.131 rows=4 loops=1)
52                          Buckets: 1024  Batches: 1  Memory Usage: 9kB
53                          ->  CTE Scan on sum_saidas  (cost=0.00..0.18 rows=9 width=12) (actual time=0.113..0.120 rows=4 loops=1)
54                  ->  Hash  (cost=0.18..0.18 rows=9 width=12) (actual time=0.216..0.216 rows=6 loops=1)
55                      Buckets: 1024  Batches: 1  Memory Usage: 9kB
56                      ->  CTE Scan on sum_entradas  (cost=0.00..0.18 rows=9 width=12) (actual time=0.196..0.207 rows=6 loops=1)
57  ->  Seq Scan on condominio_moradia  (cost=0.00..32.60 rows=2260 width=4) (actual time=0.038..0.038 rows=1 loops=1)
58  ->  Hash  (cost=113.27..113.27 rows=3920 width=28) (actual time=0.794..0.794 rows=0 loops=1)
59      Buckets: 4096  Batches: 1  Memory Usage: 32kB
60      ->  Hash Join  (cost=9.93..113.27 rows=3920 width=28) (actual time=0.793..0.793 rows=0 loops=1)
61          Hash Cond: (condominio.fk_id_endereco = endereco.id_endereco)
62          ->  Nested Loop  (cost=5.67..98.30 rows=3920 width=21) (actual time=0.681..0.681 rows=0 loops=1)
63              ->  Seq Scan on condominio  (cost=0.00..1.70 rows=70 width=8) (actual time=0.023..0.029 rows=70 loops=1)
64              ->  Materialize  (cost=5.67..47.74 rows=56 width=13) (actual time=0.009..0.009 rows=0 loops=70)
65                  ->  Hash Join  (cost=5.67..47.46 rows=56 width=13) (actual time=0.622..0.622 rows=0 loops=1)
66                      Hash Cond: (moradia_pessoa.fk_id_moradia = moradia.id_moradia)
67                      ->  Hash Join  (cost=2.65..44.29 rows=56 width=17) (actual time=0.537..0.537 rows=0 loops=1)
68                          Hash Cond: (moradia_pessoa.fk_id_pessoa = pessoa.id_pessoa)
69                          ->  Seq Scan on moradia_pessoa  (cost=0.00..32.60 rows=2260 width=8) (actual time=0.020..0.020 rows=1 loops=1)
70                          ->  Hash  (cost=2.59..2.59 rows=5 width=21) (actual time=0.492..0.492 rows=0 loops=1)
71                              Buckets: 1024  Batches: 1  Memory Usage: 8kB
72                              ->  Hash Join  (cost=0.16..2.59 rows=5 width=21) (actual time=0.492..0.492 rows=0 loops=1)
73                                  Hash Cond: (pessoa.id_pessoa = top_five.id_pessoa)
74                                  ->  Seq Scan on pessoa  (cost=0.00..2.00 rows=100 width=17) (actual time=0.023..0.023 rows=1 loops=1)
75                                  ->  Hash  (cost=0.10..0.10 rows=5 width=4) (actual time=0.449..0.449 rows=0 loops=1)
76                                      Buckets: 1024  Batches: 1  Memory Usage: 8kB
77                                      ->  CTE Scan on top_five  (cost=0.00..0.10 rows=5 width=4) (actual time=0.449..0.449 rows=0 loops=1)
78                  ->  Hash  (cost=1.90..1.90 rows=90 width=4) (actual time=0.066..0.066 rows=90 loops=1)
79                      Buckets: 1024  Batches: 1  Memory Usage: 12kB
80                      ->  Seq Scan on moradia  (cost=0.00..1.90 rows=90 width=4) (actual time=0.023..0.035 rows=90 loops=1)
81          ->  Hash  (cost=3.00..3.00 rows=100 width=15) (actual time=0.092..0.092 rows=100 loops=1)
82              Buckets: 1024  Batches: 1  Memory Usage: 13kB
83              ->  Seq Scan on endereco  (cost=0.00..3.00 rows=100 width=15) (actual time=0.027..0.051 rows=100 loops=1)
84  Planning Time: 2.393 ms
85  Execution Time: 1.533 ms
```

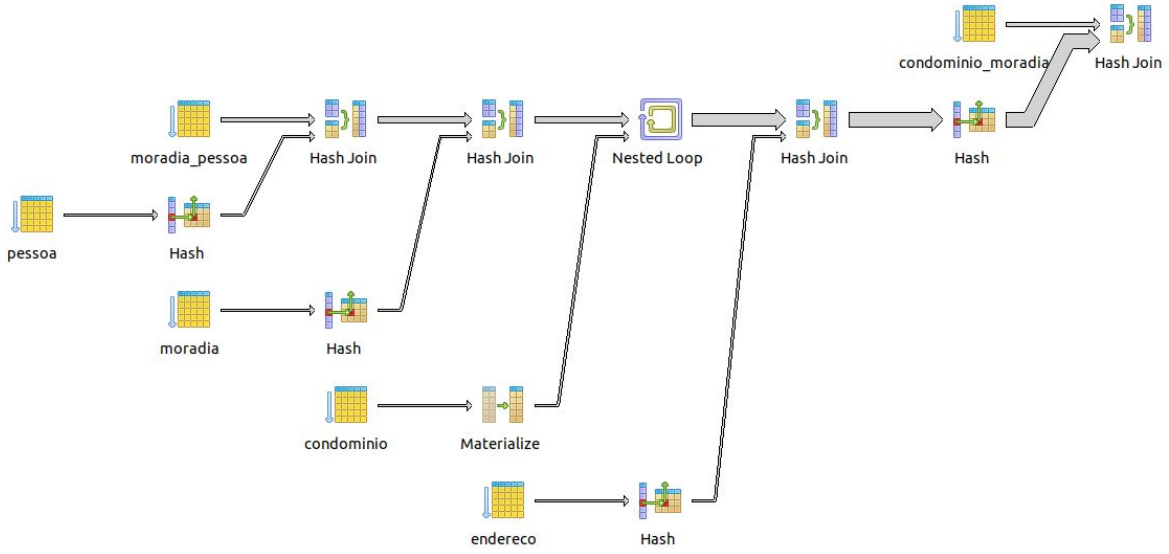
a)

```
WITH sum_entradas AS (
        SELECT pessoa.id_pessoa AS id_pessoa, SUM(DATE_PART('epoch', Entrada_Saida.data_hora)) AS soma
        FROM adm_condominio.Pessoa AS Pessoa
        JOIN adm_condominio.Es_Pessoa AS Es_Pessoa ON Pessoa.id_pessoa = Es_Pessoa.fk_id_pessoa
        JOIN adm_condominio.Entrada_Saida AS Entrada_Saida ON Entrada_Saida.id_es = Es_Pessoa.fk_id_es
        WHERE Entrada_Saida.acao = 'e'
        GROUP BY 1
),

sum_saidas AS (
        SELECT pessoa.id_pessoa AS id_pessoa, SUM(DATE_PART('epoch', Entrada_Saida.data_hora )) AS soma
        FROM adm_condominio.Pessoa AS Pessoa
        JOIN adm_condominio.Es_Pessoa AS Es_Pessoa ON Pessoa.id_pessoa = Es_Pessoa.fk_id_pessoa
        JOIN adm_condominio.Entrada_Saida AS Entrada_Saida ON Entrada_Saida.id_es = Es_Pessoa.fk_id_es
        WHERE Entrada_Saida.acao = 's'
        GROUP BY 1
```

)

