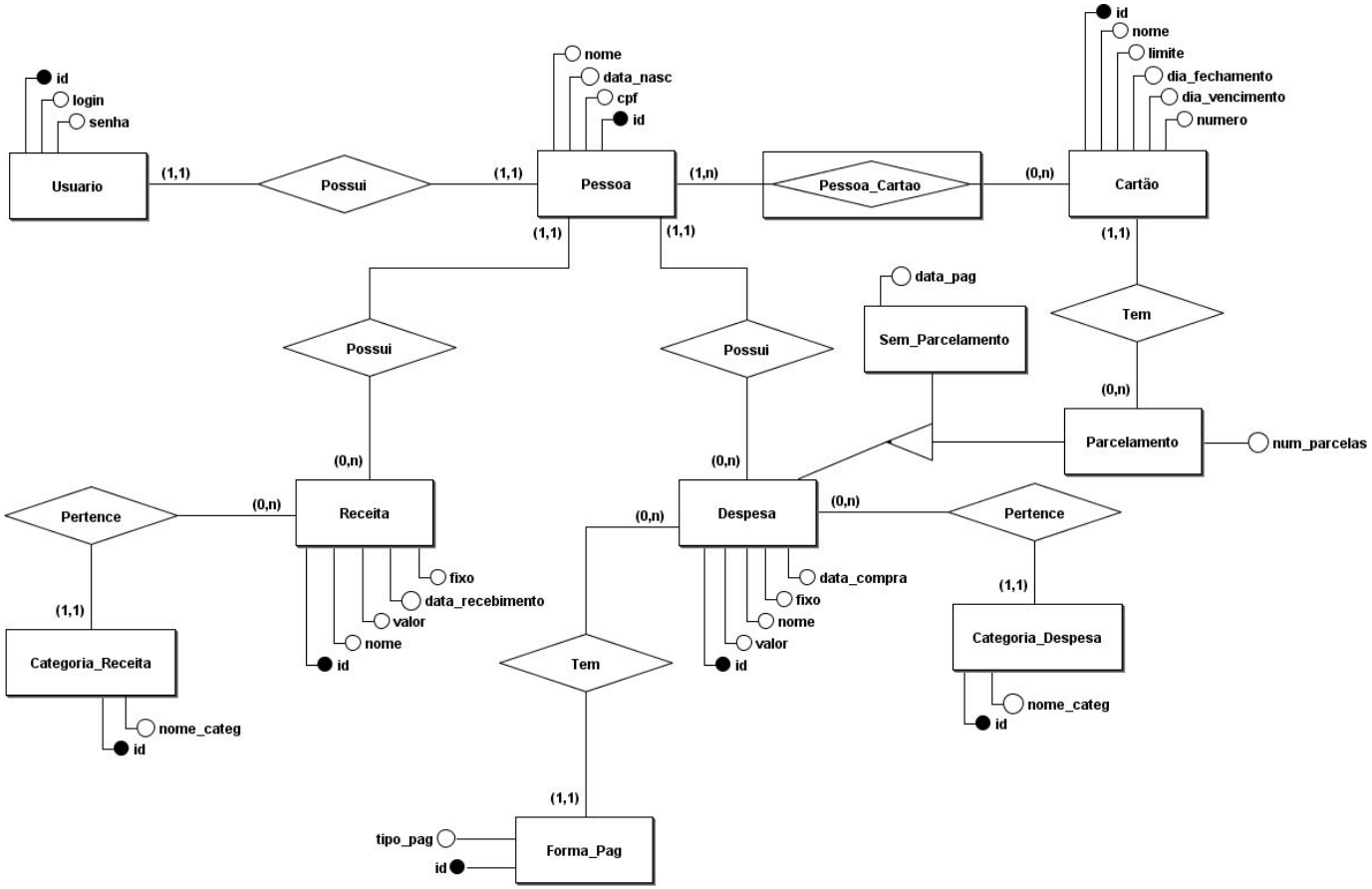
Ctrl+Money

Alunos: Brendon Mauro
Jennifer de Castro
Joel Will
Larissa Motta

Modelo conceitual



Performances

Todos os testes foram realizados no PG Admin 3 v 1.22.2

Especificações do computador

(Notebook Acer Aspire)

- Nome do Sistema Operacional: Microsoft Windows 10 Home Single Language
- Versão: 10.0.17134 Compilação 17134
- Modelo do sistema: Aspire E1-572
- Processador:Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz, 2301 Mhz, 2 Núcleo(s), 4 Processador(es) Lógico(s)
- Memória Física (RAM) Instalada 8,00 GB

- Informações do SSD
- Modelo: KINGSTON SA400S37240G
- Tamanho 223,57 GB (240.054.796.800 bytes)
- Interface: SATA Rev. 3.0 (6Gb/s) compatível com a versão anterior SATA Rev. 2.0 (3Gb/s)
- Leituras: ATÉ - 550MBs
- Gravações: ATÉ - 490MBs

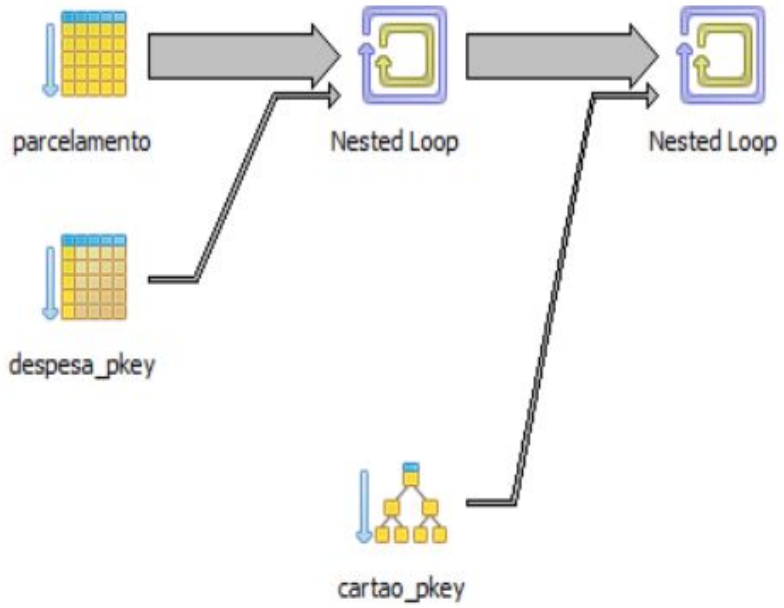
Testes com índices B-tree

Consulta de despesas de um cartão por um determinado mês:

Query 1

```
SELECT d.* as qtd_d FROM despesa d
JOIN parcelamento p ON (p.fk_despesa = d.id)
JOIN cartao c ON (p.fk_cartao = c.id)
WHERE c.id =  85107 AND d.data_compra > '2018-11-01' AND d.data_compra < '2018-11-30'
```

