FILA ZERO

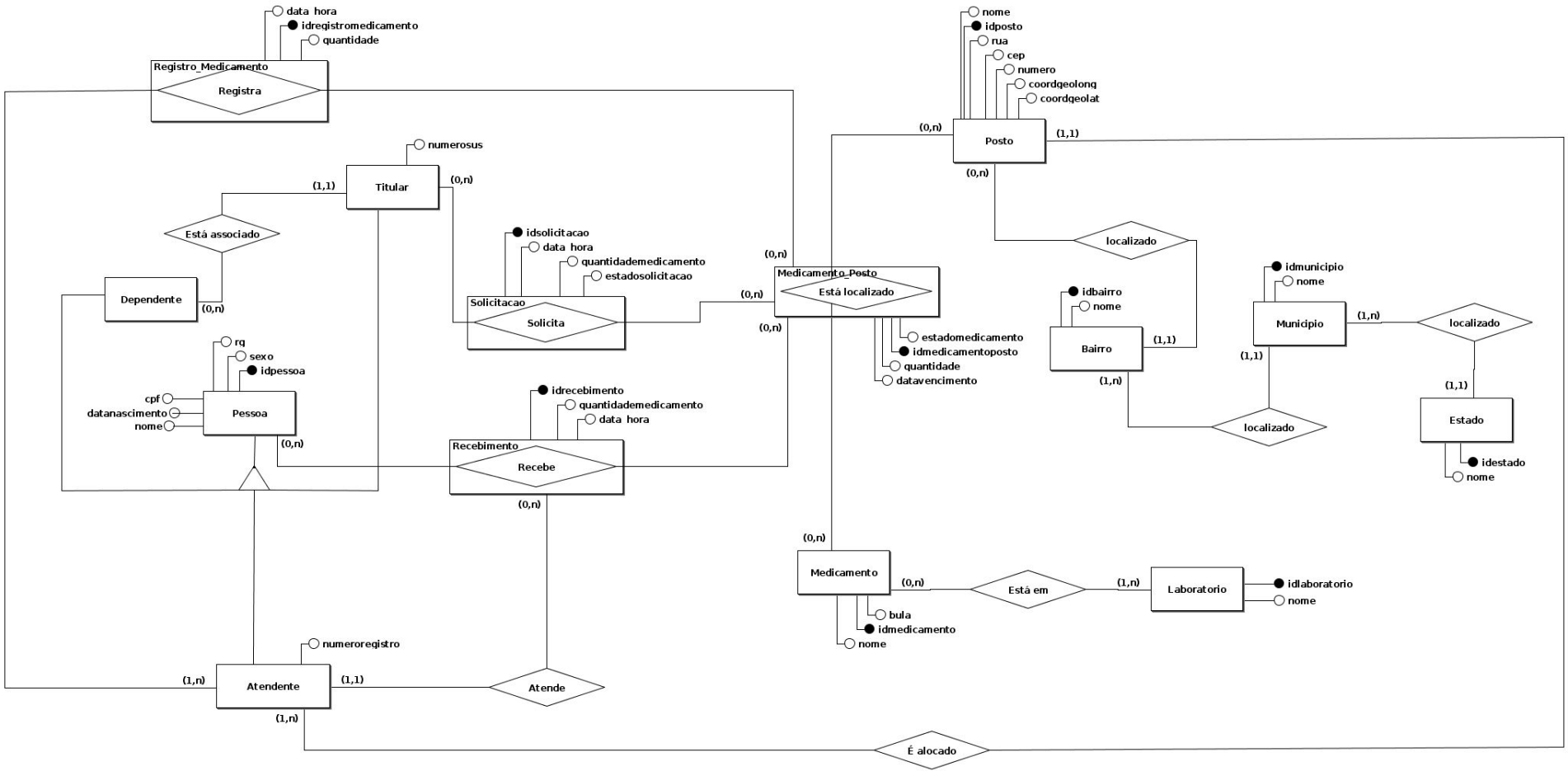
- **Harã Heique**
- **Jennifer Amaral**
- **Lucas Gomes**
- **Luiz Henrique**

