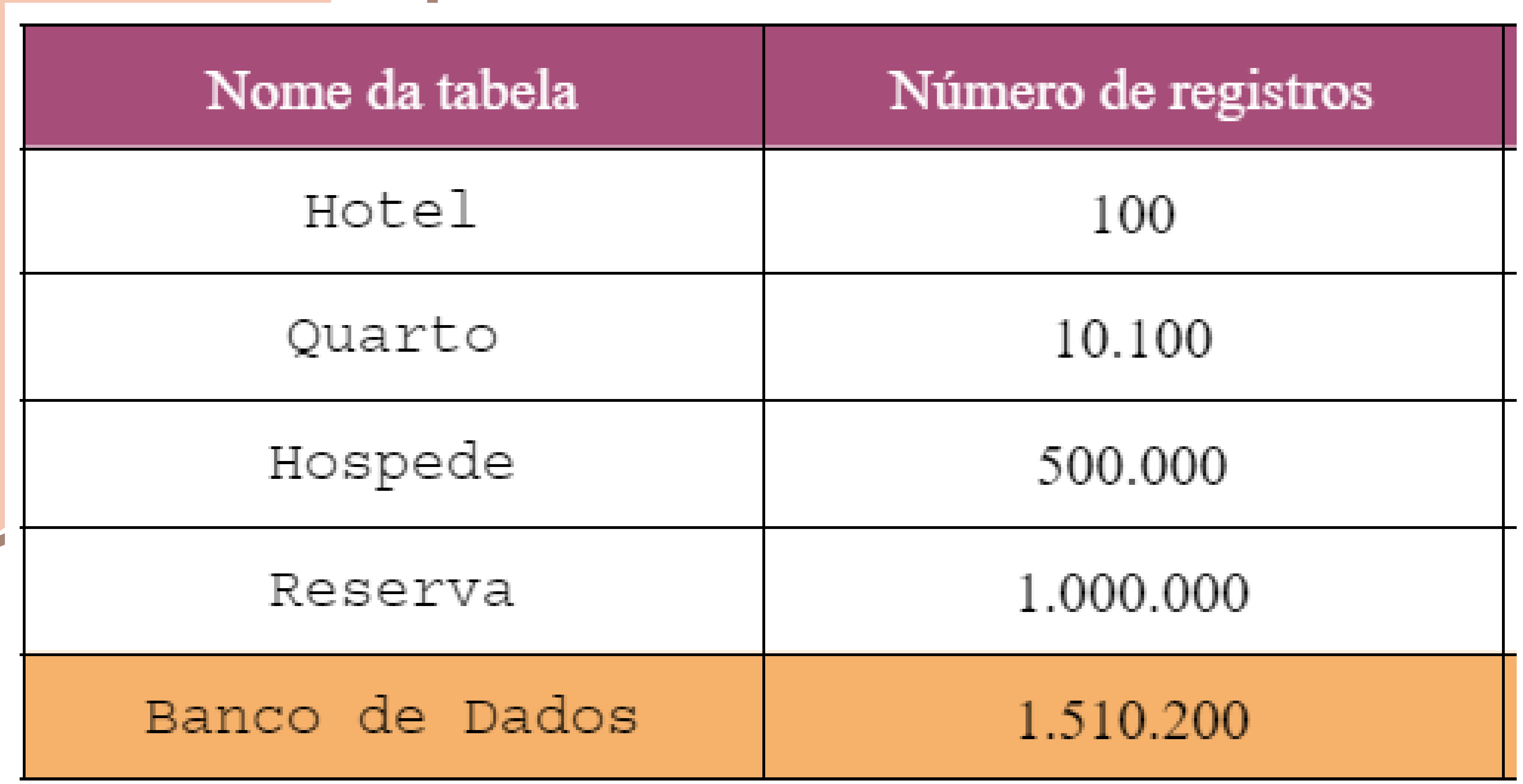
Universidade Federal de São Carlos - Campus Sorocaba
Sistema de Banco de Dados

OTIMIZAÇÃO DE CONSULTAS

REDE DE HOTÉIS HAMPTONS

GRUPO 14:

Felipe Ottoni Pereira - 804317
Letícia Almeida Paulino de Alencar Ferreira - 80480
Ricardo Yugo Suzuki - 802003



Nome da tabela	Número de registros
Hotel	100
Quarto	10.100
Hospede	500.000
Reserva	1.000.000
Banco de Dados	1.510.200

REDE DE HOTÉIS HAMPTONS

CONSULTA 1

Quais quartos na localização X , da categoria Y, entre a data A e a data B, com capacidade Z, estão disponíveis?

Inicial

```
SELECT  numero, id_hotel, nome_hotel, estrelas, endereco,preco
FROM quarto NATURAL JOIN hotel NATURAL JOIN (
    SELECT id_hotel, numero FROM (
        SELECT id_hotel, nome_hotel, estrelas, endereco,   numero, preco
        FROM hotel ho NATURAL JOIN quarto q
        WHERE
        estado LIKE '<X>' AND
        categoria = '<Y>' AND
        capacidade = <Z>
    )t1
    INTERSECT
    SELECT id_hotel, numero  FROM (
        SELECT id_hotel, numero
        FROM quarto
        EXCEPT
        SELECT id_hotel, numero_quarto
        FROM reserva
        WHERE (‘<A>’, ‘<B>’) OVERLAPS
          (data_entrada,data_saida)
    )t2
)t3;
```