```
SELECT Pessoa.nome, Endereco.cidade
        FROM adm_condominio.Pessoa AS Pessoa
        JOIN adm_condominio.Moradia_Pessoa AS Moradia_Pessoa
        ON Moradia_Pessoa.fk_id_pessoa = Pessoa.id_pessoa
        JOIN adm_condominio.Moradia AS Moradia
        ON Moradia.id_moradia = Moradia_Pessoa.fk_id_moradia
        JOIN adm_condominio.Condominio_Moradia AS Condominio_Moradia
        JOIN adm_condominio.Condominio AS Condominio
        ON Condominio.id_condominio = Condominio_Moradia.fk_id_condominio
        ON Condominio_Moradia.fk_id_condominio = Condominio.id_condominio
        JOIN adm_condominio.Endereco AS Endereco
        ON Endereco.id_endereco = Condominio.fk_id_endereco;
```



	QUERY PLAN text
1	Hash Join (cost=5526.98..30926.33 rows=5107600 width=23) (actual time=1.886..2.683 rows=1600 loops=1)
2	Hash Cond: (condominio moradia.fk id condominio = condominio.id condominio)
3	-> Seq Scan on condominio moradia (cost=0.00..32.60 rows=2260 width=4) (actual time=0.006..0.014 rows=40 loops=1)
4	-> Hash (cost=2467.48..2467.48 rows=158200 width=27) (actual time=1.775..1.775 rows=2800 loops=1)
5	Buckets: 65536 Batches: 4 Memory Usage: 554kB
6	-> Hash Join (cost=10.53..2467.48 rows=158200 width=27) (actual time=0.117..1.080 rows=2800 loops=1)
7	Hash Cond: (condominio.fk id endereco = endereco.id endereco)
8	-> Nested Loop (cost=6.28..2030.36 rows=158200 width=21) (actual time=0.078..0.529 rows=2800 loops=1)
9	-> Hash Join (cost=6.28..50.98 rows=2260 width=13) (actual time=0.070..0.105 rows=40 loops=1)
10	Hash Cond: (moradia pessoa.fk id moradia = moradia.id moradia)
11	-> Hash Join (cost=3.25..41.90 rows=2260 width=17) (actual time=0.037..0.060 rows=40 loops=1)
12	Hash Cond: (moradia pessoa.fk id pessoa = pessoa.id pessoa)
13	-> Seq Scan on moradia pessoa (cost=0.00..32.60 rows=2260 width=8) (actual time=0.003..0.009 rows=40 loops=1)
14	-> Hash (cost=2.00..2.00 rows=100 width=17) (actual time=0.031..0.031 rows=100 loops=1)
15	Buckets: 1024 Batches: 1 Memory Usage: 13kB
16	-> Seq Scan on pessoa (cost=0.00..2.00 rows=100 width=17) (actual time=0.005..0.016 rows=100 loops=1)
17	-> Hash (cost=1.90..1.90 rows=90 width=4) (actual time=0.030..0.030 rows=90 loops=1)
18	Buckets: 1024 Batches: 1 Memory Usage: 12kB
19	-> Seq Scan on moradia (cost=0.00..1.90 rows=90 width=4) (actual time=0.004..0.019 rows=90 loops=1)
20	-> Materialize (cost=0.00..2.05 rows=70 width=8) (actual time=0.000..0.004 rows=70 loops=40)
21	-> Seq Scan on condominio (cost=0.00..1.70 rows=70 width=8) (actual time=0.004..0.013 rows=70 loops=1)
22	-> Hash (cost=3.00..3.00 rows=100 width=14) (actual time=0.033..0.033 rows=100 loops=1)
23	Buckets: 1024 Batches: 1 Memory Usage: 13kB
24	-> Seq Scan on endereco (cost=0.00..3.00 rows=100 width=14) (actual time=0.006..0.018 rows=100 loops=1)
25	Planning time: 0.365 ms
26	Execution time: 2.788 ms

Mudanças realizadas: Retiramos a subquery ‘top_five’.

modificações que ocorreram no plano de execução: De cara ja podemos notar que a altura do plano de execução diminuiu muito, isso se deve ao fato de que a subquery top_five era muito custosa, possuindo funções de agregação e diversos JOINs aninhados além de uma função de ordenação.

b)

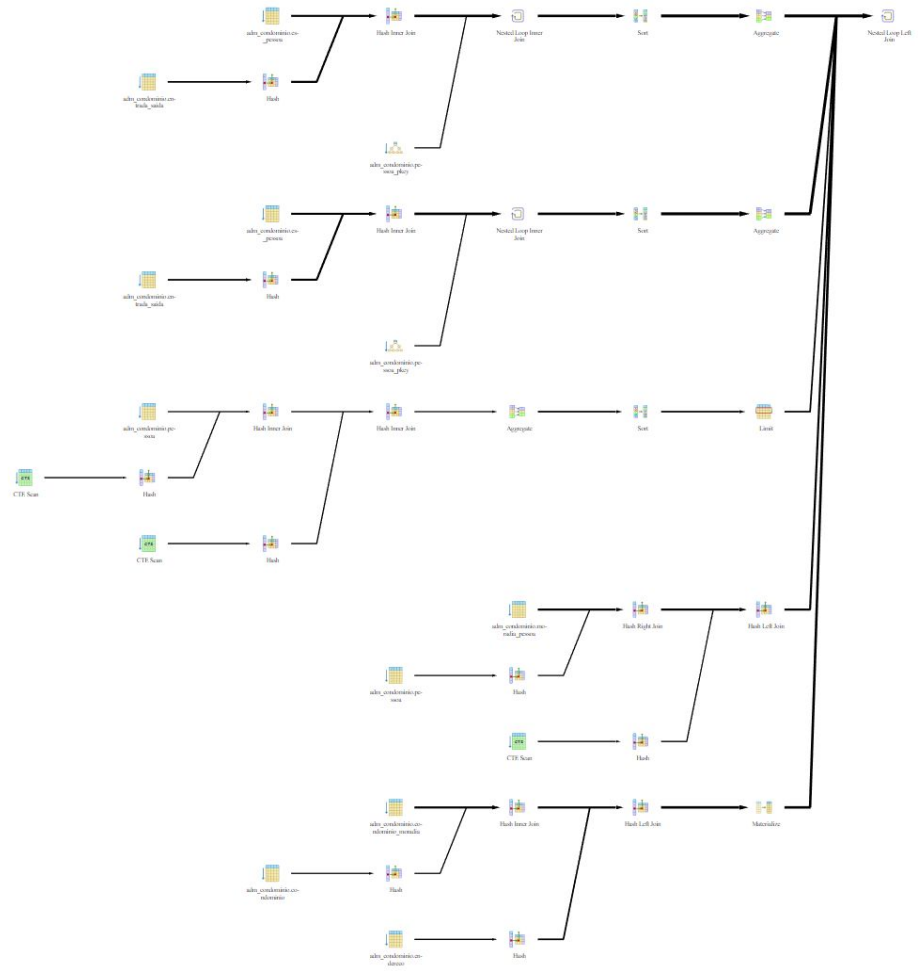
```
WITH sum_entradas AS (
		SELECT pessoa.id_pessoa AS id_pessoa, SUM(DATE_PART('epoch', Entrada_Saida.data_hora)) AS soma
		FROM adm_condominio.Pessoa AS Pessoa
		JOIN adm_condominio.Es_Pessoa AS Es_Pessoa ON Pessoa.id_pessoa = Es_Pessoa.fk_id_pessoa
		JOIN adm_condominio.Entrada_Saida AS Entrada_Saida ON Entrada_Saida.id_es = Es_Pessoa.fk_id_es
		WHERE Entrada_Saida.acao = 'e'
		GROUP BY 1
),