Sem Index


parcelamento
despesa_pkey
Nested Loop
Nested Loop
cartao_pkey

Detalhamento do Explain

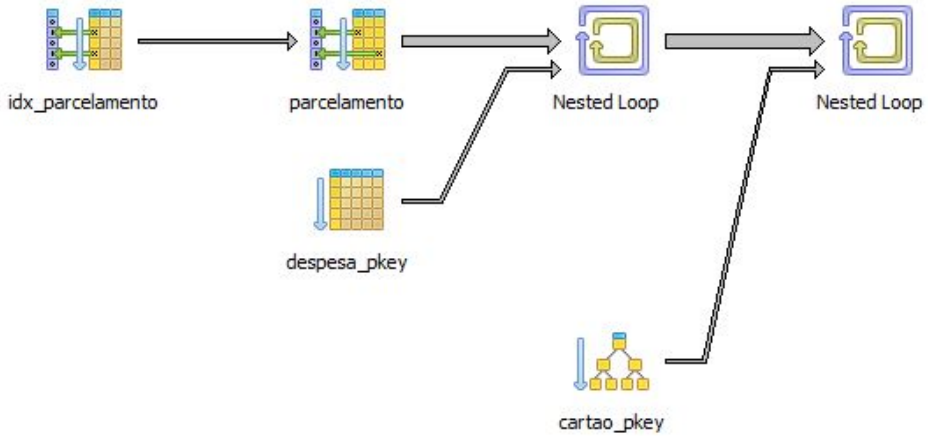
Sem index

```
Nested Loop  (cost=0.85..21571.59 rows=1 width=47) (actual
time=95.962..337.982 rows=5 loops=1)
  Output: d.valor, d.data_compra, d.fixo, d.id, d.nome,
d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag
  ->  Nested Loop  (cost=0.43..21563.14 rows=1 width=51) (actual
time=94.725..336.699 rows=5 loops=1)
        Output: d.valor, d.data_compra, d.fixo, d.id, d.nome,
d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag,
p.fk_cartao
        ->  Seq Scan on public.parcelamento p  (cost=0.00..21487.00
rows=9 width=8) (actual time=80.458..315.858 rows=14 loops=1)
              Output: p.num_parcelas, p.fk_despesa, p.fk_cartao
              Filter: (p.fk_cartao = 85107)
              Rows Removed by Filter: 1199986
        ->  Index Scan using despesa_pkey on public.despesa d
(cost=0.43..8.45 rows=1 width=47) (actual time=1.484..1.484 rows=0
loops=14)
              Output: d.valor, d.data_compra, d.fixo, d.id, d.nome,
d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag
              Index Cond: (d.id = p.fk_despesa)
              Filter: ((d.data_compra > '2018-11-01'::date) AND
(d.data_compra < '2018-11-30'::date))
              Rows Removed by Filter: 1
  ->  Index Only Scan using cartao_pkey on public.cartao c
(cost=0.42..8.44 rows=1 width=4) (actual time=0.253..0.254 rows=1
loops=5)
        Output: c.id
        Index Cond: (c.id = 85107)
        Heap Fetches: 5
Planning time: 42.614 ms
Execution time: 338.262 ms
```

Criação do index

```
CREATE INDEX idx_parcelamento ON parcelamento(fk_cartao);
```

Com Index


idx_parcelamento
parcelamento
Nested Loop
Nested Loop
despesa_pkey
cartao_pkey

Detalhamento do Explain

Com index

```
Nested Loop  (cost=5.34..124.19 rows=1 width=47) (actual
time=0.210..0.659 rows=5 loops=1)
  ->  Nested Loop  (cost=4.92..115.74 rows=1 width=51)
(actual time=0.200..0.639 rows=5 loops=1)
        ->  Bitmap Heap Scan on parcelamento p
(cost=4.50..39.60 rows=9 width=8) (actual time=0.154..0.541
rows=14 loops=1)
              Recheck Cond: (fk_cartao = 85107)
              Heap Blocks: exact=14
              ->  Bitmap Index Scan on idx_parcelamento
(cost=0.00..4.50 rows=9 width=0) (actual time=0.090..0.090
rows=14 loops=1)
                    Index Cond: (fk_cartao = 85107)
        ->  Index Scan using despesa_pkey on despesa d
(cost=0.43..8.45 rows=1 width=47) (actual time=0.006..0.006
rows=0 loops=14)
              Index Cond: (id = p.fk_despesa)
              Filter: ((data_compra > '2018-11-01'::date) AND
(data_compra < '2018-11-30'::date))
              Rows Removed by Filter: 1
  ->  Index Only Scan using cartao_pkey on cartao c
(cost=0.42..8.44 rows=1 width=4) (actual time=0.003..0.003
rows=1 loops=5)
        Index Cond: (id = 85107)
        Heap Fetches: 5
Planning time: 1.092 ms
Execution time: 0.731 ms
```

Testes de Performance

Sem index

Nº de vezes	Planning time(ms)	Execution time (ms)
1	0.543	191.964
2	0.679	265.687
3	0.815	90.497
4	0.353	190.347
5	0.369	192.308
6	0.369	187.168
7	0.345	191.624
Média	0.459 ms	191.348 ms

Com index

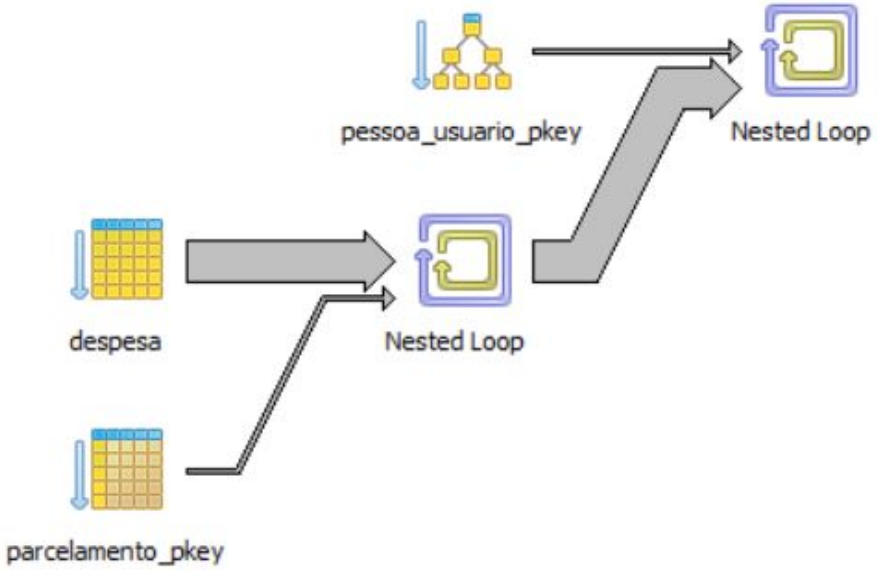
Nº de vezes	Planning time(ms)	Execution time (ms)
1	1.092	0.731
2	0.457	0.191
3	0.632	0.141
4	0.368	0.153
5	0.371	0.152
6	0.656	0.141
7	0.365	0.152
Média	0.497 ms	0.158 ms

Consulta de despesas de despesas parceladas de um determinado mês:

Query 2

```
SELECT parcelamento.*,despesa.* FROM despesa
INNER JOIN pessoa_usuario on (pessoa_usuario.id = despesa.fk_pessoa_usuario)
INNER JOIN parcelamento on (parcelamento.fk_despesa = despesa.id)
WHERE pessoa_usuario.id = 2291 AND despesa.data_compra > '2018-11-01' AND despesa.data_compra < '2018-11-30';
```