QUARTOS COM A LOCALIZACÃO, CATEGORIA E CAPACIDADE DESEJADA

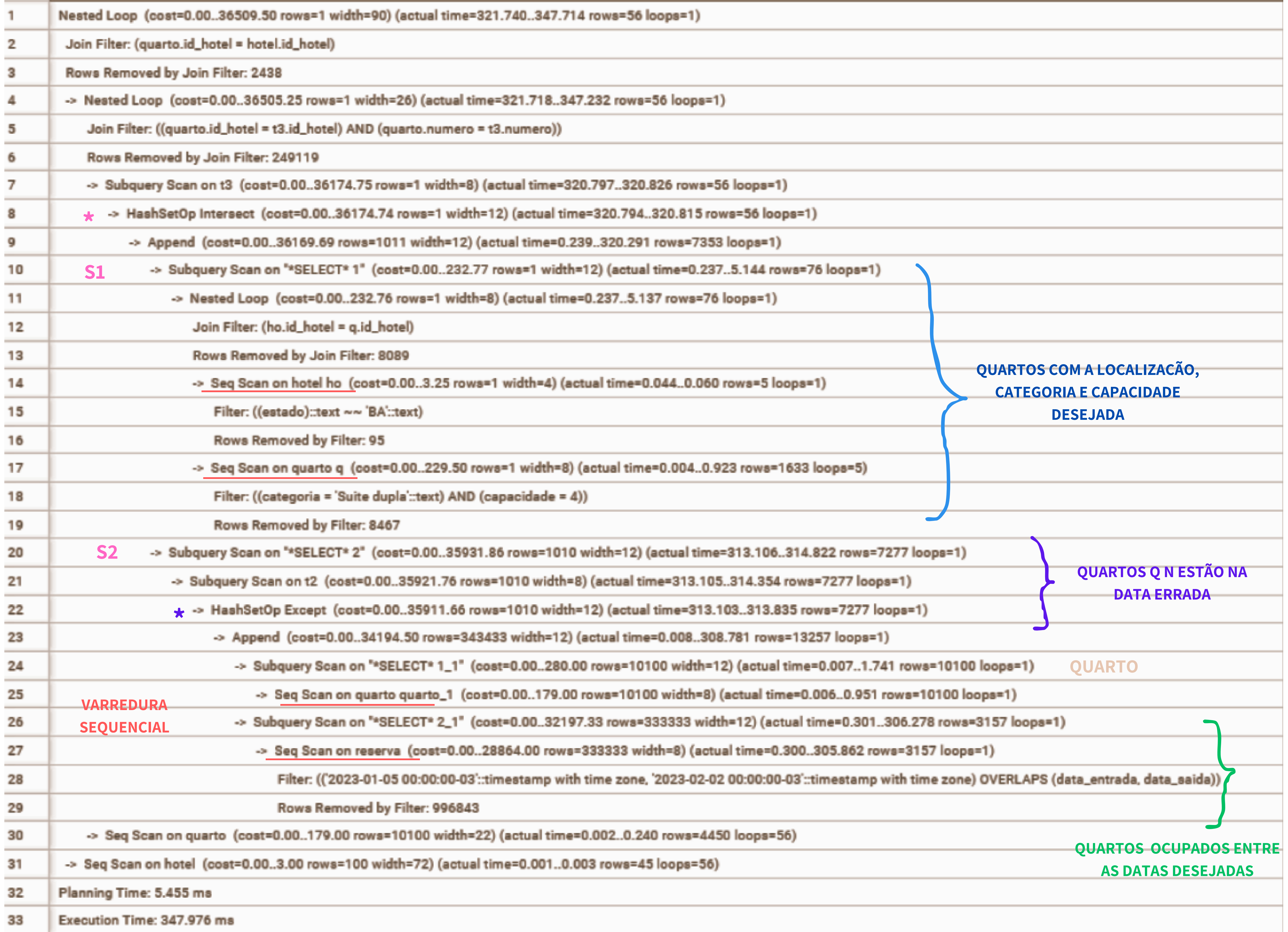
CONSULTA HOTEL E QUARTO

QUARTOS DISPONIVEIS

CONSULTA QUARTO

QUARTOS OCUPADOS ENTRE AS DATAS DESEJADAS

CONSULTA RESERVAS


S1
QUARTOS COM A LOCALIZACÃO, CATEGORIA E CAPACIDADE DESEJADA
S2
QUARTOS Q N ESTÃO NA DATA ERRADA
QUARTO
VARREDURA SEQUENCIAL
QUARTOS OCUPADOS ENTRE AS DATAS DESEJADAS

```
Nested Loop  (cost=0.00..36509.50 rows=1 width=90) (actual time=321.740..347.714 rows=56 loops=1)
  Join Filter: (quarto.id_hotel = hotel.id_hotel)
  Rows Removed by Join Filter: 2438
  ->  Nested Loop  (cost=0.00..36505.25 rows=1 width=26) (actual time=321.718..347.232 rows=56 loops=1)
        Join Filter: ((quarto.id_hotel = t3.id_hotel) AND (quarto.numero = t3.numero))
        Rows Removed by Join Filter: 249119
        ->  Subquery Scan on t3  (cost=0.00..36174.75 rows=1 width=8) (actual time=320.797..320.826 rows=56 loops=1)
              ->  HashSetOp Intersect  (cost=0.00..36174.74 rows=1 width=12) (actual time=320.794..320.815 rows=56 loops=1)
                    ->  Append  (cost=0.00..36169.69 rows=1011 width=12) (actual time=0.239..320.291 rows=7353 loops=1)
                          ->  Subquery Scan on "*SELECT* 1"  (cost=0.00..232.77 rows=1 width=12) (actual time=0.237..5.144 rows=76 loops=1)
                                ->  Nested Loop  (cost=0.00..232.76 rows=1 width=8) (actual time=0.237..5.137 rows=76 loops=1)
                                      Join Filter: (ho.id_hotel = q.id_hotel)
                                      Rows Removed by Join Filter: 8089
                                      ->  Seq Scan on hotel ho  (cost=0.00..3.25 rows=1 width=4) (actual time=0.044..0.060 rows=5 loops=1)
                                            Filter: ((estado)::text ~~ 'BA'::text)
                                            Rows Removed by Filter: 95
                                      ->  Seq Scan on quarto q  (cost=0.00..229.50 rows=1 width=8) (actual time=0.004..0.923 rows=1633 loops=5)
                                            Filter: ((categoria = 'Suite dupla'::text) AND (capacidade = 4))
                                            Rows Removed by Filter: 8467
                          ->  Subquery Scan on "*SELECT* 2"  (cost=0.00..35931.86 rows=1010 width=12) (actual time=313.106..314.822 rows=7277 loops=1)
                                ->  Subquery Scan on t2  (cost=0.00..35921.76 rows=1010 width=8) (actual time=313.105..314.354 rows=7277 loops=1)
                                      ->  HashSetOp Except  (cost=0.00..35911.66 rows=1010 width=12) (actual time=313.103..313.835 rows=7277 loops=1)
                                            ->  Append  (cost=0.00..34194.50 rows=343433 width=12) (actual time=0.008..308.781 rows=13257 loops=1)
                                                  ->  Subquery Scan on "*SELECT* 1_1"  (cost=0.00..280.00 rows=10100 width=12) (actual time=0.007..1.741 rows=10100 loops=1)
                                                        ->  Seq Scan on quarto quarto_1  (cost=0.00..179.00 rows=10100 width=8) (actual time=0.006..0.951 rows=10100 loops=1)
                                                  ->  Subquery Scan on "*SELECT* 2_1"  (cost=0.00..32197.33 rows=333333 width=12) (actual time=0.301..306.278 rows=3157 loops=1)
                                                        ->  Seq Scan on reserva  (cost=0.00..28864.00 rows=333333 width=8) (actual time=0.300..305.862 rows=3157 loops=1)
                                                              Filter: (('2023-01-05 00:00:00-03'::timestamp with time zone, '2023-02-02 00:00:00-03'::timestamp with time zone) OVERLAPS (data_entrada, data_saida))
                                                              Rows Removed by Filter: 996843
        ->  Seq Scan on quarto  (cost=0.00..179.00 rows=10100 width=22) (actual time=0.002..0.240 rows=4450 loops=56)
  ->  Seq Scan on hotel  (cost=0.00..3.00 rows=100 width=72) (actual time=0.001..0.003 rows=45 loops=56)
Planning Time: 5.455 ms
Execution Time: 347.976 ms
```