sum_saidas AS (
		SELECT pessoa.id_pessoa AS id_pessoa, SUM(DATE_PART('epoch', Entrada_Saida.data_hora)) AS soma
		FROM adm_condominio.Pessoa AS Pessoa
		JOIN adm_condominio.Es_Pessoa AS Es_Pessoa ON Pessoa.id_pessoa = Es_Pessoa.fk_id_pessoa
		JOIN adm_condominio.Entrada_Saida AS Entrada_Saida ON Entrada_Saida.id_es = Es_Pessoa.fk_id_es
		WHERE Entrada_Saida.acao = 's'
		GROUP BY 1
),

top_five AS (
	SELECT Pessoa.id_pessoa, SUM(sum_entradas.soma - sum_saidas.soma)
		FROM adm_condominio.Pessoa AS Pessoa
		JOIN sum_entradas ON sum_entradas.id_pessoa = Pessoa.id_pessoa
		JOIN sum_saidas ON sum_saidas.id_pessoa = Pessoa.id_pessoa
		GROUP BY 1 ORDER BY 2 DESC
		LIMIT 5
)

SELECT Pessoa.nome, Endereco.cidade
	FROM adm_condominio.Pessoa AS Pessoa
LEFT JOIN adm_condominio.Moradia_Pessoa AS Moradia_Pessoa
	 ON Moradia_Pessoa.fk_id_pessoa = Pessoa.id_pessoa
LEFT JOIN adm_condominio.Moradia AS Moradia
	 ON Moradia.id_moradia = Moradia_Pessoa.fk_id_moradia
LEFT JOIN adm_condominio.Condominio_Moradia AS Condominio_Moradia
LEFT JOIN adm_condominio.Condominio AS Condominio
	 ON Condominio.id_condominio = Condominio_Moradia.fk_id_condominio
	 ON Condominio_Moradia.fk_id_condominio = Condominio.id_condominio
LEFT JOIN adm_condominio.Endereco AS Endereco
```

```
        ON Endereco.id_endereco = Condominio.fk_id_endereco
LEFT JOIN top_five
        ON top_five.id_pessoa = Pessoa.id_pessoa;
```



	QUERY PLAN text
1	Nested Loop Left Join (cost=126.74..64070.50 rows=5107600 width=24) (actual time=0.609..1.552 rows=4000 loops=1)
2	CTE sum_entradas
3	-> GroupAggregate (cost=56.37..56.55 rows=9 width=12) (actual time=0.097..0.102 rows=6 loops=1)
4	Group Key: pessoa_1.id_pessoa
5	-> Sort (cost=56.37..56.39 rows=9 width=12) (actual time=0.089..0.089 rows=6 loops=1)
6	Sort Key: pessoa_1.id_pessoa
7	Sort Method: quicksort Memory: 25kB
8	-> Nested Loop (cost=16.29..56.23 rows=9 width=12) (actual time=0.065..0.079 rows=6 loops=1)
9	-> Hash Join (cost=16.15..54.74 rows=9 width=12) (actual time=0.049..0.054 rows=6 loops=1)
10	Hash Cond: (es_pessoa.fk_id_es = entrada_saida.id_es)
11	-> Seq Scan on es_pessoa (cost=0.00..32.60 rows=2260 width=8) (actual time=0.014..0.015 rows=10 loops=1)
12	-> Hash (cost=16.13..16.13 rows=2 width=12) (actual time=0.022..0.022 rows=11 loops=1)
13	Buckets: 1024 Batches: 1 Memory Usage: 9kB
14	-> Seq Scan on entrada_saida (cost=0.00..16.13 rows=2 width=12) (actual time=0.013..0.017 rows=11 loops=1)
15	Filter: (acao = 'e'::bpchar)
16	Rows Removed by Filter: 9
17	-> Index Only Scan using pessoa_pkey on pessoa pessoa_1 (cost=0.14..0.17 rows=1 width=4) (actual time=0.004..0.004 rows=1 loops=6)
18	Index Cond: (id_pessoa = es_pessoa.fk_id_pessoa)
19	Heap Fetches: 6
20	CTE sum_saidas
21	-> GroupAggregate (cost=56.37..56.55 rows=9 width=12) (actual time=0.064..0.067 rows=4 loops=1)
22	Group Key: pessoa_2.id_pessoa
23	-> Sort (cost=56.37..56.39 rows=9 width=12) (actual time=0.062..0.062 rows=4 loops=1)
24	Sort Key: pessoa_2.id_pessoa
25	Sort Method: quicksort Memory: 25kB
26	-> Nested Loop (cost=16.29..56.23 rows=9 width=12) (actual time=0.048..0.056 rows=4 loops=1)
27	-> Hash Join (cost=16.15..54.74 rows=9 width=12) (actual time=0.040..0.043 rows=4 loops=1)
28	Hash Cond: (es_pessoa_1.fk_id_es = entrada_saida_1.id_es)
29	-> Seq Scan on es_pessoa es_pessoa_1 (cost=0.00..32.60 rows=2260 width=8) (actual time=0.009..0.010 rows=10 loops=1)
30	-> Hash (cost=16.13..16.13 rows=2 width=12) (actual time=0.018..0.018 rows=9 loops=1)
31	Buckets: 1024 Batches: 1 Memory Usage: 9kB
32	-> Seq Scan on entrada_saida entrada_saida_1 (cost=0.00..16.13 rows=2 width=12) (actual time=0.010..0.014 rows=9 loops=1)
33	Filter: (acao = 's'::bpchar)
34	Rows Removed by Filter: 11
35	-> Index Only Scan using pessoa_pkey on pessoa pessoa_2 (cost=0.14..0.17 rows=1 width=4) (actual time=0.003..0.003 rows=1 loops=4)
36	Index Cond: (id_pessoa = es_pessoa_1.fk_id_pessoa)
37	Heap Fetches: 4
38	CTE top_five
39	-> Limit (cost=3.39..3.41 rows=5 width=12) (actual time=0.377..0.377 rows=0 loops=1)
40	-> Sort (cost=3.39..3.42 rows=9 width=12) (actual time=0.376..0.376 rows=0 loops=1)
41	Sort Key: (sum((sum_entradas.soma - sum_saidas.soma))) DESC
42	Sort Method: quicksort Memory: 25kB
43	-> HashAggregate (cost=3.16..3.25 rows=9 width=12) (actual time=0.356..0.356 rows=0 loops=1)
44	Group Key: pessoa_3.id_pessoa
45	-> Hash Join (cost=0.58..3.09 rows=9 width=20) (actual time=0.355..0.355 rows=0 loops=1)
46	Hash Cond: (pessoa_3.id_pessoa = sum_entradas.id_pessoa)
47	-> Hash Join (cost=0.29..2.76 rows=9 width=16) (actual time=0.104..0.117 rows=4 loops=1)
48	Hash Cond: (pessoa_3.id_pessoa = sum_saidas.id_pessoa)
49	-> Seq Scan on pessoa pessoa_3 (cost=0.00..2.00 rows=100 width=4) (actual time=0.012..0.017 rows=100 loops=1)
50	-> Hash (cost=0.18..0.18 rows=9 width=12) (actual time=0.076..0.076 rows=4 loops=1)
51	Buckets: 1024 Batches: 1 Memory Usage: 9kB
52	-> CTE Scan on sum_saidas (cost=0.00..0.18 rows=9 width=12) (actual time=0.065..0.071 rows=4 loops=1)
53	-> Hash (cost=0.18..0.18 rows=9 width=12) (actual time=0.110..0.110 rows=6 loops=1)
54	Buckets: 1024 Batches: 1 Memory Usage: 9kB
55	-> CTE Scan on sum_entradas (cost=0.00..0.18 rows=9 width=12) (actual time=0.099..0.105 rows=6 loops=1)
56	-> Hash Left Join (cost=3.41..51.67 rows=2260 width=13) (actual time=0.480..0.520 rows=100 loops=1)
57	Hash Cond: (pessoa.id_pessoa = top_five.id_pessoa)
58	-> Hash Right Join (cost=3.25..41.90 rows=2260 width=17) (actual time=0.086..0.114 rows=100 loops=1)