Introdução

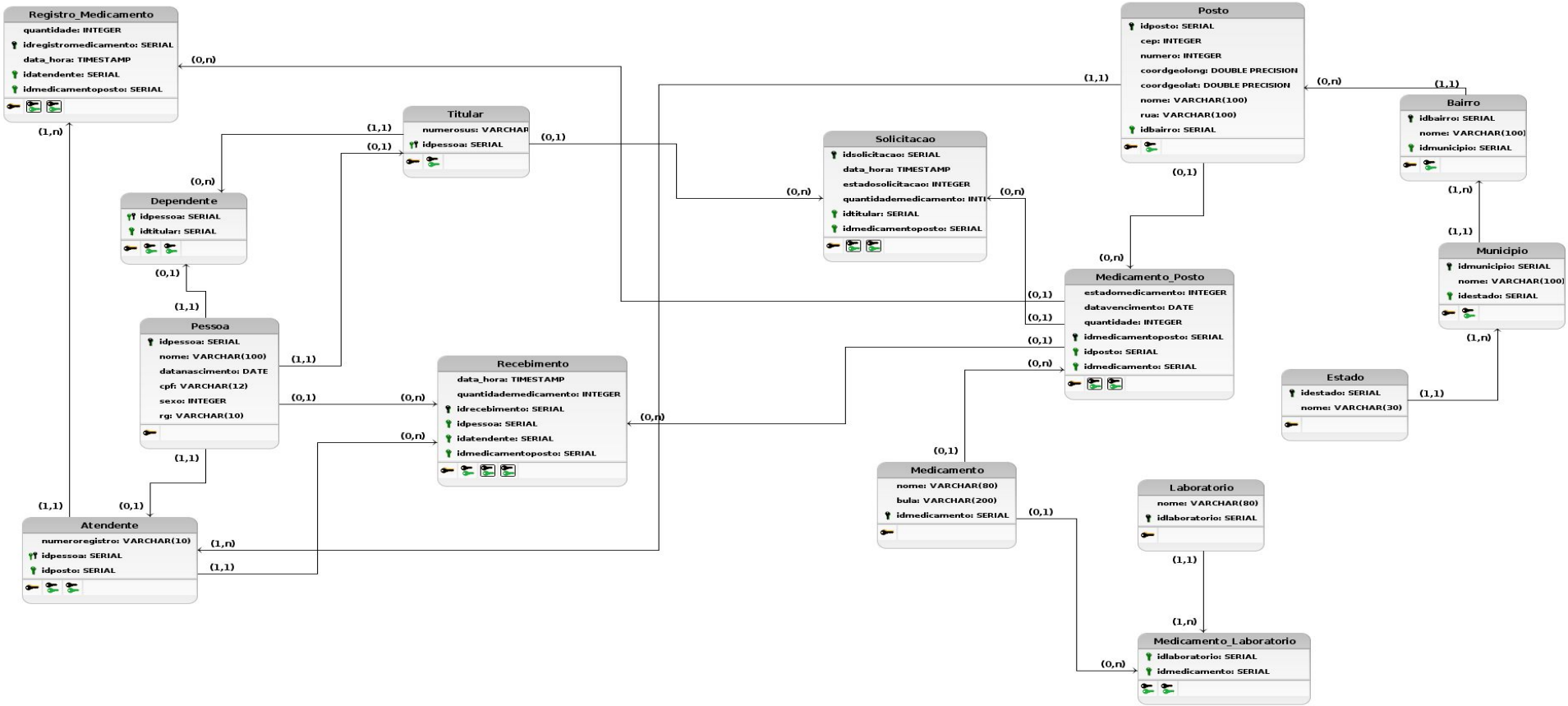
Os motivos da escolha do sistema proposto são da população não ter a necessidade de ir até o posto sem ter a certeza da obtenção do medicamento, o que consequentemente evitaria filas enormes para a solicitação.