CONSULTA 1

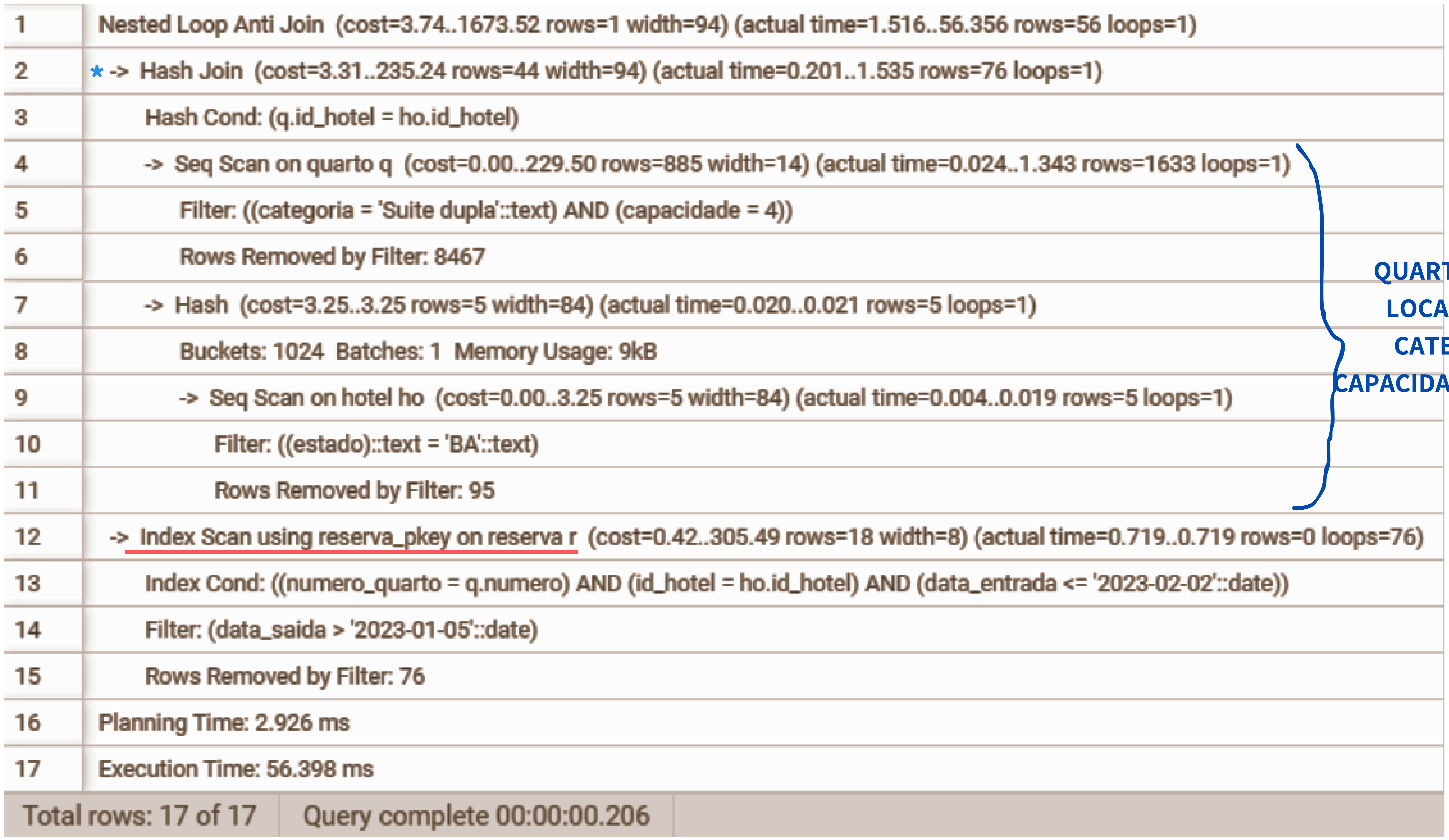
```
CREATE INDEX index_datas on reserva (data_entrada, data_saida);
```

Otimizada

```
SELECT numero, t1.id_hotel, nome_hotel, estrelas, endereco, preco
FROM (
    SELECT ho.id_hotel, numero, nome_hotel, estrelas, endereco, preco
    FROM hotel ho
    NATURAL JOIN quarto q
    WHERE estado = 'BA'
    AND categoria = 'Suite dupla'
    AND capacidade = 4
) t1
WHERE NOT EXISTS (
    SELECT 1
    FROM reserva r
    WHERE r.id_hotel = t1.id_hotel
    AND r.numero_quarto = t1.numero
    AND r.data_entrada <= '2023-02-02' AND r.data_saida > '2023-01-05'
);
```

QUARTOS COM A LOCALIZACÃO, CATEGORIA E CAPACIDADE DESEJADA

Há alguma reserva que esteja nessas condições?

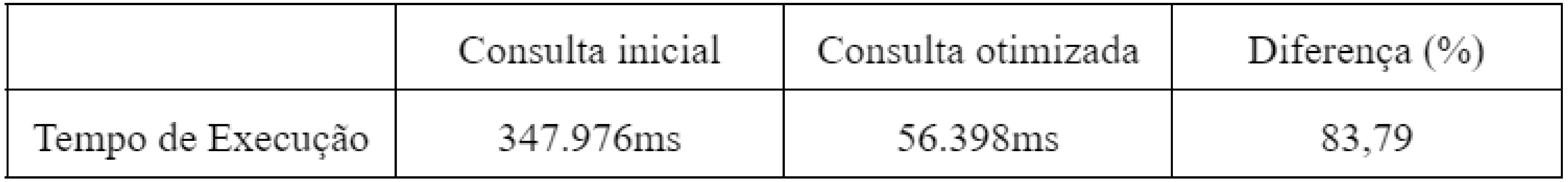


```
Nested Loop Anti Join  (cost=3.74..1673.52 rows=1 width=94) (actual time=1.516..56.356 rows=56 loops=1)
* -> Hash Join  (cost=3.31..235.24 rows=44 width=94) (actual time=0.201..1.535 rows=76 loops=1)
      Hash Cond: (q.id_hotel = ho.id_hotel)
      -> Seq Scan on quarto q  (cost=0.00..229.50 rows=885 width=14) (actual time=0.024..1.343 rows=1633 loops=1)
            Filter: ((categoria = 'Suite dupla'::text) AND (capacidade = 4))
            Rows Removed by Filter: 8467
      -> Hash  (cost=3.25..3.25 rows=5 width=84) (actual time=0.020..0.021 rows=5 loops=1)
            Buckets: 1024  Batches: 1  Memory Usage: 9kB
            -> Seq Scan on hotel ho  (cost=0.00..3.25 rows=5 width=84) (actual time=0.004..0.019 rows=5 loops=1)
                  Filter: ((estado)::text = 'BA'::text)
                  Rows Removed by Filter: 95
   -> Index Scan using reserva_pkey on reserva r  (cost=0.42..305.49 rows=18 width=8) (actual time=0.719..0.719 rows=0 loops=76)
      Index Cond: ((numero_quarto = q.numero) AND (id_hotel = ho.id_hotel) AND (data_entrada <= '2023-02-02'::date))
      Filter: (data_saida > '2023-01-05'::date)
      Rows Removed by Filter: 76
Planning Time: 2.926 ms
Execution Time: 56.398 ms
Total rows: 17 of 17    Query complete 00:00:00.206
```

$\}$ QUARTOS COM A LOCALIZACÃO, CATEGORIA E CAPACIDADE DESEJADA

PARA OTIMIZAR MAIS AINDA:

```sql
CREATE INDEX index_reserva on reserva (numero_quarto, id_hotel, data_entrada, data_saida);
CREATE INDEX index_estado on quarto(categoria, capacidade);
```



	Consulta inicial	Consulta otimizada	Diferença (%)
Tempo de Execução	347.976ms	56.398ms	83,79

Tabela 2: Comparação entre a consulta 1 e sua otimização

CONSULTA 2

Inicial

```
explain analyze
select ho.nome, ho.endereco, r.id_hotel, r.data_entrada, r.data_saida
from hospede ho, reserva r
where ho.cpf_hospede = r.cpf_hospede
and ho.nome ILIKE '%Almeida' and r.modo_pagamento = 'Credito' and r.ano_entrada >= 2014
```