```
59          Hash Cond: (moradia_pessoa.fk_id_pessoa = pessoa.id_pessoa)
60          -> Seq Scan on moradia_pessoa  (cost=0.00..32.60 rows=2260 width=8) (actual time=0.013..0.015 rows=40 loops=1)
61          -> Hash  (cost=2.00..2.00 rows=100 width=17) (actual time=0.060..0.060 rows=100 loops=1)
62                Buckets: 1024  Batches: 1  Memory Usage: 13kB
63                -> Seq Scan on pessoa  (cost=0.00..2.00 rows=100 width=17) (actual time=0.021..0.035 rows=100 loops=1)
64       -> Hash  (cost=0.10..0.10 rows=5 width=4) (actual time=0.378..0.378 rows=0 loops=1)
65          Buckets: 1024  Batches: 1  Memory Usage: 8kB
66          -> CTE Scan on top_five  (cost=0.00..0.10 rows=5 width=4) (actual time=0.378..0.378 rows=0 loops=1)
67    -> Materialize  (cost=6.83..62.97 rows=2260 width=11) (actual time=0.001..0.004 rows=40 loops=100)
68       -> Hash Left Join  (cost=6.83..51.67 rows=2260 width=11) (actual time=0.125..0.145 rows=40 loops=1)
69          Hash Cond: (condominio.fk_id_endereco = endereco.id_endereco)
70          -> Hash Join  (cost=2.58..41.23 rows=2260 width=4) (actual time=0.060..0.073 rows=40 loops=1)
71                Hash Cond: (condominio_moradia.fk_id_condominio = condominio.id_condominio)
72                -> Seq Scan on condominio_moradia  (cost=0.00..32.60 rows=2260 width=4) (actual time=0.013..0.016 rows=40 loops=1)
73                -> Hash  (cost=1.70..1.70 rows=70 width=8) (actual time=0.036..0.036 rows=70 loops=1)
74                      Buckets: 1024  Batches: 1  Memory Usage: 11kB
75                      -> Seq Scan on condominio  (cost=0.00..1.70 rows=70 width=8) (actual time=0.012..0.022 rows=70 loops=1)
76          -> Hash  (cost=3.00..3.00 rows=100 width=15) (actual time=0.052..0.052 rows=100 loops=1)
77                Buckets: 1024  Batches: 1  Memory Usage: 13kB
78                -> Seq Scan on endereco  (cost=0.00..3.00 rows=100 width=15) (actual time=0.014..0.027 rows=100 loops=1)
79    Planning Time: 1.011 ms
80    Execution Time: 1.973 ms
```

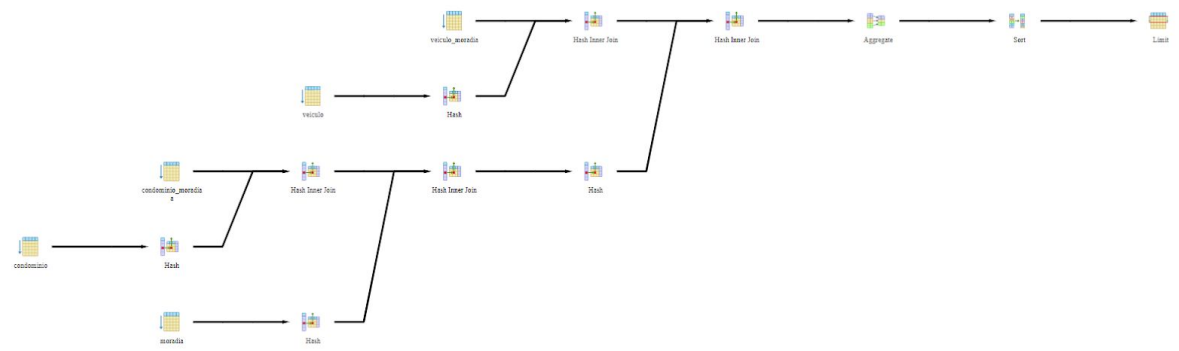
Mudanças realizadas: Subsituímos todos os INNER JOIN por LEFT JOIN

modificações que ocorreram no plano de execução: O tempo de execução aumentou - porém apenas levemente. Num geral, o uso de LEFT JOIN diminui a complexidade a curto prazo, mas o percorrimento de múltiplas relações pode ter ocasionada esse aumento sensível - e considerando que o valor apresentado acima está pelo sistema de *caching* do SGBD.

Query III

- Original

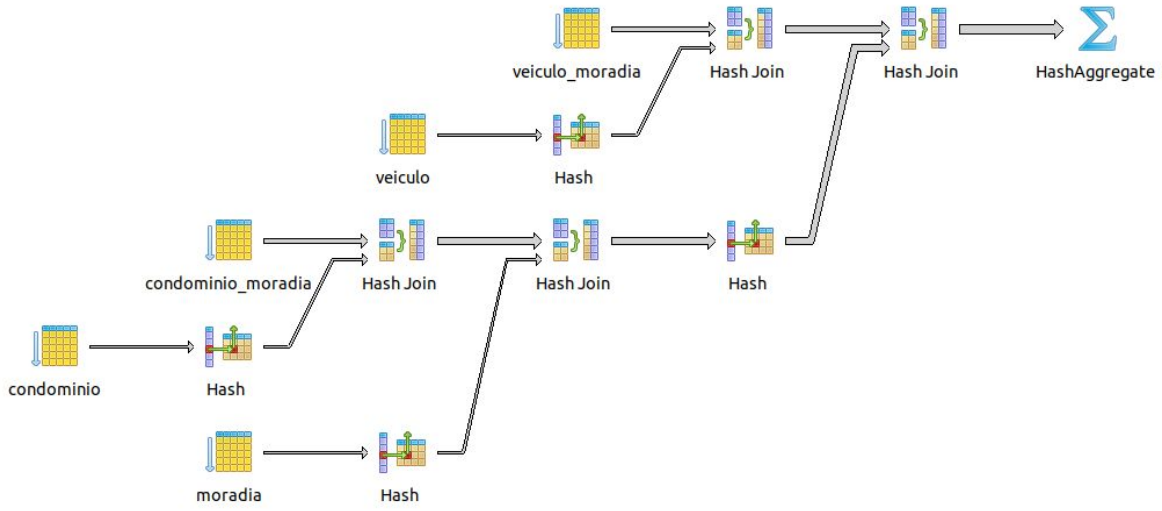
```
SELECT v.marca, COUNT(v.*) FROM adm_condominio.Veiculo AS v
        INNER JOIN adm_condominio.Veiculo_Moradia AS vM ON vM.fk_id_veiculo = v.id_veiculo
        INNER JOIN adm_condominio.Moradia AS m ON m.id_moradia = vM.fk_id_moradia AND m.tipo_moradia = 'a'
        INNER JOIN adm_condominio.Condominio_Moradia AS cM ON cM.fk_id_moradia = m.id_moradia
        INNER JOIN adm_condominio.Condominio AS c ON c.id_condominio = cM.fk_id_condominio AND c.tipo_condominio = 'e'
        GROUP BY v.marca
        ORDER BY 2
        LIMIT 3;
```



	QUERY PLAN text
1	Limit (cost=327.84..327.85 rows=3 width=16) (actual time=0.315..0.318 rows=3 loops=1)
2	-> Sort (cost=327.84..327.93 rows=34 width=16) (actual time=0.314..0.314 rows=3 loops=1)
3	Sort Key: (count(v.*))
4	Sort Method: quicksort Memory: 25kB
5	-> HashAggregate (cost=327.06..327.40 rows=34 width=16) (actual time=0.291..0.293 rows=4 loops=1)
6	Group Key: v.marca
7	-> Hash Join (cost=59.20..228.71 rows=19671 width=76) (actual time=0.274..0.284 rows=4 loops=1)
8	Hash Cond: (vm.fk_id_moradia = m.id_moradia)
9	-> Hash Join (cost=2.35..41.00 rows=2260 width=80) (actual time=0.129..0.138 rows=20 loops=1)
10	Hash Cond: (vm.fk_id_veiculo = v.id_veiculo)
11	-> Seq Scan on veiculo_moradia vm (cost=0.00..32.60 rows=2260 width=8) (actual time=0.029..0.030 rows=20 loops=1)
12	-> Hash (cost=1.60..1.60 rows=60 width=80) (actual time=0.082..0.082 rows=60 loops=1)
13	Buckets: 1024 Batches: 1 Memory Usage: 15kB
14	-> Seq Scan on veiculo v (cost=0.00..1.60 rows=60 width=80) (actual time=0.030..0.059 rows=60 loops=1)
15	-> Hash (cost=47.06..47.06 rows=783 width=8) (actual time=0.132..0.132 rows=9 loops=1)
16	Buckets: 1024 Batches: 1 Memory Usage: 9kB
17	-> Hash Join (cost=5.20..47.06 rows=783 width=8) (actual time=0.114..0.126 rows=9 loops=1)
18	Hash Cond: (cm.fk_id_moradia = m.id_moradia)
19	-> Hash Join (cost=2.34..40.99 rows=1195 width=4) (actual time=0.061..0.070 rows=19 loops=1)
20	Hash Cond: (cm.fk_id_condominio = c.id_condominio)
21	-> Seq Scan on condominio_moradia cm (cost=0.00..32.60 rows=2260 width=8) (actual time=0.013..0.015 rows=40 loops=1)
22	-> Hash (cost=1.88..1.88 rows=37 width=4) (actual time=0.034..0.034 rows=37 loops=1)
23	Buckets: 1024 Batches: 1 Memory Usage: 10kB
24	-> Seq Scan on condominio c (cost=0.00..1.88 rows=37 width=4) (actual time=0.015..0.027 rows=37 loops=1)
25	Filter: (tipo_condominio = 'e'::bpchar)
26	Rows Removed by Filter: 33
27	-> Hash (cost=2.13..2.13 rows=59 width=4) (actual time=0.042..0.043 rows=59 loops=1)
28	Buckets: 1024 Batches: 1 Memory Usage: 11kB
29	-> Seq Scan on moradia m (cost=0.00..2.13 rows=59 width=4) (actual time=0.017..0.029 rows=59 loops=1)
30	Filter: (tipo_moradia = 'a'::bpchar)
31	Rows Removed by Filter: 31
32	Planning Time: 0.772 ms
33	Execution Time: 0.489 ms

a)

