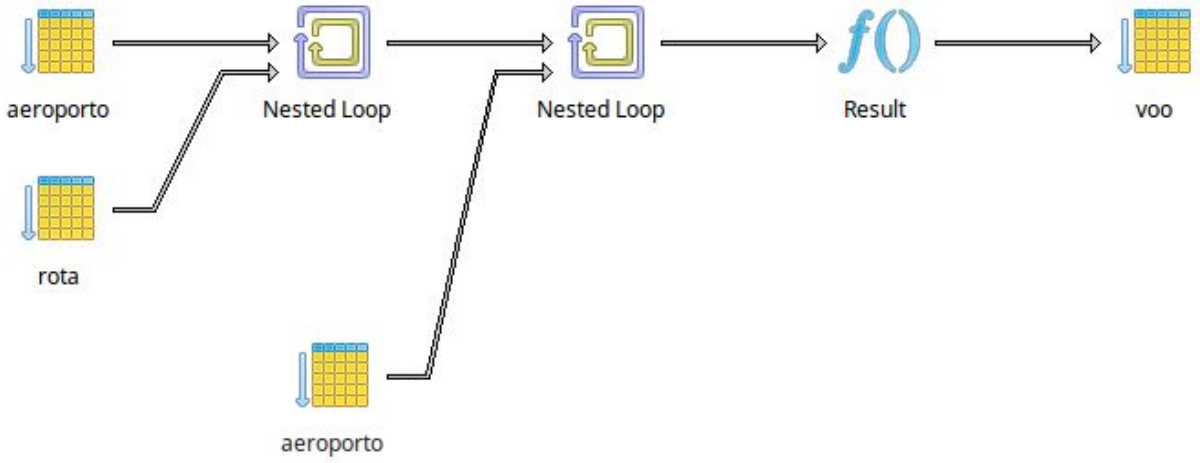
Turma 04

Marcos Henrique Monteiro da Silva	8802392
Marcos Vinicius Oliveira Lunardelli	9019302
Peterson Medeiros Santos	9004550

ANTIGO

```
SELECT * FROM voo
WHERE rot_codigo IN
    (SELECT rot_codigo FROM rota WHERE aer_icao_origem IN
        (SELECT aer_icao FROM aeroporto WHERE aeroporto.aer_loc_cidade='São
Paulo') AND aer_icao_destino IN
              (SELECT aer_icao FROM aeroporto WHERE
aeroporto.aer_loc_cidade='Rio de Janeiro')AND voo_data='01/03/2018');
```