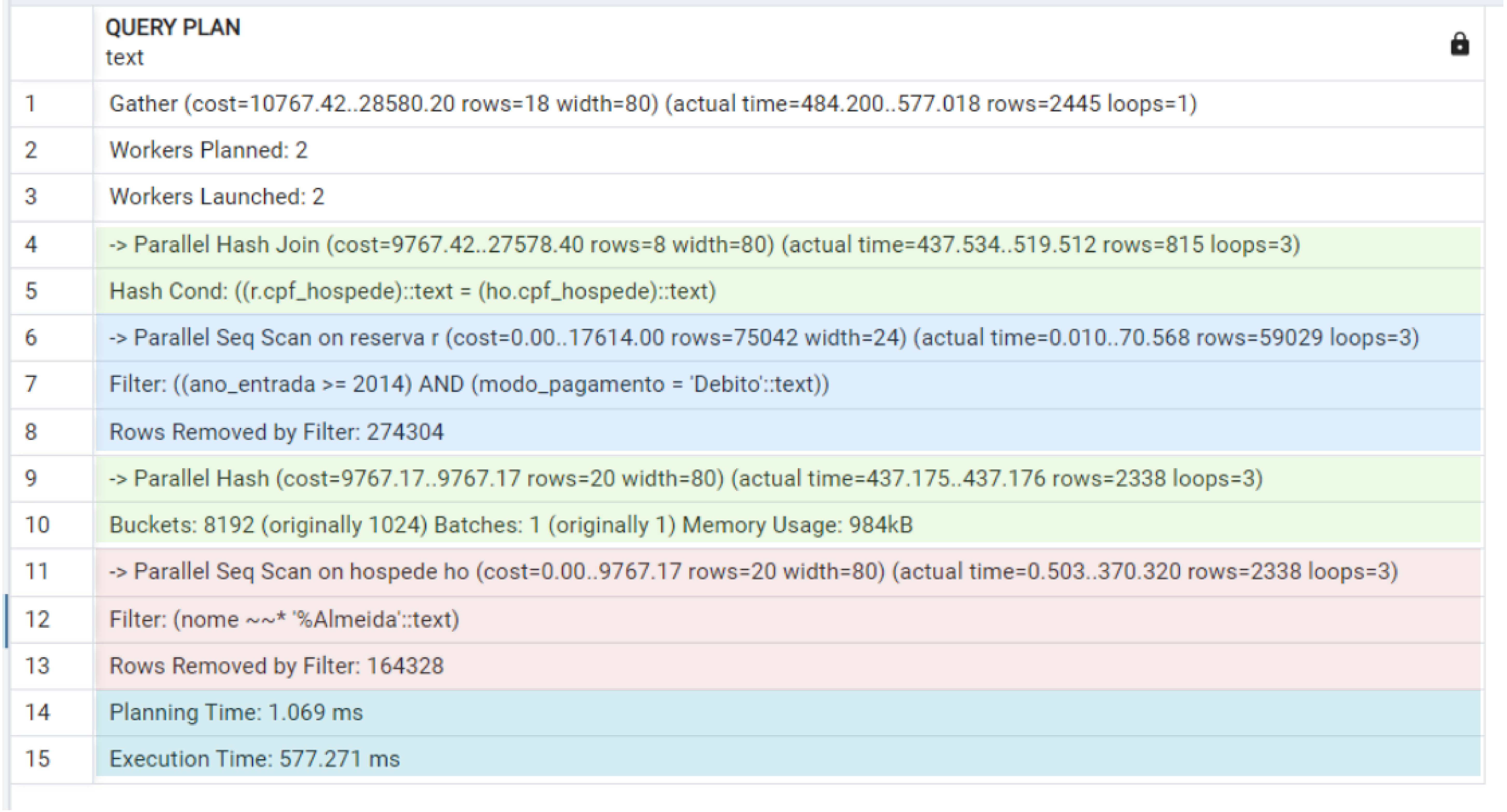
Produto cartesiano

não é case-sensitive

Expressão regular

atributos candidatos

Plano de Consulta



	QUERY PLAN text
1	Gather (cost=10767.42..28580.20 rows=18 width=80) (actual time=484.200..577.018 rows=2445 loops=1)
2	Workers Planned: 2
3	Workers Launched: 2
4	-> Parallel Hash Join (cost=9767.42..27578.40 rows=8 width=80) (actual time=437.534..519.512 rows=815 loops=3)
5	Hash Cond: ((r.cpf_hospede)::text = (ho.cpf_hospede)::text)
6	-> Parallel Seq Scan on reserva r (cost=0.00..17614.00 rows=75042 width=24) (actual time=0.010..70.568 rows=59029 loops=3)
7	Filter: ((ano_entrada >= 2014) AND (modo_pagamento = 'Debito'::text))
8	Rows Removed by Filter: 274304
9	-> Parallel Hash (cost=9767.17..9767.17 rows=20 width=80) (actual time=437.175..437.176 rows=2338 loops=3)
10	Buckets: 8192 (originally 1024) Batches: 1 (originally 1) Memory Usage: 984kB
11	-> Parallel Seq Scan on hospede ho (cost=0.00..9767.17 rows=20 width=80) (actual time=0.503..370.320 rows=2338 loops=3)
12	Filter: (nome ~~* '%Almeida'::text)
13	Rows Removed by Filter: 164328
14	Planning Time: 1.069 ms
15	Execution Time: 577.271 ms

CONSULTA 2

Extensão para utilizar o índice do tipo GIN

```
SELECT * FROM pg_extension WHERE extname = 'pg_trgm';
CREATE EXTENSION pg_trgm;
```

Otimizada

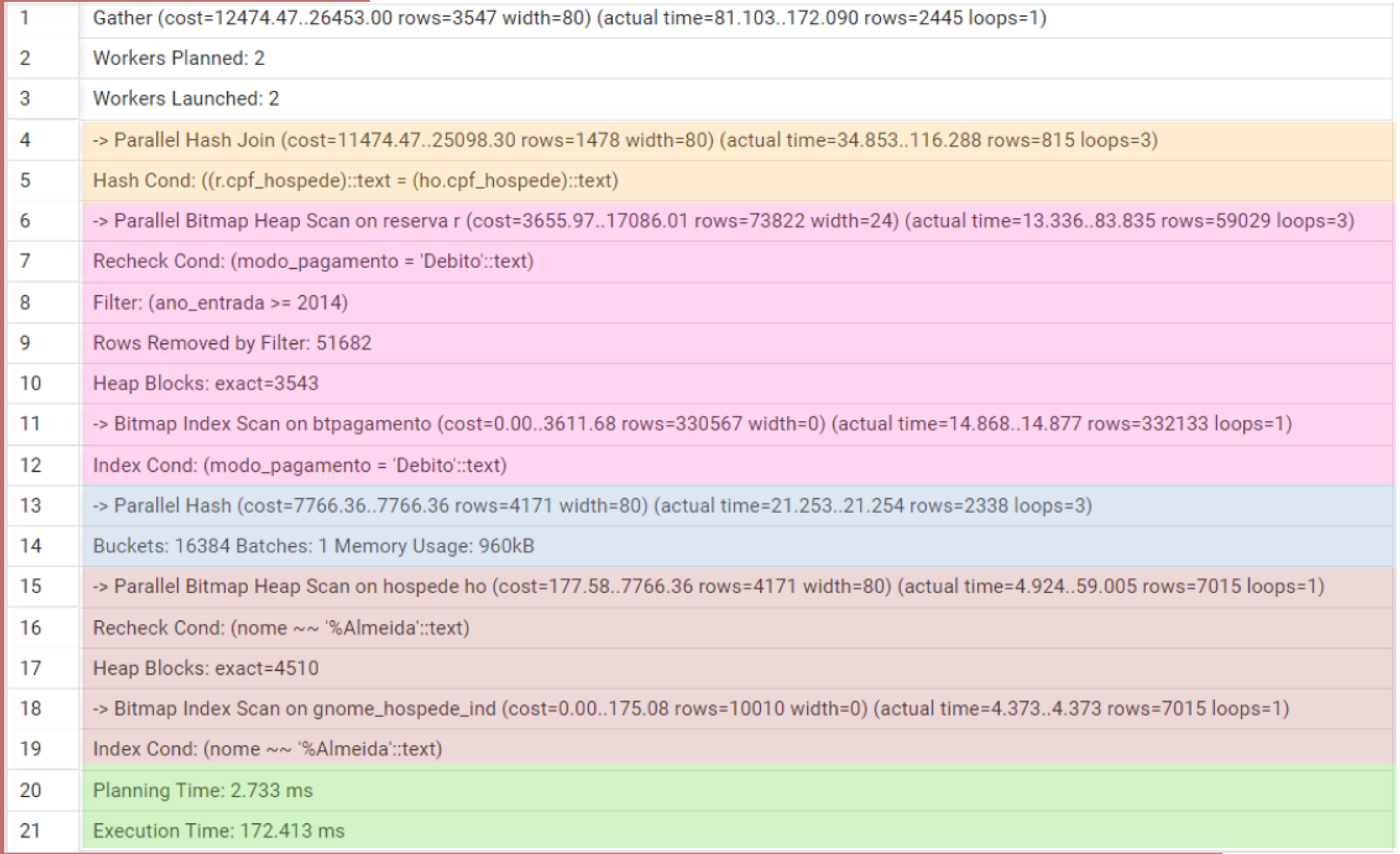
```
explain analyze
select ho.nome, ho.endereco, r.id_hotel, r.data_entrada, r.data_saida
from hospede ho natural join reserva r
where ho.nome LIKE '%Almeida' and r.modo_pagamento = 'Credito' and r.ano_entrada >= 2014
```

```
CREATE INDEX gnome_hospede_ind
ON hospede USING gin (nome gin_trgm_ops)
```

```
create index ano_entrada_ind
on reserva using btree (ano_entrada)
```