```
SELECT v.marca,v.estado , COUNT(v.*) FROM adm_condominio.Veiculo AS v
        INNER JOIN adm_condominio.Veiculo_Moradia AS vM ON vM.fk_id_veiculo = v.id_veiculo
        INNER JOIN adm_condominio.Moradia AS m ON m.id_moradia = vM.fk_id_moradia AND m.tipo_moradia = 'a'
        INNER JOIN adm_condominio.Condominio_Moradia AS cM ON cM.fk_id_moradia = m.id_moradia
        INNER JOIN adm_condominio.Condominio AS c ON c.id_condominio = cM.fk_id_condominio AND c.tipo_condominio = 'e'
        WHERE v.estado != 'SP'
        GROUP BY v.marca, v.estado;
```



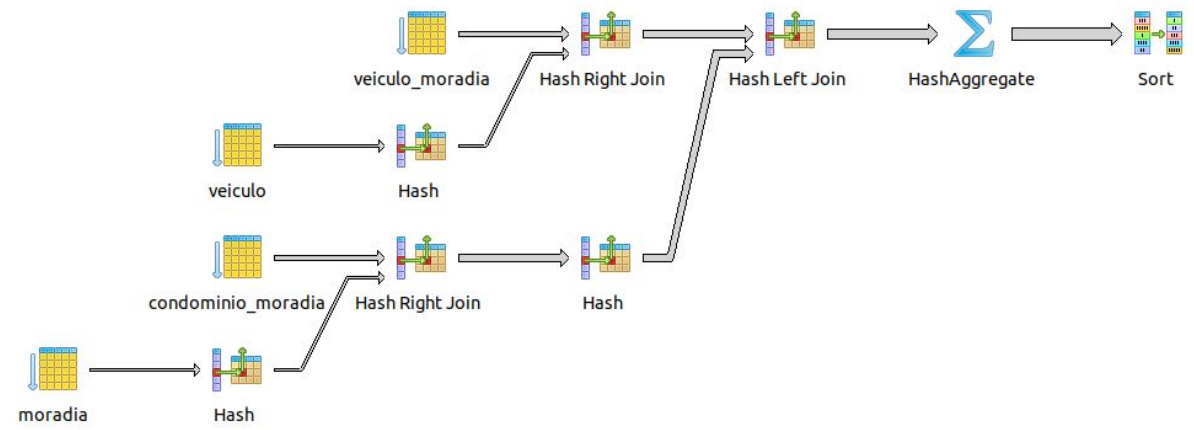
	QUERY PLAN text
1	HashAggregate (cost=172.41..173.41 rows=100 width=17) (actual time=1.230..1.236 rows=39 loops=1)
2	Group Key: pessoa.nome
3	-> Hash Join (cost=148.56..168.30 rows=822 width=17) (actual time=1.165..1.193 rows=125 loops=1)
4	Hash Cond: (assembleia.fk_id_corpo_admin = corpo_administrativo.id_corpo)
5	-> Seq Scan on assembleia (cost=0.00..13.00 rows=300 width=8) (actual time=0.006..0.010 rows=40 loops=1)
6	-> Hash (cost=145.14..145.14 rows=274 width=17) (actual time=1.153..1.153 rows=280 loops=1)
7	Buckets: 1024 Batches: 1 Memory Usage: 22kB
8	-> Nested Loop (cost=0.00..145.14 rows=274 width=17) (actual time=0.020..1.101 rows=280 loops=1)
9	Join Filter: ((pessoa.id_pessoa = corpo_administrativo.id_sindico) OR (pessoa.id_pessoa = corpo_administrativo.id_subsindico) OR (pessoa.id_pessoa = corpo_administrat
10	Rows Removed by Join Filter: 5320
11	-> Seq Scan on pessoa (cost=0.00..2.00 rows=100 width=17) (actual time=0.004..0.013 rows=100 loops=1)
12	-> Materialize (cost=0.00..3.28 rows=56 width=24) (actual time=0.000..0.003 rows=56 loops=100)
13	-> Seq Scan on corpo_administrativo (cost=0.00..3.00 rows=56 width=24) (actual time=0.006..0.034 rows=56 loops=1)
14	Filter: (data_eleicao <= ((now())::date - '2 years'::interval))
15	Rows Removed by Filter: 44
16	Planning time: 0.172 ms
17	Execution time: 1.276 ms

Mudanças realizadas: adicionado a coluna “estado” em nosso select, removemos a ordenação do resultado da query e o limitador de resultados (ORDER BY e LIMIT) e além disso adicionamos uma condição WHERE onde o estado não deve ser SP.

modificações que ocorreram no plano de execução: a retirada de limitadores fazem com que sejam feitas um número maiores de tuplas na relação, o que ocasiona um maior custo na sua projeção, o que justifica o aumento de sua execução.

b)