Sem Index


pessoa_usuario_pkey
Nested Loop
despesa
Nested Loop
parcelamento_pkey

Detalhamento do Explain

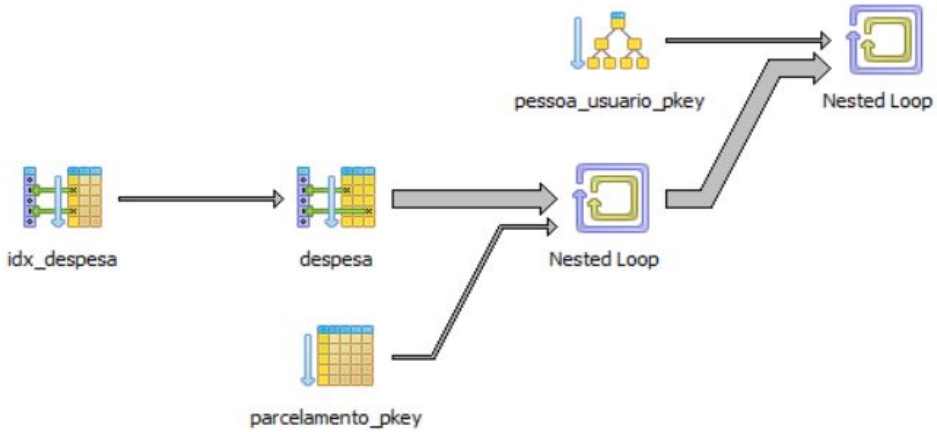
Sem index

```
Data output:
Nested Loop  (cost=0.71..41471.08 rows=5 width=59) (actual
time=2.926..247.296 rows=11 loops=1)
  ->  Index Only Scan using pessoa_usuario_pkey on
pessoa_usuario  (cost=0.29..8.30 rows=1 width=4) (actual
time=0.089..0.091 rows=1 loops=1)
        Index Cond: (id = 2291)
        Heap Fetches: 1
  ->  Nested Loop  (cost=0.43..41462.73 rows=5 width=59)
(actual time=2.835..247.197 rows=11 loops=1)
        ->  Seq Scan on despesa  (cost=0.00..41412.00
rows=6 width=47) (actual time=2.088..238.969 rows=13
loops=1)
              Filter: ((data_compra > '2018-11-01'::date)
AND (data_compra < '2018-11-30'::date) AND
(fk_pessoa_usuario = 2291))
              Rows Removed by Filter: 1499987
        ->  Index Scan using parcelamento_pkey on
parcelamento  (cost=0.43..8.45 rows=1 width=12) (actual
time=0.626..0.627 rows=1 loops=13)
              Index Cond: (fk_despesa = despesa.id)
Planning time: 0.782 ms
Execution time: 247.345 ms
```

Criação do index

```
CREATE INDEX idx_despesa ON despesa(fk_pessoa_usuario);
```

Com Index


pessoa_usuario_pkey
Nested Loop
idx_despesa
despesa
Nested Loop
parcelamento_pkey

Detalhamento do Explain

Com index

```
Data output:

Nested Loop  (cost=6.24..611.76 rows=5 width=59) (actual
time=0.188..1.959 rows=11 loops=1)
  ->  Index Only Scan using pessoa_usuario_pkey on
pessoa_usuario  (cost=0.29..8.30 rows=1 width=4) (actual
time=0.007..0.007 rows=1 loops=1)
        Index Cond: (id = 2291)
        Heap Fetches: 1
  ->  Nested Loop  (cost=5.96..603.41 rows=5 width=59)
(actual time=0.172..1.941 rows=11 loops=1)
        ->  Bitmap Heap Scan on despesa  (cost=5.53..552.68
rows=6 width=47) (actual time=0.165..1.855 rows=13 loops=1)
              Recheck Cond: (fk_pessoa_usuario = 2291)
              Filter: ((data_compra > '2018-11-01'::date) AND
(data_compra < '2018-11-30'::date))
              Rows Removed by Filter: 156
              Heap Blocks: exact=169
              ->  Bitmap Index Scan on idx_despesa
(cost=0.00..5.53 rows=147 width=0) (actual time=0.078..0.078
rows=169 loops=1)
                    Index Cond: (fk_pessoa_usuario = 2291)
        ->  Index Scan using parcelamento_pkey on
parcelamento  (cost=0.43..8.45 rows=1 width=12) (actual
time=0.005..0.006 rows=1 loops=13)
              Index Cond: (fk_despesa = despesa.id)
Planning time: 0.987 ms
Execution time: 2.011 ms
```

Testes de Performance

Sem index

Nº de vezes	Planning time(ms)	Execution time (ms)
1	0.782	247.345
2	0.288	239.091
3	0.304	241.255
4	0.327	279.761
5	0.288	237.932
6	0.291	231.078
7	0.288	233.426
Média	0.299 ms	239.809 ms

Com index

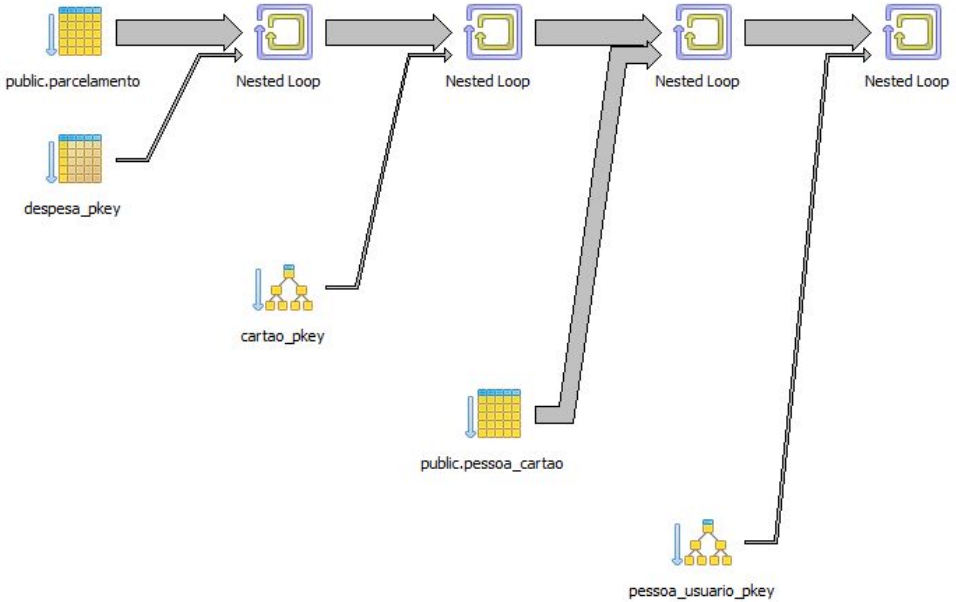
Nº de vezes	Planning time(ms)	Execution time (ms)
1	0.987	2.011
2	.386	0.596
3	0.326	0.334
4	0.620	0.320
5	0.306	0.311
6	0.320	0.344
7	0.323	0.344
Média	0.395 ms	0.388 ms

Consulta de despesas de um cartão e de uma determinada pessoa por um determinado mês:

Query 3

```
SELECT d.* FROM despesa d
JOIN parcelamento p ON (p.fk_despesa = d.id)
JOIN cartao c ON (c.id = p.fk_cartao)
JOIN pessoa_cartao pc ON (c.id = pc.fk_cartao)
JOIN pessoa_usuario pu ON (pu.id = pc.fk_pessoa_usuario)
WHERE pc.fk_pessoa_usuario = 2291 AND pc.fk_cartao = 19247 AND d.data_compra > '2018-11-01'
AND d.data_compra < '2018-11-30';
```

Sem Index


public.parcelamento
despesa_pkey
Nested Loop
cartao_pkey
Nested Loop
public.pessoa_cartao
Nested Loop
pessoa_usuario_pkey
Nested Loop