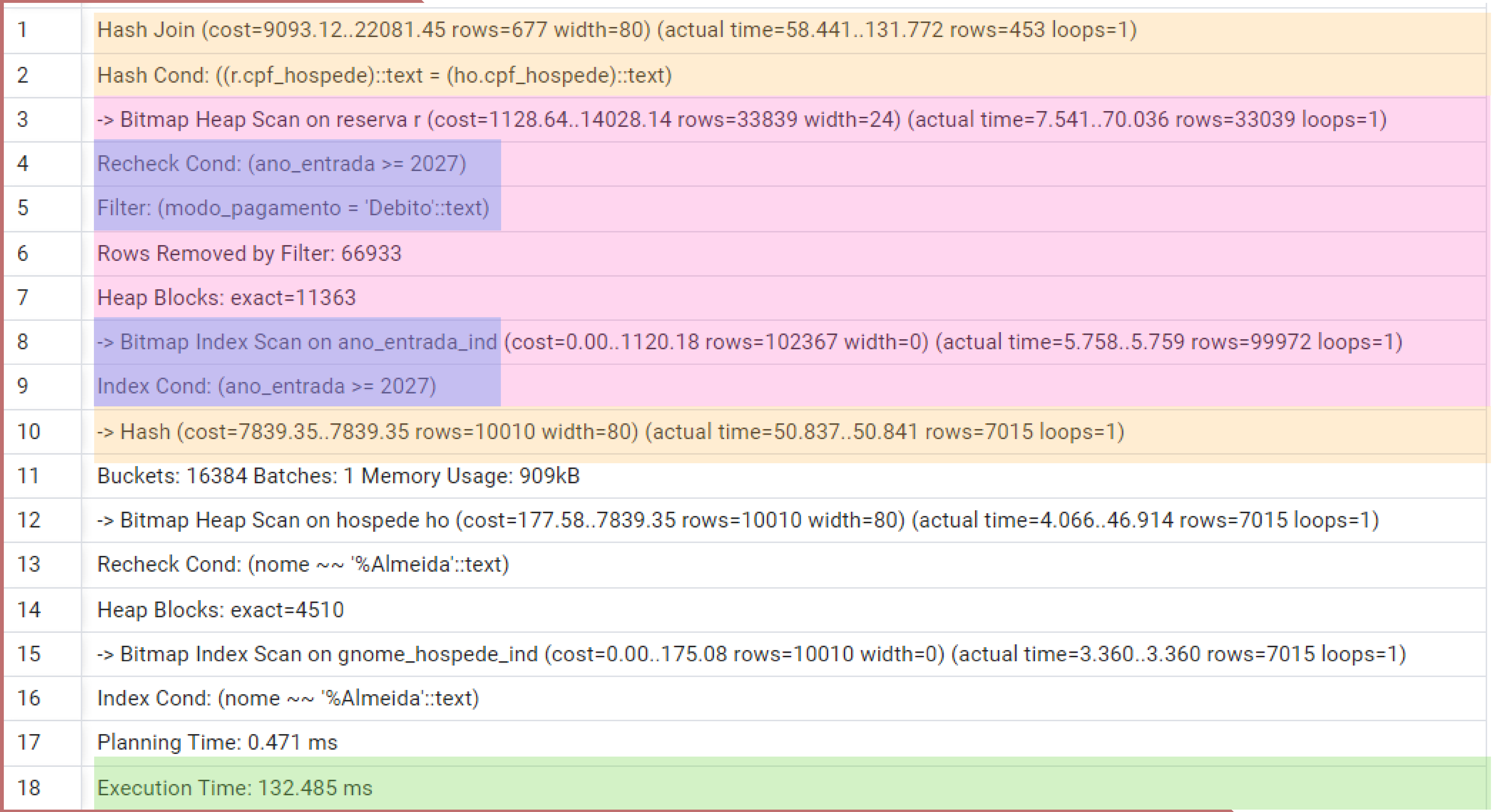
```
create index btpagamento on reserva(modo_pagamento)
```

Plano de Consulta

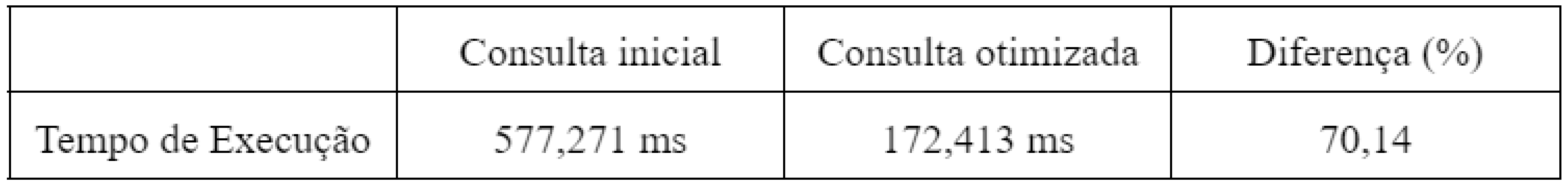


```
Gather (cost=12474.47..26453.00 rows=3547 width=80) (actual time=81.103..172.090 rows=2445 loops=1)
Workers Planned: 2
Workers Launched: 2
-> Parallel Hash Join (cost=11474.47..25098.30 rows=1478 width=80) (actual time=34.853..116.288 rows=815 loops=3)
Hash Cond: ((r.cpf_hospede)::text = (ho.cpf_hospede)::text)
-> Parallel Bitmap Heap Scan on reserva r (cost=3655.97..17086.01 rows=73822 width=24) (actual time=13.336..83.835 rows=59029 loops=3)
Recheck Cond: (modo_pagamento = 'Debito'::text)
Filter: (ano_entrada >= 2014)
Rows Removed by Filter: 51682
Heap Blocks: exact=3543
-> Bitmap Index Scan on btpagamento (cost=0.00..3611.68 rows=330567 width=0) (actual time=14.868..14.877 rows=332133 loops=1)
Index Cond: (modo_pagamento = 'Debito'::text)
-> Parallel Hash (cost=7766.36..7766.36 rows=4171 width=80) (actual time=21.253..21.254 rows=2338 loops=3)
Buckets: 16384 Batches: 1 Memory Usage: 960kB
-> Parallel Bitmap Heap Scan on hospede ho (cost=177.58..7766.36 rows=4171 width=80) (actual time=4.924..59.005 rows=7015 loops=1)
Recheck Cond: (nome ~~ '%Almeida'::text)
Heap Blocks: exact=4510
-> Bitmap Index Scan on gnome_hospede_ind (cost=0.00..175.08 rows=10010 width=0) (actual time=4.373..4.373 rows=7015 loops=1)
Index Cond: (nome ~~ '%Almeida'::text)
Planning Time: 2.733 ms
Execution Time: 172.413 ms
```