```
SELECT v.marca, COUNT(v.*) FROM adm_condominio.Veiculo AS v
LEFT JOIN adm_condominio.Veiculo_Moradia AS vM ON vM.fk_id_veiculo = v.id_veiculo
LEFT JOIN adm_condominio.Moradia AS m ON m.id_moradia = vM.fk_id_moradia AND m.tipo_moradia = 'a'
LEFT JOIN adm_condominio.Condominio_Moradia AS cM ON cM.fk_id_moradia = m.id_moradia
LEFT JOIN adm_condominio.Condominio AS c ON c.id_condominio = cM.fk_id_condominio AND c.tipo_condominio = 'e'
GROUP BY v.marca
ORDER BY 2;
```



	QUERY PLAN text
1	Sort (cost=332.60..332.71 rows=41 width=16) (actual time=0.275..0.280 rows=41 loops=1)
2	Sort Key: (count(v.*))
3	Sort Method: quicksort Memory: 27kB
4	-> HashAggregate (cost=331.09..331.50 rows=41 width=16) (actual time=0.237..0.251 rows=41 loops=1)
5	Group Key: v.marca
6	-> Hash Left Join (cost=52.65..203.40 rows=25538 width=76) (actual time=0.154..0.198 rows=80 loops=1)
7	Hash Cond: (vm.fk_id_moradia = m.id_moradia)
8	-> Hash Right Join (cost=2.80..41.45 rows=2260 width=80) (actual time=0.081..0.107 rows=80 loops=1)
9	Hash Cond: (vm.fk_id_veiculo = v.id_veiculo)
10	-> Seq Scan on veiculo_moradia vm (cost=0.00..32.60 rows=2260 width=8) (actual time=0.003..0.005 rows=20 loops=1)
11	-> Hash (cost=1.80..1.80 rows=80 width=80) (actual time=0.072..0.072 rows=80 loops=1)
12	Buckets: 1024 Batches: 1 Memory Usage: 17kB
13	-> Seq Scan on veiculo v (cost=0.00..1.80 rows=80 width=80) (actual time=0.016..0.052 rows=80 loops=1)
14	-> Hash (cost=41.52..41.52 rows=667 width=4) (actual time=0.069..0.069 rows=59 loops=1)
15	Buckets: 1024 Batches: 1 Memory Usage: 11kB
16	-> Hash Right Join (cost=2.86..41.52 rows=667 width=4) (actual time=0.036..0.059 rows=59 loops=1)
17	Hash Cond: (cm.fk_id_moradia = m.id_moradia)
18	-> Seq Scan on condominio_moradia cm (cost=0.00..32.60 rows=2260 width=8) (actual time=0.003..0.006 rows=40 loops=1)
19	-> Hash (cost=2.12..2.12 rows=59 width=4) (actual time=0.030..0.030 rows=59 loops=1)
20	Buckets: 1024 Batches: 1 Memory Usage: 11kB
21	-> Seq Scan on moradia m (cost=0.00..2.12 rows=59 width=4) (actual time=0.005..0.020 rows=59 loops=1)
22	Filter: (tipo_moradia = 'a'::bpchar)
23	Rows Removed by Filter: 31
24	Planning time: 0.370 ms
25	Execution time: 0.345 ms

Mudanças realizadas: Mudamos todos “INNER JOIN” por “LEFT JOIN” e retiramos o limitador.

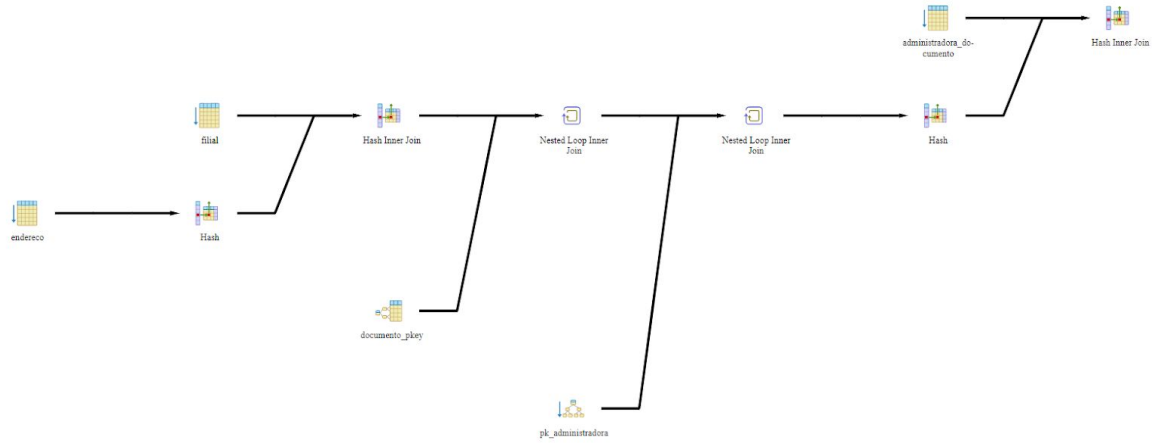
Modificações que ocorreram no plano de execução: O tempo diminuiu levemente. Podemos argumentar que a complexidade diminui levemente devido a não necessidade de uma operação de comparação das tuplas geradas, apesar de isso também ocasionar num maior número de projeções.

Query IV

- Original

```
SELECT Administradora.id_administradora,
                CASE WHEN Documento.status_documento = '1'
                                THEN 'Aprovado'
                        WHEN Documento.status_documento = '2'
                                THEN 'Recusado'
                        WHEN Documento.status_documento = '3'
                                THEN 'Em Aprovação'
                        WHEN Documento.status_documento = '4'
                                THEN 'Em Processamento'
                        WHEN Documento.status_documento = '5'
                                THEN 'Não Disponível'
                        END AS status_documento,
```