Detalhamento do Explain

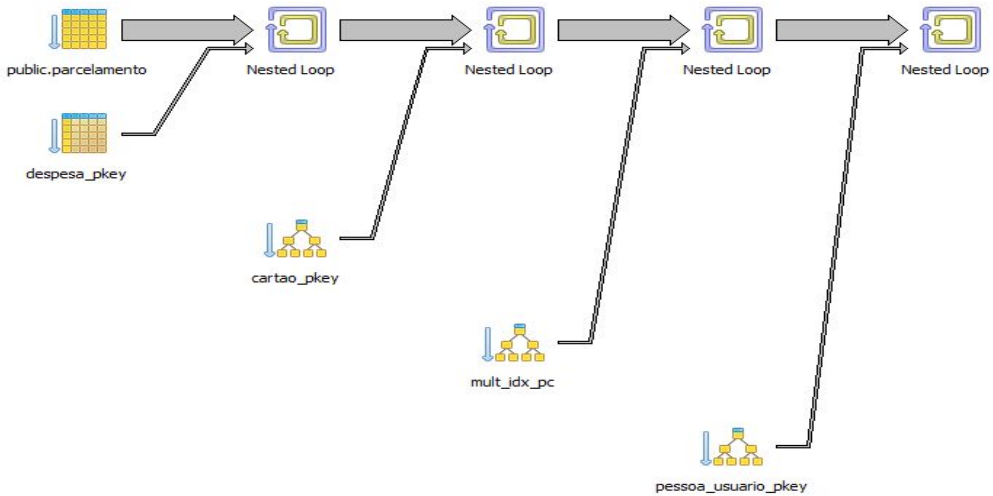
Sem index

```
Nested Loop  (cost=1.13..22551.91 rows=1 width=47) (actual time=91.409..187.305 rows=1 loops=1)
  Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa,
d.fk_forma_pag
  ->  Nested Loop  (cost=0.85..22543.60 rows=1 width=51) (actual time=91.399..187.294 rows=1 loops=1)
        Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa,
d.fk_forma_pag, pc.fk_pessoa_usuario
        ->  Nested Loop  (cost=0.85..21571.59 rows=1 width=51) (actual time=87.018..182.752 rows=1 loops=1)
              Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario,
d.fk_categoria_despesa, d.fk_forma_pag, c.id
              ->  Nested Loop  (cost=0.43..21563.14 rows=1 width=51) (actual time=87.009..182.742 rows=1
loops=1)
                    Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario,
d.fk_categoria_despesa, d.fk_forma_pag, p.fk_cartao
                    ->  Seq Scan on public.parcelamento p  (cost=0.00..21487.00 rows=9 width=8) (actual
time=6.646..182.611 rows=8 loops=1)
                          Output: p.num_parcelas, p.fk_despesa, p.fk_cartao
                          Filter: (p.fk_cartao = 19247)
                          Rows Removed by Filter: 1199992
                    ->  Index Scan using despesa_pkey on public.despesa d  (cost=0.43..8.45 rows=1
width=47) (actual time=0.012..0.012 rows=0 loops=8)
                          Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario,
d.fk_categoria_despesa, d.fk_forma_pag
                          Index Cond: (d.id = p.fk_despesa)
                          Filter: ((d.data_compra > '2018-11-01'::date) AND (d.data_compra <
'2018-11-30'::date))
                          Rows Removed by Filter: 1
              ->  Index Only Scan using cartao_pkey on public.cartao c  (cost=0.42..8.44 rows=1 width=4)
(actual time=0.008..0.008 rows=1 loops=1)
                    Output: c.id
                    Index Cond: (c.id = 19247)
                    Heap Fetches: 1
        ->  Seq Scan on public.pessoa_cartao pc  (cost=0.00..972.00 rows=1 width=8) (actual
time=4.380..4.540 rows=1 loops=1)
              Output: pc.fk_pessoa_usuario, pc.fk_cartao
              Filter: ((pc.fk_cartao = 19247) AND (pc.fk_pessoa_usuario = 2291))
              Rows Removed by Filter: 49999
  ->  Index Only Scan using pessoa_usuario_pkey on public.pessoa_usuario pu  (cost=0.29..8.30 rows=1
width=4) (actual time=0.008..0.008 rows=1 loops=1)
        Output: pu.id
        Index Cond: (pu.id = 2291)
        Heap Fetches: 1
Planning time: 0.840 ms
Execution time: 187.365 ms
```

Criação do index

```
CREATE INDEX mult_idx_pc ON pessoa_cartao (fk_cartao, fk_pessoa_usuario);
```

Com Index


public.parcelamento
despesa_pkey
Nested Loop
cartao_pkey
Nested Loop
mult_idx_pc
Nested Loop
pessoa_usuario_pkey
Nested Loop

Detalhamento do Explain

Com index

```
Nested Loop  (cost=1.42..21588.22 rows=1 width=47) (actual time=277.121..373.010 rows=1 loops=1)
  Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag
  ->  Nested Loop  (cost=1.14..21579.91 rows=1 width=51) (actual time=277.110..372.997 rows=1 loops=1)
        Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag,
pc.fk_pessoa_usuario
        ->  Nested Loop  (cost=0.85..21571.59 rows=1 width=51) (actual time=276.608..372.494 rows=1 loops=1)
              Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa,
d.fk_forma_pag, c.id
              ->  Nested Loop  (cost=0.43..21563.14 rows=1 width=51) (actual time=276.594..372.478 rows=1 loops=1)
                    Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa,
d.fk_forma_pag, p.fk_cartao
                    ->  Seq Scan on public.parcelamento p  (cost=0.00..21487.00 rows=9 width=8) (actual time=23.843..372.286
rows=8 loops=1)
                          Output: p.num_parcelas, p.fk_despesa, p.fk_cartao
                          Filter: (p.fk_cartao = 19247)
                          Rows Removed by Filter: 1199992
                    ->  Index Scan using despesa_pkey on public.despesa d  (cost=0.43..8.45 rows=1 width=47) (actual
time=0.018..0.018 rows=0 loops=8)
                          Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa,
d.fk_forma_pag
                          Index Cond: (d.id = p.fk_despesa)
                          Filter: ((d.data_compra > '2018-11-01'::date) AND (d.data_compra < '2018-11-30'::date))
                          Rows Removed by Filter: 1
              ->  Index Only Scan using cartao_pkey on public.cartao c  (cost=0.42..8.44 rows=1 width=4) (actual
time=0.012..0.013 rows=1 loops=1)
                    Output: c.id
                    Index Cond: (c.id = 19247)
                    Heap Fetches: 1
        ->  Index Only Scan using mult_idx_pc on public.pessoa_cartao pc  (cost=0.29..8.31 rows=1 width=8) (actual
time=0.500..0.500 rows=1 loops=1)
              Output: pc.fk_cartao, pc.fk_pessoa_usuario
              Index Cond: ((pc.fk_cartao = 19247) AND (pc.fk_pessoa_usuario = 2291))
              Heap Fetches: 1
  ->  Index Only Scan using pessoa_usuario_pkey on public.pessoa_usuario pu  (cost=0.29..8.30 rows=1
width=4) (actual time=0.008..0.010 rows=1 loops=1)
        Output: pu.id
        Index Cond: (pu.id = 2291)
        Heap Fetches: 1
Planning time: 3.549 ms
Execution time: 373.130 ms
```