Plano de Consulta



```
Hash Join (cost=9093.12..22081.45 rows=677 width=80) (actual time=58.441..131.772 rows=453 loops=1)
Hash Cond: ((r.cpf_hospede)::text = (ho.cpf_hospede)::text)
-> Bitmap Heap Scan on reserva r (cost=1128.64..14028.14 rows=33839 width=24) (actual time=7.541..70.036 rows=33039 loops=1)
Recheck Cond: (ano_entrada >= 2027)
Filter: (modo_pagamento = 'Debito'::text)
Rows Removed by Filter: 66933
Heap Blocks: exact=11363
-> Bitmap Index Scan on ano_entrada_ind (cost=0.00..1120.18 rows=102367 width=0) (actual time=5.758..5.759 rows=99972 loops=1)
Index Cond: (ano_entrada >= 2027)
-> Hash (cost=7839.35..7839.35 rows=10010 width=80) (actual time=50.837..50.841 rows=7015 loops=1)
Buckets: 16384 Batches: 1 Memory Usage: 909kB
-> Bitmap Heap Scan on hospede ho (cost=177.58..7839.35 rows=10010 width=80) (actual time=4.066..46.914 rows=7015 loops=1)
Recheck Cond: (nome ~~ '%Almeida'::text)
Heap Blocks: exact=4510
-> Bitmap Index Scan on gnome_hospede_ind (cost=0.00..175.08 rows=10010 width=0) (actual time=3.360..3.360 rows=7015 loops=1)
Index Cond: (nome ~~ '%Almeida'::text)
Planning Time: 0.471 ms
Execution Time: 132.485 ms
```

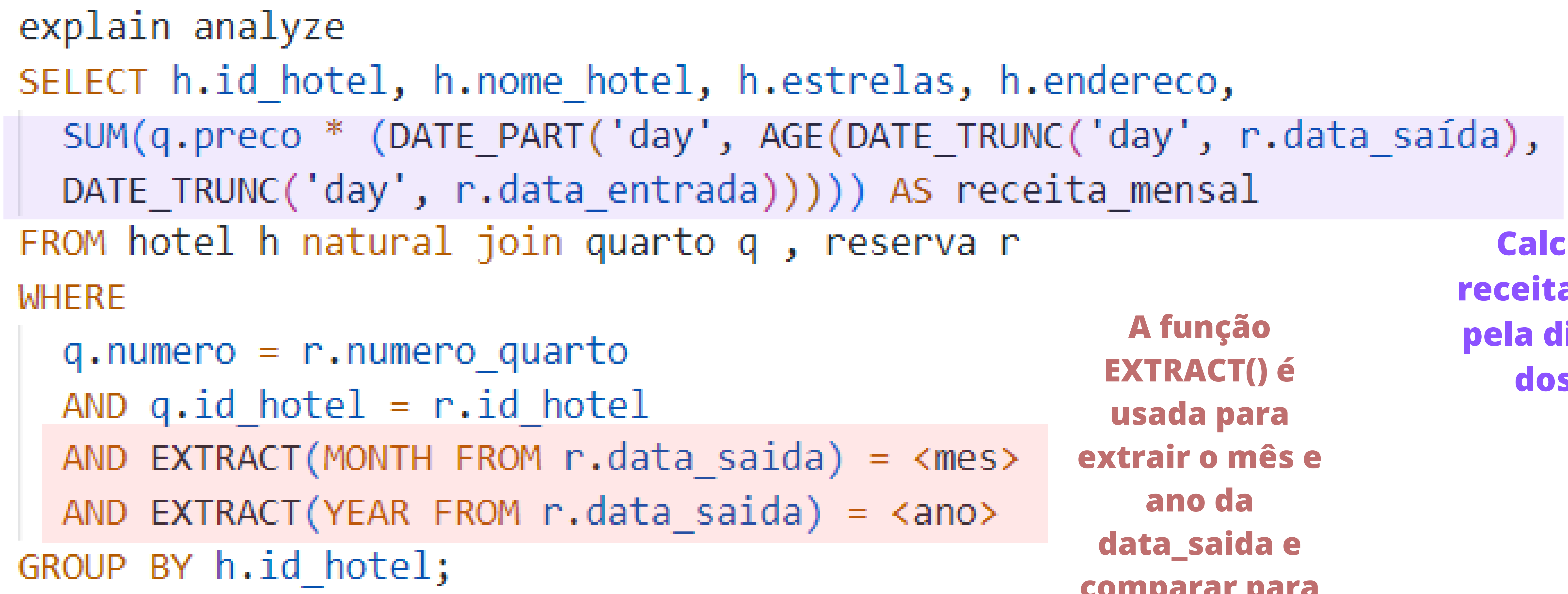


	Consulta inicial	Consulta otimizada	Diferença (%)
Tempo de Execução	577,271 ms	172,413 ms	70,14