```
                Documento.data AS data_criacao
FROM adm_condominio.Administradora AS Administradora
JOIN adm_condominio.Administradora_Documento AS Administradora_Documento
  ON Administradora_Documento.fk_id_documento = Administradora.id_administradora
JOIN adm_condominio.Documento AS Documento
  ON Documento.id_documento = Administradora_Documento.fk_id_documento
JOIN adm_condominio.Filial AS Filial
  ON Filial.fk_id_administradora = Administradora.id_administradora
JOIN adm_condominio.Endereco AS Endereco
  ON Endereco.id_endereco = Filial.fk_id_endereco
WHERE NOT Documento.status_documento = '1'
        AND Endereco.estado IN ('ES', 'MG',  'RJ', 'SP');
```

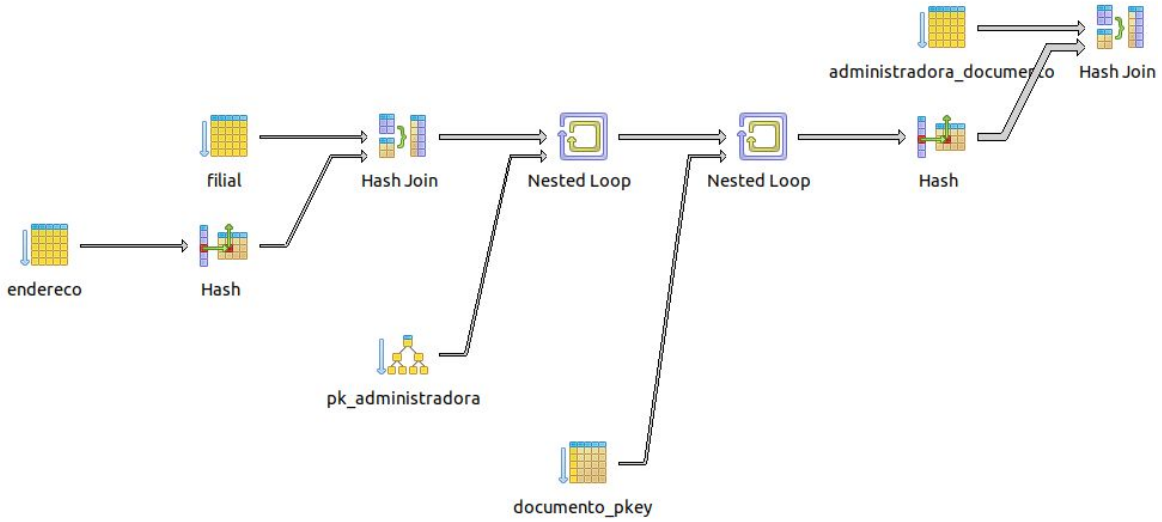


	QUERY PLAN text
1	Hash Join (cost=26.75..69.26 rows=83 width=44) (actual time=0.182..0.182 rows=0 loops=1)
2	Hash Cond: (administradora_documento.fk_id_documento = administradora.id_administradora)
3	-> Seq Scan on administradora_documento (cost=0.00..32.60 rows=2260 width=4) (actual time=0.043..0.043 rows=1 loops=1)
4	-> Hash (cost=26.66..26.66 rows=7 width=78) (actual time=0.113..0.113 rows=0 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 8kB
6	-> Nested Loop (cost=3.84..26.66 rows=7 width=78) (actual time=0.113..0.113 rows=0 loops=1)
7	Join Filter: (documento.id_documento = administradora.id_administradora)
8	-> Nested Loop (cost=3.69..22.67 rows=13 width=74) (actual time=0.113..0.113 rows=0 loops=1)
9	-> Hash Join (cost=3.55..18.24 rows=15 width=4) (actual time=0.112..0.112 rows=0 loops=1)
10	Hash Cond: (filial.fk_id_endereco = endereco.id_endereco)
11	-> Seq Scan on filial (cost=0.00..13.70 rows=370 width=8) (actual time=0.021..0.021 rows=1 loops=1)
12	-> Hash (cost=3.50..3.50 rows=4 width=4) (actual time=0.064..0.064 rows=0 loops=1)
13	Buckets: 1024 Batches: 1 Memory Usage: 8kB
14	-> Seq Scan on endereco (cost=0.00..3.50 rows=4 width=4) (actual time=0.063..0.063 rows=0 loops=1)
15	Filter: ((estado)::text = ANY ('{ES,MG,RJ,SP}'::text[]))
16	Rows Removed by Filter: 100
17	-> Index Scan using documento_pkey on documento (cost=0.14..0.29 rows=1 width=70) (never executed)
18	Index Cond: (id_documento = filial.fk_id_administradora)
19	Filter: ((status_documento)::text <> '1'::text)
20	-> Index Only Scan using pk_administradora on administradora (cost=0.15..0.29 rows=1 width=4) (never executed)
21	Index Cond: (id_administradora = filial.fk_id_administradora)
22	Heap Fetches: 0
23	Planning Time: 1.249 ms
24	Execution Time: 0.264 ms

a)