Testes de Performance

Sem index

Nº de vezes	Planning time(ms)	Execution time (ms)
1	0.840	187.365
2	0.470	186.432
3	0.469	187.379
4	0.731	190.532
5	0.460	179.485
6	0.499	180.346
7	0.464	180.609
Média	0.527 ms	184.426 ms

Com index

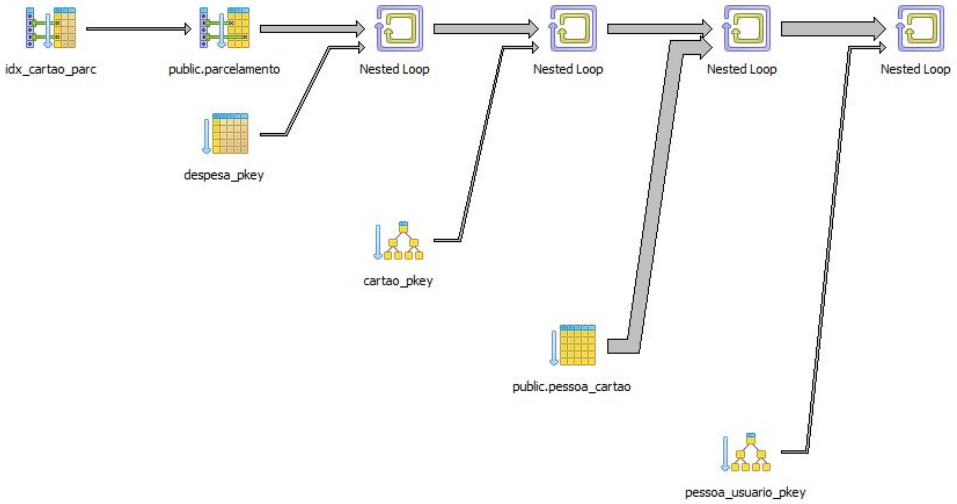
Nº de vezes	Planning time(ms)	Execution time (ms)
1	3.549	373.130
2	.1.596	360.373
3	1.686	383.483
4	1.318	377.022
5	1.322	406.688
6	1.324	236.815
7	1.332	266.099
Média	1.452 ms	353.021 ms

Utilizando a mesma consulta, mas com o mesmo índice utilizado na Query 1

Segundo teste Query 3

- Primeiro será utilizado o mesmo índice da Query 1, mas sem o índice de múltiplas colunas.
- Depois, será utilizado mesmo índice da Query 1, mas com o índice de múltiplas colunas.

Sem Index (multicolumn)



Detalhamento do Explain

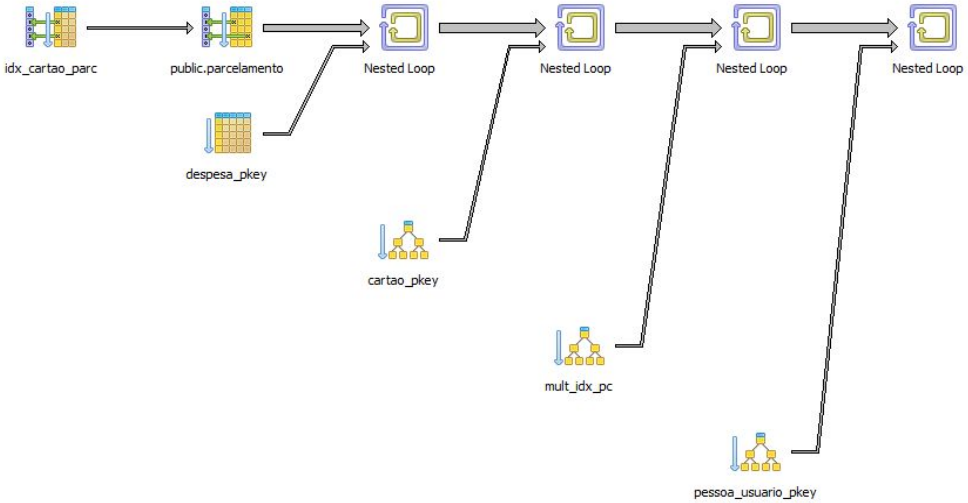
Sem index

```
Nested Loop  (cost=5.63..1104.51 rows=1 width=47) (actual time=4.536..4.743 rows=1 loops=1)
  Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag
  ->  Nested Loop  (cost=5.34..1096.20 rows=1 width=51) (actual time=4.528..4.734 rows=1 loops=1)
        Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag,
pc.fk_pessoa_usuario
        ->  Nested Loop  (cost=5.34..124.19 rows=1 width=51) (actual time=0.131..0.171 rows=1 loops=1)
              Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag, c.id
              ->  Nested Loop  (cost=4.92..115.74 rows=1 width=51) (actual time=0.123..0.163 rows=1 loops=1)
                    Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag, p.fk_cartao
                    ->  Bitmap Heap Scan on public.parcelamento p  (cost=4.50..39.60 rows=9 width=8) (actual time=0.098..0.105 rows=8 loops=1)
                          Output: p.num_parcelas, p.fk_despesa, p.fk_cartao
                          Recheck Cond: (p.fk_cartao = 19247)
                          Heap Blocks: exact=8
                          ->  Bitmap Index Scan on idx_cartao_parc  (cost=0.00..4.50 rows=9 width=0) (actual time=0.093..0.093 rows=8 loops=1)
                                Index Cond: (p.fk_cartao = 19247)
                    ->  Index Scan using despesa_pkey on public.despesa d  (cost=0.43..8.45 rows=1 width=47) (actual time=0.007..0.007 rows=0
loops=8)
                          Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag
                          Index Cond: (d.id = p.fk_despesa)
                          Filter: ((d.data_compra > '2018-11-01'::date) AND (d.data_compra < '2018-11-30'::date))
                          Rows Removed by Filter: 1
              ->  Index Only Scan using cartao_pkey on public.cartao c  (cost=0.42..8.44 rows=1 width=4) (actual time=0.008..0.008 rows=1
loops=1)
                    Output: c.id
                    Index Cond: (c.id = 19247)
                    Heap Fetches: 1
        ->  Seq Scan on public.pessoa_cartao pc  (cost=0.00..972.00 rows=1 width=8) (actual time=4.396..4.561 rows=1 loops=1)
              Output: pc.fk_pessoa_usuario, pc.fk_cartao
              Filter: ((pc.fk_cartao = 19247) AND (pc.fk_pessoa_usuario = 2291))
              Rows Removed by Filter: 49999
  ->  Index Only Scan using pessoa_usuario_pkey on public.pessoa_usuario pu  (cost=0.29..8.30 rows=1 width=4) (actual time=0.007..0.008
rows=1 loops=1)
        Output: pu.id
        Index Cond: (pu.id = 2291)
        Heap Fetches: 1
Planning time: 1.434 ms
Execution time: 4.832 ms
```

Criação do index (Multicolumn)

```
CREATE INDEX mult_idx_pc ON pessoa_cartao (fk_cartao, fk_pessoa_usuario);
```

Com Index (multicolumn)