Tabela 4: Comparação entre a consulta 2 e sua otimização

CONSULTA 3

Inicial

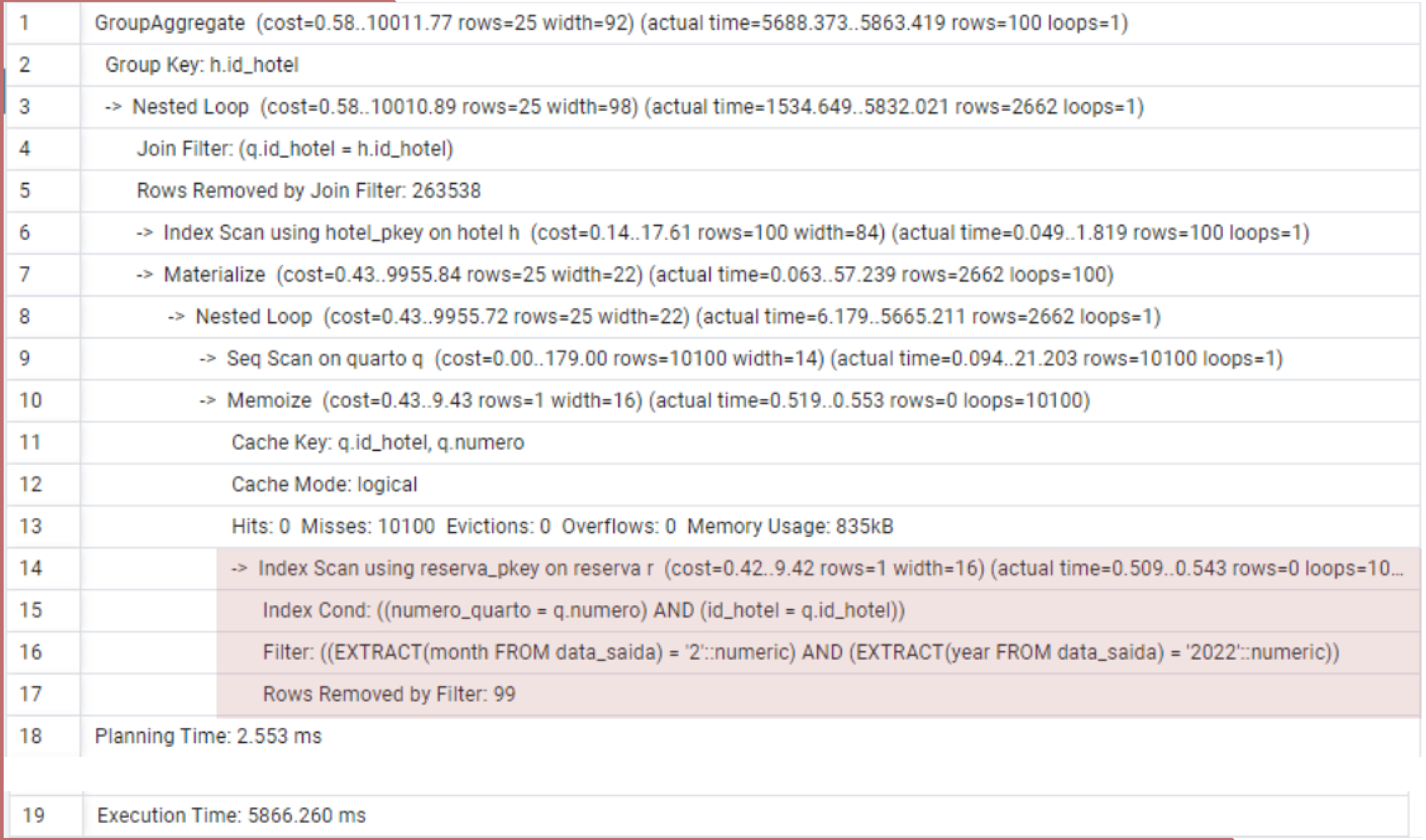


```sql
explain analyze
SELECT h.id_hotel, h.nome_hotel, h.estrelas, h.endereco,
  SUM(q.preco * (DATE_PART('day', AGE(DATE_TRUNC('day', r.data_saída),
  DATE_TRUNC('day', r.data_entrada))))) AS receita_mensal
FROM hotel h natural join quarto q , reserva r
WHERE
  q.numero = r.numero_quarto
  AND q.id_hotel = r.id_hotel
  AND EXTRACT(MONTH FROM r.data_saida) = <mes>
  AND EXTRACT(YEAR FROM r.data_saida) = <ano>
GROUP BY h.id_hotel;
```

Calculo da receita mensal pela diferença dos dias

A função EXTRACT() é usada para extrair o mês e ano da data_saida e comparar para filtrar apenas o mês e ano escolhidos

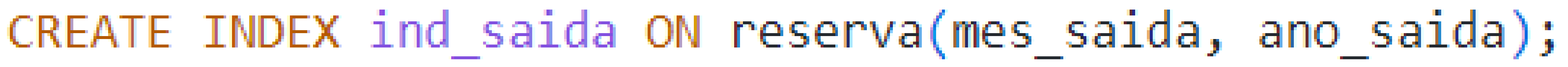
Plano de Consulta



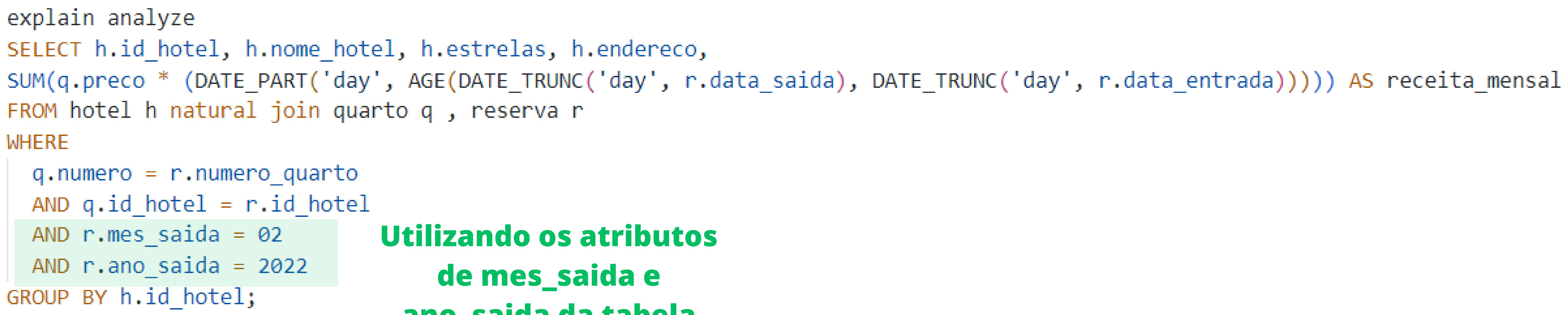
```
GroupAggregate  (cost=0.58..10011.77 rows=25 width=92) (actual time=5688.373..5863.419 rows=100 loops=1)
  Group Key: h.id_hotel
  ->  Nested Loop  (cost=0.58..10010.89 rows=25 width=98) (actual time=1534.649..5832.021 rows=2662 loops=1)
        Join Filter: (q.id_hotel = h.id_hotel)
        Rows Removed by Join Filter: 263538
        ->  Index Scan using hotel_pkey on hotel h  (cost=0.14..17.61 rows=100 width=84) (actual time=0.049..1.819 rows=100 loops=1)
        ->  Materialize  (cost=0.43..9955.84 rows=25 width=22) (actual time=0.063..57.239 rows=2662 loops=100)
              ->  Nested Loop  (cost=0.43..9955.72 rows=25 width=22) (actual time=6.179..5665.211 rows=2662 loops=1)
                    ->  Seq Scan on quarto q  (cost=0.00..179.00 rows=10100 width=14) (actual time=0.094..21.203 rows=10100 loops=1)
                    ->  Memoize  (cost=0.43..9.43 rows=1 width=16) (actual time=0.519..0.553 rows=0 loops=10100)
                          Cache Key: q.id_hotel, q.numero
                          Cache Mode: logical
                          Hits: 0  Misses: 10100  Evictions: 0  Overflows: 0  Memory Usage: 835kB
                          ->  Index Scan using reserva_pkey on reserva r  (cost=0.42..9.42 rows=1 width=16) (actual time=0.509..0.543 rows=0 loops=10...
                                Index Cond: ((numero_quarto = q.numero) AND (id_hotel = q.id_hotel))
                                Filter: ((EXTRACT(month FROM data_saida) = '2'::numeric) AND (EXTRACT(year FROM data_saida) = '2022'::numeric))
                                Rows Removed by Filter: 99
Planning Time: 2.553 ms

Execution Time: 5866.260 ms
```