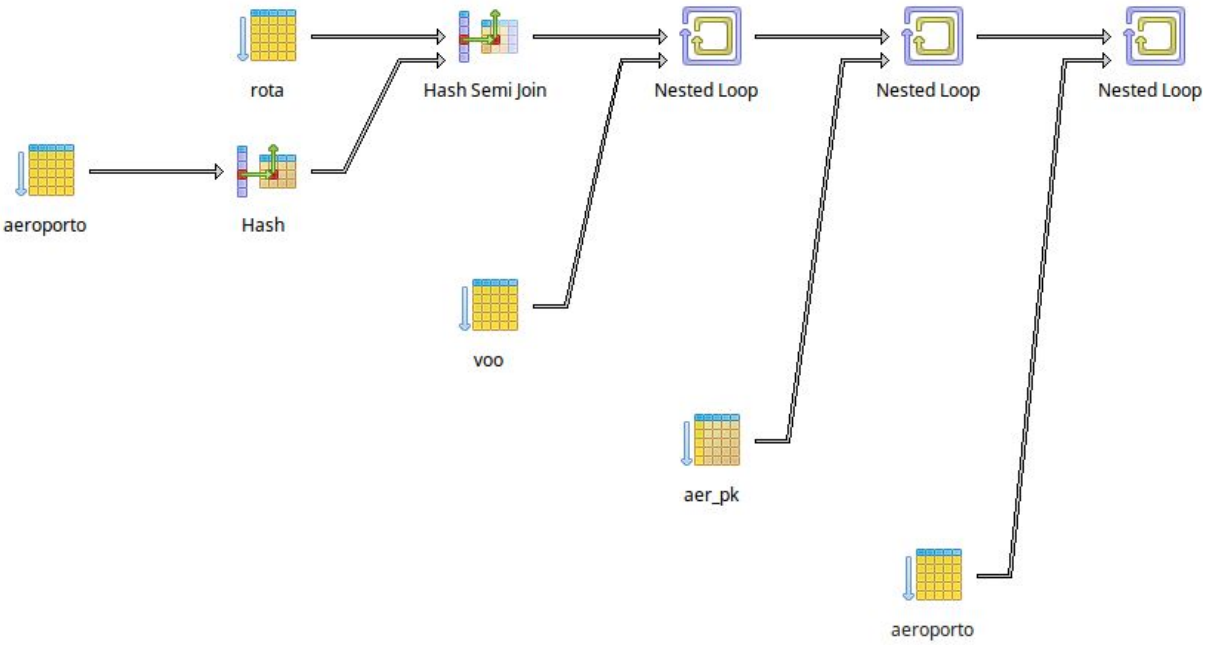


	QUERY PLAN text
1	Seq Scan on voo (cost=0.00..560.25 rows=150 width=36) (actual time=0.062..0.742 rows=1 loops=1)
2	Filter: (SubPlan 1)
3	Rows Removed by Filter: 299
4	SubPlan 1
5	-> Result (cost=0.00..3.69 rows=1 width=4) (actual time=0.000..0.000 rows=0 loops=300)
6	One-Time Filter: (voo.voo_data = '2018-01-03'::date)
7	-> Nested Loop (cost=0.00..3.69 rows=1 width=4) (actual time=0.038..0.038 rows=1 loops=1)
8	Join Filter: (rota.aer_icao_destino = aeroporto_1.aer_icao)
9	Rows Removed by Join Filter: 7
10	-> Nested Loop (cost=0.00..2.47 rows=1 width=24) (actual time=0.016..0.021 rows=3 loops=1)
11	Join Filter: (rota.aer_icao_origem = aeroporto.aer_icao)
12	Rows Removed by Join Filter: 19
13	-> Seq Scan on aeroporto (cost=0.00..1.20 rows=1 width=20) (actual time=0.008..0.008 rows=2 loops=1)
14	Filter: (aer_loc_cidade = 'São Paulo'::bpchar)
15	Rows Removed by Filter: 6
16	-> Seq Scan on rota (cost=0.00..1.12 rows=12 width=44) (actual time=0.003..0.003 rows=11 loops=2)
17	-> Seq Scan on aeroporto aeroporto_1 (cost=0.00..1.20 rows=1 width=20) (actual time=0.003..0.003 rows=3 loops=3
18	Filter: (aer_loc_cidade = 'Rio de Janeiro'::bpchar)
19	Rows Removed by Filter: 13
20	Planning time: 0.310 ms
21	Execution time: 0.780 ms

A primeira versão da consulta faz uso principalmente de junções de laço aninhadas. Se mantido este plano de execução, em um caso típico de uso, a realização de loops duplos pode afetar seriamente o desempenho, pois apenas uma ínfima parte da relação seria relevante para satisfazer a condição dada.

ATUAL

```
SELECT * FROM voo INNER JOIN rota
    ON(voo.rot_codigo = rota.rot_codigo) INNER JOIN aeroporto
    ON(rota.aer_icao_origem = aeroporto.aer_icao)
    WHERE aer_icao_origem IN (SELECT aer_icao FROM aeroporto WHERE
aer_loc_cidade = 'Campinas')
    AND aer_icao_destino IN (SELECT aer_icao FROM aeroporto WHERE aer_loc_cidade
= 'São Paulo') AND voo_data = '08/21/2020';
```



	QUERY PLAN text
1	Nested Loop (cost=1.35..11.54 rows=1 width=429) (actual time=0.118..0.122 rows=1 loops=1)
2	Join Filter: (rota.aer_icao_destino = aeroporto_2.aer_icao)
3	Rows Removed by Join Filter: 1
4	-> Nested Loop (cost=1.35..10.33 rows=1 width=429) (actual time=0.111..0.114 rows=1 loops=1)
5	-> Nested Loop (cost=1.21..9.16 rows=1 width=132) (actual time=0.096..0.098 rows=1 loops=1)
6	Join Filter: (rota.rot_codigo = voo.rot_codigo)
7	Rows Removed by Join Filter: 2
8	-> Hash Semi Join (cost=1.21..2.38 rows=1 width=96) (actual time=0.048..0.050 rows=1 loops=1)
9	Hash Cond: (rota.aer_icao_origem = aeroporto_1.aer_icao)
10	-> Seq Scan on rota (cost=0.00..1.12 rows=12 width=76) (actual time=0.018..0.018 rows=12 loops=1)
11	-> Hash (cost=1.20..1.20 rows=1 width=20) (actual time=0.018..0.018 rows=2 loops=1)
12	Buckets: 1024 Batches: 1 Memory Usage: 9kB
13	-> Seq Scan on aeroporto aeroporto_1 (cost=0.00..1.20 rows=1 width=20) (actual time=0.012..0.013 rows=2 loops=1)
14	Filter: (aer_loc_cidade = 'Campinas'::bpchar)
15	Rows Removed by Filter: 14
16	-> Seq Scan on voo (cost=0.00..6.75 rows=3 width=36) (actual time=0.015..0.046 rows=3 loops=1)
17	Filter: (voo_data = '2020-08-21'::date)
18	Rows Removed by Filter: 297
19	-> Index Scan using aer_pk on aeroporto (cost=0.14..1.15 rows=1 width=317) (actual time=0.013..0.013 rows=1 loops=1)
20	Index Cond: (aer_icao = rota.aer_icao_origem)
21	-> Seq Scan on aeroporto aeroporto_2 (cost=0.00..1.20 rows=1 width=20) (actual time=0.006..0.007 rows=2 loops=1)
22	Filter: (aer_loc_cidade = 'São Paulo'::bpchar)
23	Rows Removed by Filter: 14
24	Planning time: 0.585 ms
25	Execution time: 0.229 ms

A nova consulta faz uso de hash-join e hash-semi join. Um hash-semi join não junta atributos de uma das duas relações envolvidas, no caso, da relação “rota”, descartando o atributo que seria combinado na relação resultante. Neste caso, ele já existe na relação voo, sendo necessário apenas para verificar a condição de igualdade.

A utilização de uma etapa adicional de hash-semi join aumenta o custo e os dados recuperados (no caso, um atributo, somente) não são utilizados. No entanto, esta verificação se faz necessária pois voos não podem ser feitos sem antes uma verificação da existência de rota entre a origem e o destino já homologada para a companhia.