Detalhamento do Explain

Com index

```
"Nested Loop  (cost=5.92..140.82 rows=1 width=47) (actual time=0.324..0.355 rows=1 loops=1)"
"  Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag"
"  ->  Nested Loop  (cost=5.63..132.51 rows=1 width=51) (actual time=0.317..0.348 rows=1 loops=1)"
"        Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag,
pc.fk_pessoa_usuario"
"        ->  Nested Loop  (cost=5.34..124.19 rows=1 width=51) (actual time=0.042..0.072 rows=1 loops=1)"
"              Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag, c.id"
"              ->  Nested Loop  (cost=4.92..115.74 rows=1 width=51) (actual time=0.034..0.064 rows=1 loops=1)"
"                    Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag, p.fk_cartao"
"                    ->  Bitmap Heap Scan on public.parcelamento p  (cost=4.50..39.60 rows=9 width=8) (actual time=0.014..0.021 rows=8 loops=1)"
"                          Output: p.num_parcelas, p.fk_despesa, p.fk_cartao"
"                          Recheck Cond: (p.fk_cartao = 19247)"
"                          Heap Blocks: exact=8"
"                          ->  Bitmap Index Scan on idx_cartao_parc  (cost=0.00..4.50 rows=9 width=0) (actual time=0.009..0.009 rows=8 loops=1)"
"                                Index Cond: (p.fk_cartao = 19247)"
"                    ->  Index Scan using despesa_pkey on public.despesa d  (cost=0.43..8.45 rows=1 width=47) (actual time=0.005..0.005 rows=0
loops=8)"
"                          Output: d.valor, d.data_compra, d.fixo, d.id, d.nome, d.fk_pessoa_usuario, d.fk_categoria_despesa, d.fk_forma_pag"
"                          Index Cond: (d.id = p.fk_despesa)"
"                          Filter: ((d.data_compra > '2018-11-01'::date) AND (d.data_compra < '2018-11-30'::date))"
"                          Rows Removed by Filter: 1"
"              ->  Index Only Scan using cartao_pkey on public.cartao c  (cost=0.42..8.44 rows=1 width=4) (actual time=0.007..0.007 rows=1
loops=1)"
"                    Output: c.id"
"                    Index Cond: (c.id = 19247)"
"                    Heap Fetches: 1"
"        ->  Index Only Scan using mult_idx_pc on public.pessoa_cartao pc  (cost=0.29..8.31 rows=1 width=8) (actual time=0.274..0.274 rows=1
loops=1)"
"              Output: pc.fk_cartao, pc.fk_pessoa_usuario"
"              Index Cond: ((pc.fk_cartao = 19247) AND (pc.fk_pessoa_usuario = 2291))"
"              Heap Fetches: 1"
"  ->  Index Only Scan using pessoa_usuario_pkey on public.pessoa_usuario pu  (cost=0.29..8.30 rows=1 width=4) (actual time=0.007..0.007
rows=1 loops=1)"
"        Output: pu.id"
"        Index Cond: (pu.id = 2291)"
"        Heap Fetches: 1"
"Planning time: 1.215 ms"
"Execution time: 0.461 ms"
```

Testes de Performance

Sem index (multicolumn)

Nº de vezes	Planning time(ms)	Execution time (ms)
1	1.434	4.832
2	0.519	4.760
3	0.505	5.830
4	0.511	4.773
5	0.494	4.699
6	0.498	4.777
7	0.494	5.095
Média	0.505 ms	4.847 ms

Com index(multicolumn)

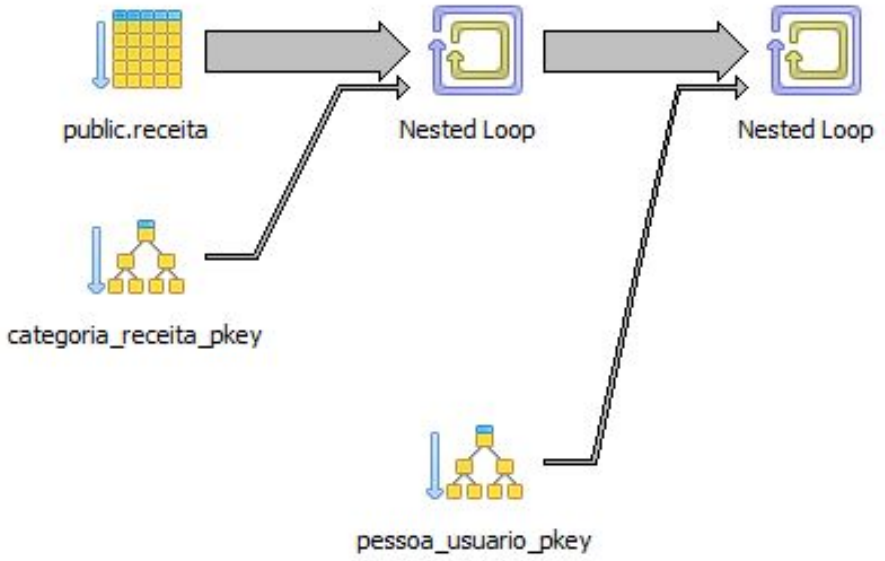
Nº de vezes	Planning time(ms)	Execution time (ms)
1	1.215	0.461
2	0.551	0.265
3	0.529	0.245
4	0.504	0.217
5	1.103	0.215
6	0.770	0.212
7	0.829	0.248
Média	0.756 ms	0.238 ms

Consulta de receitas de uma pessoa por um determinado mês:

Query 4

```
SELECT r.* FROM receita r
JOIN categoria_receita cr ON (cr.id = r.fk_categoria_receita)
JOIN pessoa_usuario pu ON (pu.id = r.fk_pessoa_usuario)
WHERE pu.id = 9242 AND cr.id = 2 AND r.data_recebimento > '2018-11-01' AND r.data_recebimento < '2018-11-30';
```

Sem Index


public.receita
Nested Loop
Nested Loop
categoria_receita_pkey
pessoa_usuario_pkey

Detalhamento do Explain

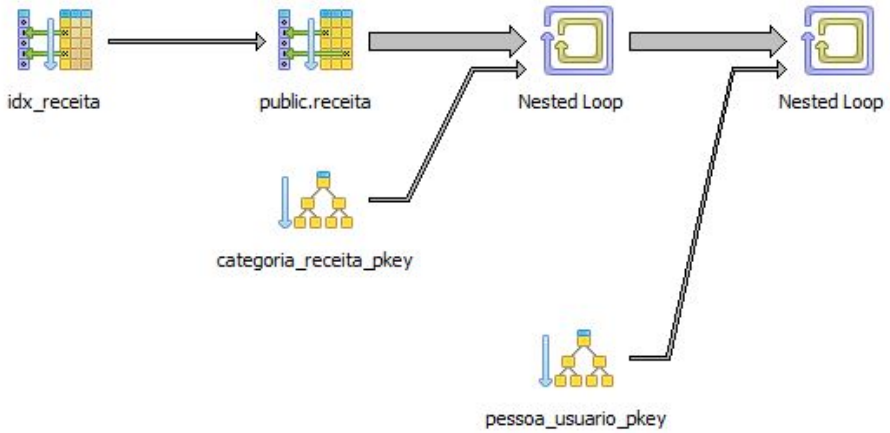
Sem index

```
"Nested Loop  (cost=0.44..43852.49 rows=1 width=43) (actual time=29.594..330.079 rows=3 loops=1)"
"  Output: r.id, r.valor, r.data_recebimento, r.fixo, r.nome, r.fk_pessoa_usuario, r.fk_categoria_receita"
"  ->  Nested Loop  (cost=0.15..43844.18 rows=1 width=43) (actual time=29.588..330.057 rows=3 loops=1)"
"        Output: r.id, r.valor, r.data_recebimento, r.fixo, r.nome, r.fk_pessoa_usuario, r.fk_categoria_receita"
"        ->  Seq Scan on public.receita r  (cost=0.00..43836.00 rows=1 width=43) (actual time=29.577..330.025 rows=3 loops=1)"
"              Output: r.id, r.valor, r.data_recebimento, r.fixo, r.nome, r.fk_pessoa_usuario, r.fk_categoria_receita"
"              Filter: ((r.data_recebimento > '2018-11-01'::date) AND (r.data_recebimento < '2018-11-30'::date) AND (r.fk_categoria_receita = 2) AND (r.fk_pessoa_usuario = 9242))"
"              Rows Removed by Filter: 1499997"
"        ->  Index Only Scan using categoria_receita_pkey on public.categoria_receita cr  (cost=0.15..8.17 rows=1 width=4) (actual time=0.007..0.007 rows=1 loops=3)"
"              Output: cr.id"
"              Index Cond: (cr.id = 2)"
"              Heap Fetches: 3"
"  ->  Index Only Scan using pessoa_usuario_pkey on public.pessoa_usuario pu  (cost=0.29..8.30 rows=1 width=4) (actual time=0.005..0.006 rows=1 loops=3)"
"        Output: pu.id"
"        Index Cond: (pu.id = 9242)"
"        Heap Fetches: 3"
"Planning time: 0.444 ms"
"Execution time: 330.170 ms"
```