CONSULTA 3

Otimizada



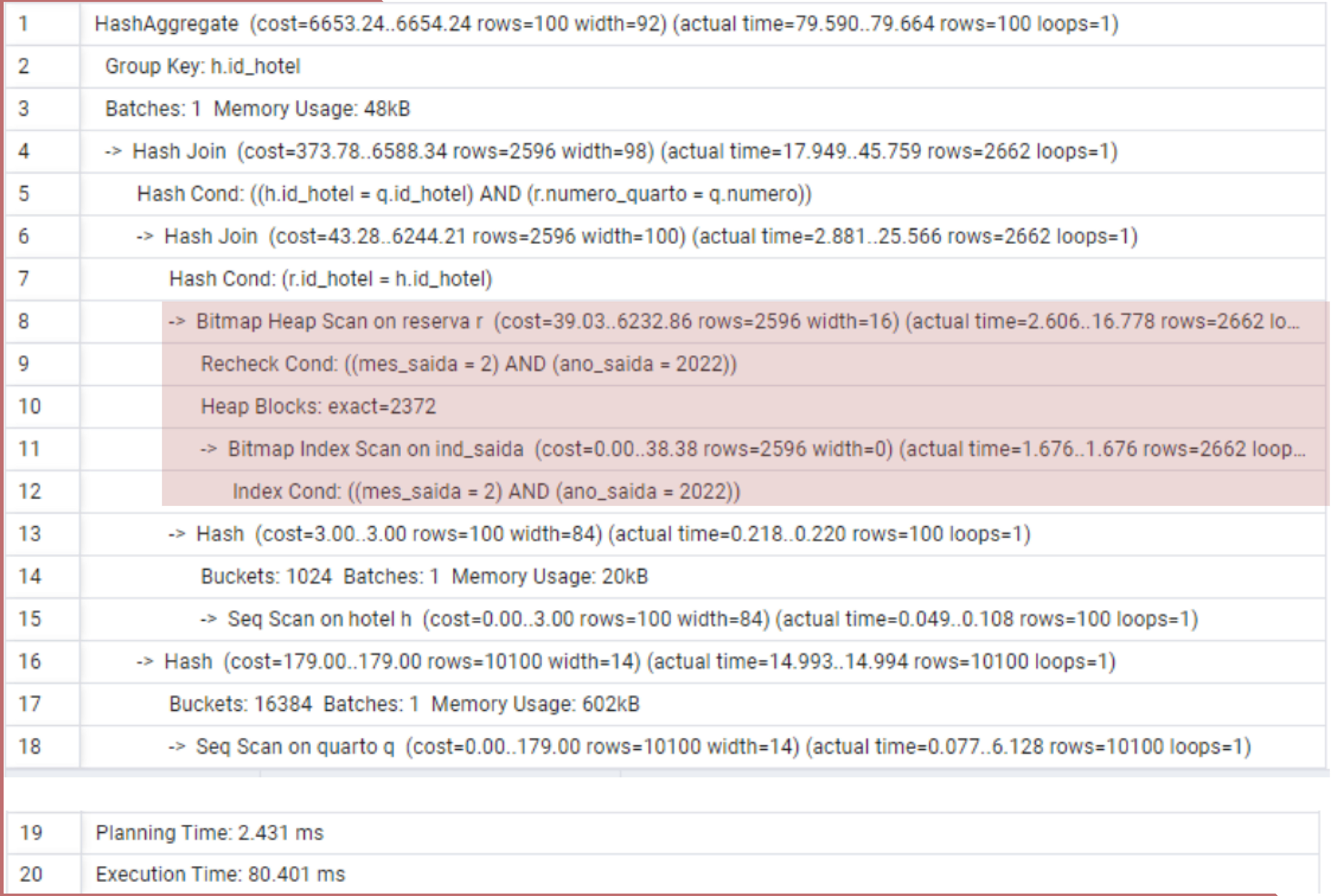
```
CREATE INDEX ind_saida ON reserva(mes_saida, ano_saida);
```



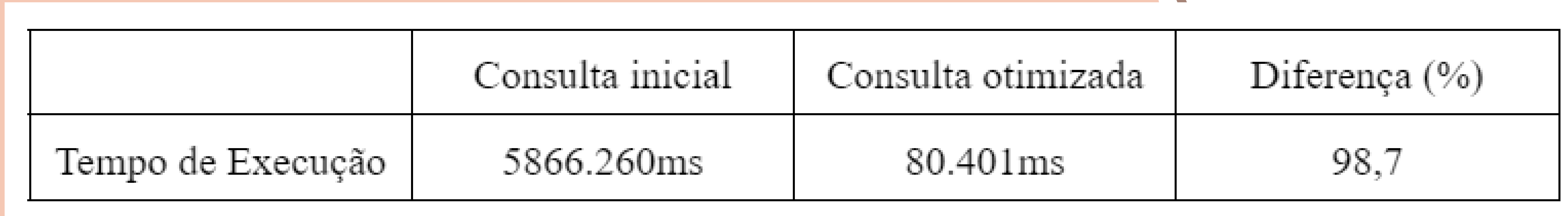
```
explain analyze
SELECT h.id_hotel, h.nome_hotel, h.estrelas, h.endereco,
SUM(q.preco * (DATE_PART('day', AGE(DATE_TRUNC('day', r.data_saida), DATE_TRUNC('day', r.data_entrada))))) AS receita_mensal
FROM hotel h natural join quarto q , reserva r
WHERE
    q.numero = r.numero_quarto
    AND q.id_hotel = r.id_hotel
    AND r.mes_saida = 02
    AND r.ano_saida = 2022
GROUP BY h.id_hotel;
```

Utilizando os atributos de mes_saida e ano_saida da tabela reserva

Plano de Consulta



```
HashAggregate  (cost=6653.24..6654.24 rows=100 width=92) (actual time=79.590..79.664 rows=100 loops=1)
  Group Key: h.id_hotel
  Batches: 1  Memory Usage: 48kB
  ->  Hash Join  (cost=373.78..6588.34 rows=2596 width=98) (actual time=17.949..45.759 rows=2662 loops=1)
        Hash Cond: ((h.id_hotel = q.id_hotel) AND (r.numero_quarto = q.numero))
        ->  Hash Join  (cost=43.28..6244.21 rows=2596 width=100) (actual time=2.881..25.566 rows=2662 loops=1)
              Hash Cond: (r.id_hotel = h.id_hotel)
              ->  Bitmap Heap Scan on reserva r  (cost=39.03..6232.86 rows=2596 width=16) (actual time=2.606..16.778 rows=2662 lo...
                    Recheck Cond: ((mes_saida = 2) AND (ano_saida = 2022))
                    Heap Blocks: exact=2372
                    ->  Bitmap Index Scan on ind_saida  (cost=0.00..38.38 rows=2596 width=0) (actual time=1.676..1.676 rows=2662 loop...
                          Index Cond: ((mes_saida = 2) AND (ano_saida = 2022))
              ->  Hash  (cost=3.00..3.00 rows=100 width=84) (actual time=0.218..0.220 rows=100 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 20kB
                    ->  Seq Scan on hotel h  (cost=0.00..3.00 rows=100 width=84) (actual time=0.049..0.108 rows=100 loops=1)
        ->  Hash  (cost=179.00..179.00 rows=10100 width=14) (actual time=14.993..14.994 rows=10100 loops=1)
              Buckets: 16384  Batches: 1  Memory Usage: 602kB
              ->  Seq Scan on quarto q  (cost=0.00..179.00 rows=10100 width=14) (actual time=0.077..6.128 rows=10100 loops=1)

Planning Time: 2.431 ms
Execution Time: 80.401 ms
```



	Consulta inicial	Consulta otimizada	Diferença (%)
Tempo de Execução	5866.260ms	80.401ms	98,7

Tabela 6: Comparação entre a consulta 3 e sua otimização

Obrigado!