```
SELECT Administradora.id_administradora AS status_documento,
```

```
Documento.data AS data_criacao
FROM adm_condominio.Administradora AS Administradora
JOIN adm_condominio.Administradora_Documento AS Administradora_Documento
ON Administradora_Documento.fk_id_documento = Administradora.id_administradora
JOIN adm_condominio.Documento AS Documento
ON Documento.id_documento = Administradora_Documento.fk_id_documento
JOIN adm_condominio.Filial AS Filial
ON Filial.fk_id_administradora = Administradora.id_administradora
JOIN adm_condominio.Endereco AS Endereco
ON Endereco.id_endereco = Filial.fk_id_endereco
WHERE NOT Documento.status_documento = '1'
AND Endereco.estado IN ('ES', 'MG',  'RJ', 'SP');
```



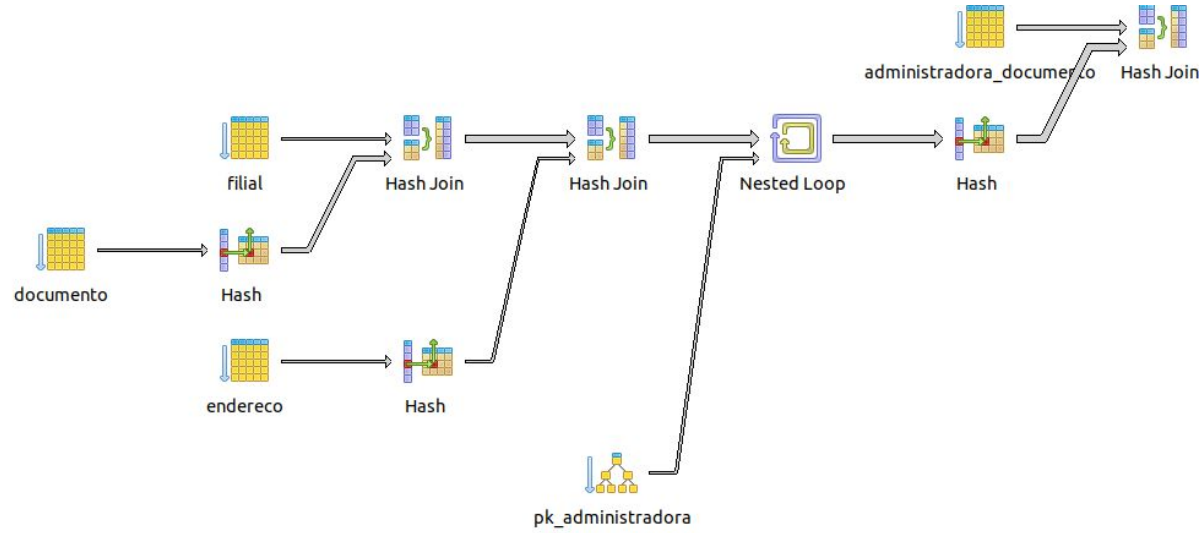
	QUERY PLAN text
1	Hash Join (cost=24.67..66.03 rows=65 width=12) (actual time=0.058..0.058 rows=0 loops=1)
2	Hash Cond: (administradora documento.fk id documento = administradora.id administradora)
3	-> Seq Scan on administradora documento (cost=0.00..32.60 rows=2260 width=4) (actual time=0.008..0.008 rows=1 loops=1)
4	-> Hash (cost=24.61..24.61 rows=5 width=20) (actual time=0.038..0.038 rows=0 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 8kB
6	-> Nested Loop (cost=3.84..24.61 rows=5 width=20) (actual time=0.037..0.037 rows=0 loops=1)
7	-> Nested Loop (cost=3.70..21.19 rows=12 width=8) (actual time=0.037..0.037 rows=0 loops=1)
8	-> Hash Join (cost=3.55..17.22 rows=12 width=4) (actual time=0.037..0.037 rows=0 loops=1)
9	Hash Cond: (filial.fk id endereco = endereco.id endereco)
10	-> Seq Scan on filial (cost=0.00..12.90 rows=290 width=8) (actual time=0.003..0.003 rows=1 loops=1)
11	-> Hash (cost=3.50..3.50 rows=4 width=4) (actual time=0.030..0.030 rows=0 loops=1)
12	Buckets: 1024 Batches: 1 Memory Usage: 8kB
13	-> Seq Scan on endereco (cost=0.00..3.50 rows=4 width=4) (actual time=0.030..0.030 rows=0 loops=1)
14	Filter: ((estado)::text = ANY ('{ES,MG,RJ,SP}'::text[]))
15	Rows Removed by Filter: 100
16	-> Index Only Scan using pk administradora on administradora (cost=0.15..0.33 rows=1 width=4) (never executed)
17	Index Cond: (id administradora = filial.fk id administradora)
18	Heap Fetches: 0
19	-> Index Scan using documento pkey on documento (cost=0.14..0.29 rows=1 width=12) (never executed)
20	Index Cond: (id documento = administradora.id administradora)
21	Filter: ((status documento)::text <> '1'::text)
22	Planning time: 0.463 ms
23	Execution time: 0.104 ms

Mudanças realizadas: Removemos o CASE WHEN

modificações que ocorreram no plano de execução: Ao removermos o CASE WHEN melhoramos o desempenho da query pois além de usarmos o dado sem conversão (ou seja, não fazemos uma operação de comparação e substituição na projeção), não é mais feitas tantas operações lógicas.

b)

```
SELECT Administradora.id_administradora,
        CASE WHEN Documento.status_documento = '1'
                THEN 'Aprovado'
        WHEN Documento.status_documento = '2'
                THEN 'Recusado'
        WHEN Documento.status_documento = '3'
                THEN 'Em Aprovação'
        WHEN Documento.status_documento = '4'
                THEN 'Em Processamento'
        WHEN Documento.status_documento = '5'
                THEN 'Não Disponível'
        END AS status_documento,
        Documento.data AS data_criacao
        FROM adm_condominio.Administradora AS Administradora
        JOIN adm_condominio.Administradora_Documento AS Administradora_Documento
        ON Administradora_Documento.fk_id_documento = Administradora.id_administradora
        JOIN adm_condominio.Documento AS Documento
        ON Documento.id_documento = Administradora_Documento.fk_id_documento
        JOIN adm_condominio.Filial AS Filial
        ON Filial.fk_id_administradora = Administradora.id_administradora
        JOIN adm_condominio.Endereco AS Endereco
        ON Endereco.id_endereco = Filial.fk_id_endereco
        WHERE Documento.status_documento = '1'
        OR Endereco.estado IN ('ES', 'MG',  'RJ', 'SP');
```



	QUERY PLAN text
1	Hash Join (cost=36.85..79.19 rows=74 width=44) (actual time=0.141..0.141 rows=0 loops=1)
2	Hash Cond: (administradora_documento.fk_id_documento = administradora.id_administradora)
3	-> Seq Scan on administradora_documento (cost=0.00..32.60 rows=2260 width=4) (actual time=0.011..0.012 rows=10 loops=1)
4	-> Hash (cost=36.77..36.77 rows=6 width=78) (actual time=0.123..0.123 rows=4 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 9kB
6	-> Nested Loop (cost=18.45..36.77 rows=6 width=78) (actual time=0.104..0.121 rows=4 loops=1)
7	Join Filter: (documento.id_documento = administradora.id_administradora)
8	-> Hash Join (cost=18.30..32.66 rows=12 width=74) (actual time=0.094..0.104 rows=4 loops=1)
9	Hash Cond: (filial.fk_id_endereco = endereco.id_endereco)
10	Join Filter: (((documento.status_documento)::text = '1'::text) OR ((endereco.estado)::text = ANY ('{ES,MG,RJ,SP}'::text[])))
11	Rows Removed by Join Filter: 16
12	-> Hash Join (cost=14.05..27.72 rows=261 width=78) (actual time=0.035..0.045 rows=20 loops=1)
13	Hash Cond: (filial.fk_id_administradora = documento.id_documento)
14	-> Seq Scan on filial (cost=0.00..12.90 rows=290 width=8) (actual time=0.004..0.006 rows=20 loops=1)
15	-> Hash (cost=11.80..11.80 rows=180 width=70) (actual time=0.026..0.026 rows=50 loops=1)
16	Buckets: 1024 Batches: 1 Memory Usage: 11kB
17	-> Seq Scan on documento (cost=0.00..11.80 rows=180 width=70) (actual time=0.005..0.014 rows=50 loops=1)
18	-> Hash (cost=3.00..3.00 rows=100 width=14) (actual time=0.046..0.046 rows=100 loops=1)
19	Buckets: 1024 Batches: 1 Memory Usage: 13kB
20	-> Seq Scan on endereco (cost=0.00..3.00 rows=100 width=14) (actual time=0.007..0.024 rows=100 loops=1)
21	-> Index Only Scan using pk_administradora on administradora (cost=0.15..0.33 rows=1 width=4) (actual time=0.003..0.003 rows=1 loops=4)
22	Index Cond: (id_administradora = filial.fk_id_administradora)
23	Heap Fetches: 4
24	Planning time: 0.784 ms
25	Execution time: 0.198 ms

Mudanças realizadas: Removemos a negação NOT na cláusula do WHERE e mudamos a condição AND por OR.

modificações que ocorreram no plano de execução: Ao trocarmos AND por OR, alteramos o resultado das consultas mas o plano de execução continua basicamente o mesmo, pois a operação ocorre ao mesmo tempo em ambos casos.