Criação do index

```
CREATE INDEX idx_receita ON receita(fk_pessoa_usuario);
```

Com Index


idx_receita
public.receita
Nested Loop
Nested Loop
categoria_receita_pkey
pessoa_usuario_pkey

Detalhamento do Explain

Com index

```
"Nested Loop  (cost=5.97..567.50 rows=1 width=43) (actual time=0.674..6.039 rows=3 loops=1)"
"  Output: r.id, r.valor, r.data_recebimento, r.fixo, r.nome, r.fk_pessoa_usuario, r.fk_categoria_receita"
"  ->  Nested Loop  (cost=5.68..559.19 rows=1 width=43) (actual time=0.668..6.027 rows=3 loops=1)"
"        Output: r.id, r.valor, r.data_recebimento, r.fixo, r.nome, r.fk_pessoa_usuario,
r.fk_categoria_receita"
"        ->  Bitmap Heap Scan on public.receita r  (cost=5.53..551.01 rows=1 width=43) (actual
time=0.660..6.014 rows=3 loops=1)"
"              Output: r.id, r.valor, r.data_recebimento, r.fixo, r.nome, r.fk_pessoa_usuario,
r.fk_categoria_receita"
"              Recheck Cond: (r.fk_pessoa_usuario = 9242)"
"              Filter: ((r.data_recebimento > '2018-11-01'::date) AND (r.data_recebimento <
'2018-11-30'::date) AND (r.fk_categoria_receita = 2))"
"              Rows Removed by Filter: 188"
"              Heap Blocks: exact=191"
"              ->  Bitmap Index Scan on idx_receita  (cost=0.00..5.53 rows=147 width=0) (actual
time=0.116..0.116 rows=191 loops=1)"
"                    Index Cond: (r.fk_pessoa_usuario = 9242)"
"        ->  Index Only Scan using categoria_receita_pkey on public.categoria_receita cr  (cost=0.15..8.17
rows=1 width=4) (actual time=0.003..0.003 rows=1 loops=3)"
"              Output: cr.id"
"              Index Cond: (cr.id = 2)"
"              Heap Fetches: 3"
"  ->  Index Only Scan using pessoa_usuario_pkey on public.pessoa_usuario pu  (cost=0.29..8.30
rows=1 width=4) (actual time=0.003..0.004 rows=1 loops=3)"
"        Output: pu.id"
"        Index Cond: (pu.id = 9242)"
"        Heap Fetches: 3"
"Planning time: 0.921 ms"
"Execution time: 6.097 ms"
```

Testes de Performance

Sem index

Nº de vezes	Planning time(ms)	Execution time (ms)
1	0.444	330.170
2	0.263	330.071
3	0.438	336.832
4	0.257	336.432
5	0.203	326.706
6	0.458	327.613
7	0.438	333.585
Média	0.368 ms	331.574 ms

Com index

Nº de vezes	Planning time(ms)	Execution time (ms)
1	0.921	6.097
2	0.498	0.398
3	0.222	0.348
4	0.454	0.359
5	0.486	0.384
6	0.493	0.388
7	0.227	0.346
Média	0.432 ms	0.375 ms

Outros tipos de índices utilizados:

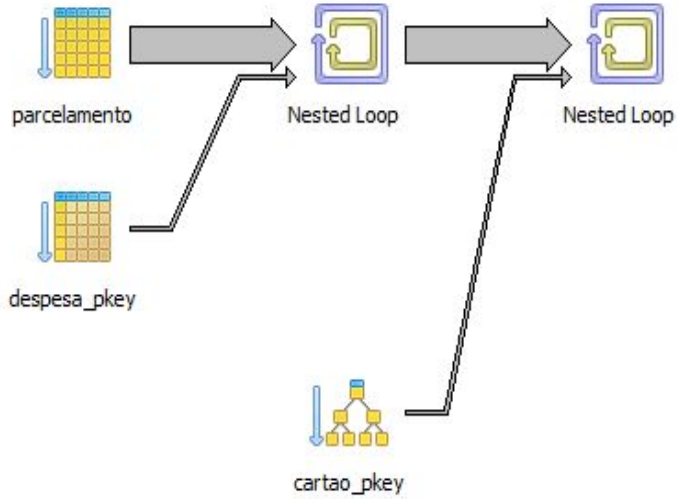
GIN
GIST
HASH

Criação do index GIN

- Tempo de criação: 15,8 segundos.
- Consulta utilizada: Query 1

```
CREATE INDEX gin_idx_despesa ON despesa USING gin (nome gin_trgm_ops);
```

Com Index GIN



Planning time: 1.201 ms

Execution time: 280.844 ms

Detalhamento do Explain

Com index GIN

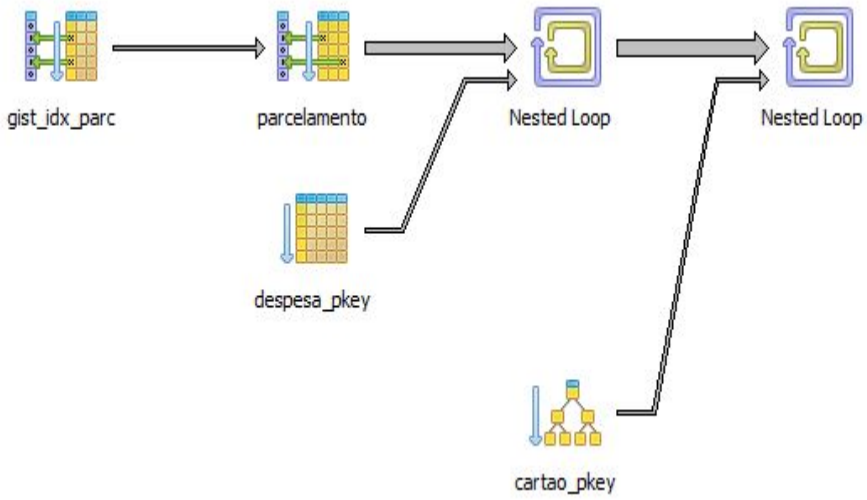
```
"Nested Loop  (cost=0.85..21571.59 rows=1 width=47) (actual
time=112.718..239.564 rows=5 loops=1)"
"  ->  Nested Loop  (cost=0.43..21563.14 rows=1 width=51) (actual
time=112.393..239.205 rows=5 loops=1)"
"        ->  Seq Scan on parcelamento p  (cost=0.00..21487.00 rows=9 width=8)
(actual time=104.033..238.996 rows=14 loops=1)"
"              Filter: (fk_cartao = 85107)"
"              Rows Removed by Filter: 1199986"
"        ->  Index Scan using despesa_pkey on despesa d  (cost=0.43..8.45 rows=1
width=47) (actual time=0.011..0.011 rows=0 loops=14)"
"              Index Cond: (id = p.fk_despesa)"
"              Filter: ((data_compra > '2018-11-01'::date) AND (data_compra <
'2018-11-30'::date))"
"              Rows Removed by Filter: 1"
"  ->  Index Only Scan using cartao_pkey on cartao c  (cost=0.42..8.44 rows=1
width=4) (actual time=0.070..0.070 rows=1 loops=5)"
"        Index Cond: (id = 85107)"
"        Heap Fetches: 5"
"Planning time: 16.473 ms"
"Execution time: 239.717 ms"
```

Criação do index GIST

- Tempo de criação: 15,5 segundos.
- Consulta utilizada: Query 1

```
CREATE INDEX gist_idx_parc ON parcelamento USING gist(fk_cartao);
```

Com Index GIST



Detalhamento do Explain

Com index GIST

```
"Nested Loop  (cost=5.20..124.05 rows=1 width=47) (actual time=0.146..0.558 rows=5 loops=1)"
"  ->  Nested Loop  (cost=4.78..115.60 rows=1 width=51) (actual time=0.138..0.534 rows=5 loops=1)"
"        ->  Bitmap Heap Scan on parcelamento p  (cost=4.35..39.46 rows=9 width=8) (actual time=0.093..0.435 rows=14 loops=1)"
"              Recheck Cond: (fk_cartao = 85107)"
"              Heap Blocks: exact=14"
"              ->  Bitmap Index Scan on gist_idx_parc  (cost=0.00..4.35 rows=9 width=0) (actual time=0.038..0.038 rows=14 loops=1)"
"                    Index Cond: (fk_cartao = 85107)"
"        ->  Index Scan using despesa_pkey on despesa d  (cost=0.43..8.45 rows=1 width=47) (actual time=0.006..0.006 rows=0 loops=14)"
"              Index Cond: (id = p.fk_despesa)"
"              Filter: ((data_compra > '2018-11-01'::date) AND (data_compra < '2018-11-30'::date))"
"              Rows Removed by Filter: 1"
"  ->  Index Only Scan using cartao_pkey on cartao c  (cost=0.42..8.44 rows=1 width=4) (actual time=0.004..0.004 rows=1 loops=5)"
"        Index Cond: (id = 85107)"
"        Heap Fetches: 5"
"Planning time: 5.651 ms"
"Execution time: 0.624 ms"
```

Testes de Performance

Sem index

Nº de vezes	Planning time(ms)	Execution time (ms)
1	0.543	191.964
2	0.679	265.687
3	0.815	90.497
4	0.353	190.347
5	0.369	192.308
6	0.369	187.168
7	0.345	191.624
Média	0.459 ms	191.348 ms

Com index GIST

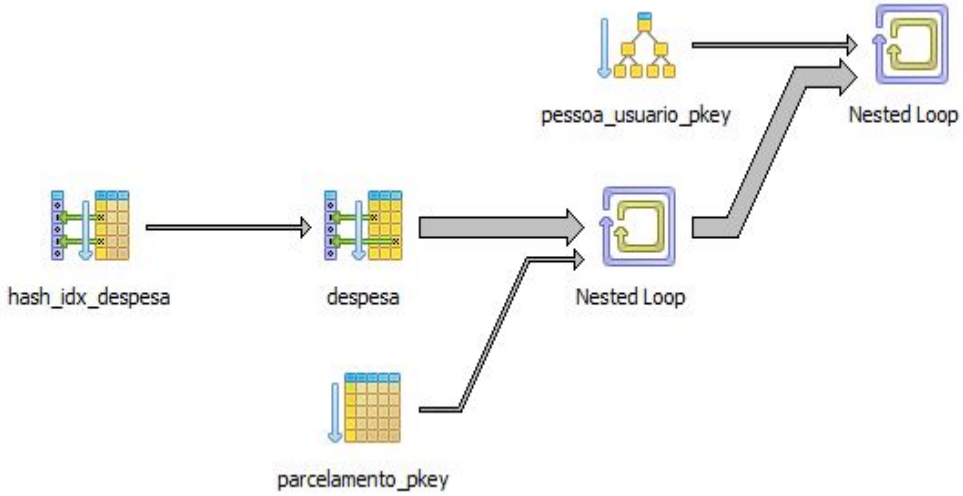
Nº de vezes	Planning time(ms)	Execution time (ms)
1	5.651	0.624
2	0.448	0.241
3	0.358	0.184
4	0.488	0.242
5	0.355	0.196
6	0.668	0.259
7	0.355	0.185
Média	0.463 ms	0.225 ms

Criação do index hash

- Tempo de criação: 2,8 segundos.
- Consulta utilizada: Query 2

```
CREATE INDEX hash_idx_despesa ON despesa USING hash (fk_pessoa_usuario);
```

Com Index
HASH



Detalhamento do Explain

Com index HASH

```
"Nested Loop  (cost=5.82..611.34 rows=5 width=59) (actual time=0.097..0.335 rows=11 loops=1)"
"  ->  Index Only Scan using pessoa_usuario_pkey on pessoa_usuario (cost=0.29..8.30 rows=1 width=4) (actual time=0.007..0.007 rows=1 loops=1)"
"        Index Cond: (id = 2291)"
"        Heap Fetches: 1"
"  ->  Nested Loop  (cost=5.53..602.98 rows=5 width=59) (actual time=0.082..0.316 rows=11 loops=1)"
"        ->  Bitmap Heap Scan on despesa  (cost=5.10..552.25 rows=6 width=47) (actual time=0.075..0.240 rows=13 loops=1)"
"              Recheck Cond: (fk_pessoa_usuario = 2291)"
"              Filter: ((data_compra > '2018-11-01'::date) AND (data_compra < '2018-11-30'::date))"
"              Rows Removed by Filter: 156"
"              Heap Blocks: exact=169"
"              ->  Bitmap Index Scan on hash_idx_despesa  (cost=0.00..5.10 rows=147 width=0) (actual time=0.030..0.031 rows=169 loops=1)"
"                    Index Cond: (fk_pessoa_usuario = 2291)"
"        ->  Index Scan using parcelamento_pkey on parcelamento  (cost=0.43..8.45 rows=1 width=12) (actual time=0.005..0.005 rows=1 loops=13)"
"              Index Cond: (fk_despesa = despesa.id)"
"Planning time: 6.036 ms"
"Execution time: 0.391 ms"
```

Testes de Performance

Sem index

Nº de vezes	Planning time(ms)	Execution time (ms)
1	0.782	247.345
2	0.288	239.091
3	0.304	241.255
4	0.327	279.761
5	0.288	237.932
6	0.291	231.078
7	0.288	233.426
Média	0.299 ms	239.809 ms

Com index HASH

Nº de vezes	Planning time(ms)	Execution time (ms)
1	6.036	0.391
2	0.307	0.364
3	0.322	0.311
4	0.321	0.337
5	0.307	0.324
6	0.311	0.33*
7	0.338	0.324
Média	0.320 ms	0.338 ms