Modelagem do Banco

Modelagem Conceitual



Modelagem Lógica

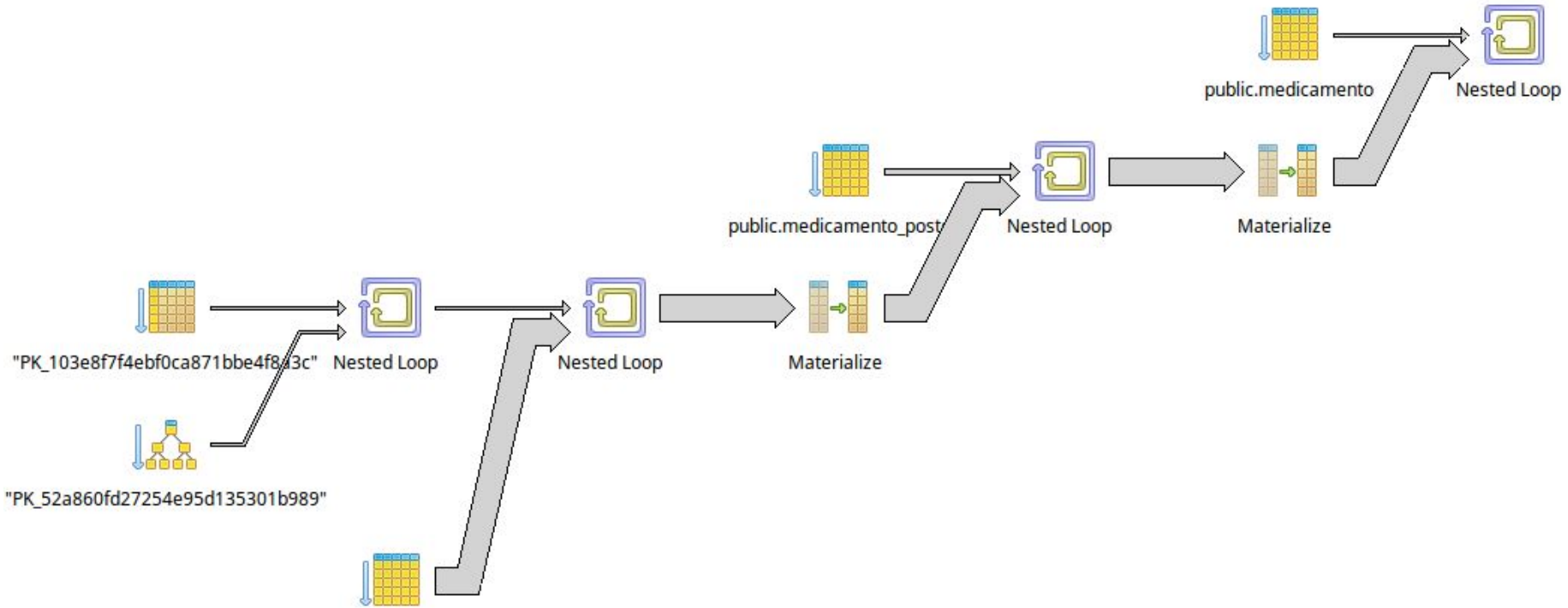


Performances

Query 1 - Retirar Medicamentos Solicitados

```
SELECT pe.nome AS "Paciente", me.nome AS "Medicamento", mp.quantidade AS
"Quantidade", so.estadosolicitacao AS "Estado" FROM pessoa AS pe
INNER JOIN titular AS ti ON (ti.idpessoa = pe.idpessoa)
INNER JOIN solicitacao AS so ON (so.idtitular = ti.idpessoa)
INNER JOIN medicamento_posto AS mp ON (mp.idmedicamentoposto =
so.idmedicamentoposto)
INNER JOIN medicamento AS me ON (me.idmedicamento = mp.idmedicamento)
WHERE estadosolicitacao = 2 AND  ti.idpessoa =1000 ;
```

Sem Index


public.medicamento
Nested Loop
public.medicamento_post
Nested Loop
Materialize
"PK_103e8f7f4ebf0ca871bbe4f8
Nested Loop
Nested Loop
Materialize
"PK_52a860fd27254e95d135301b989"

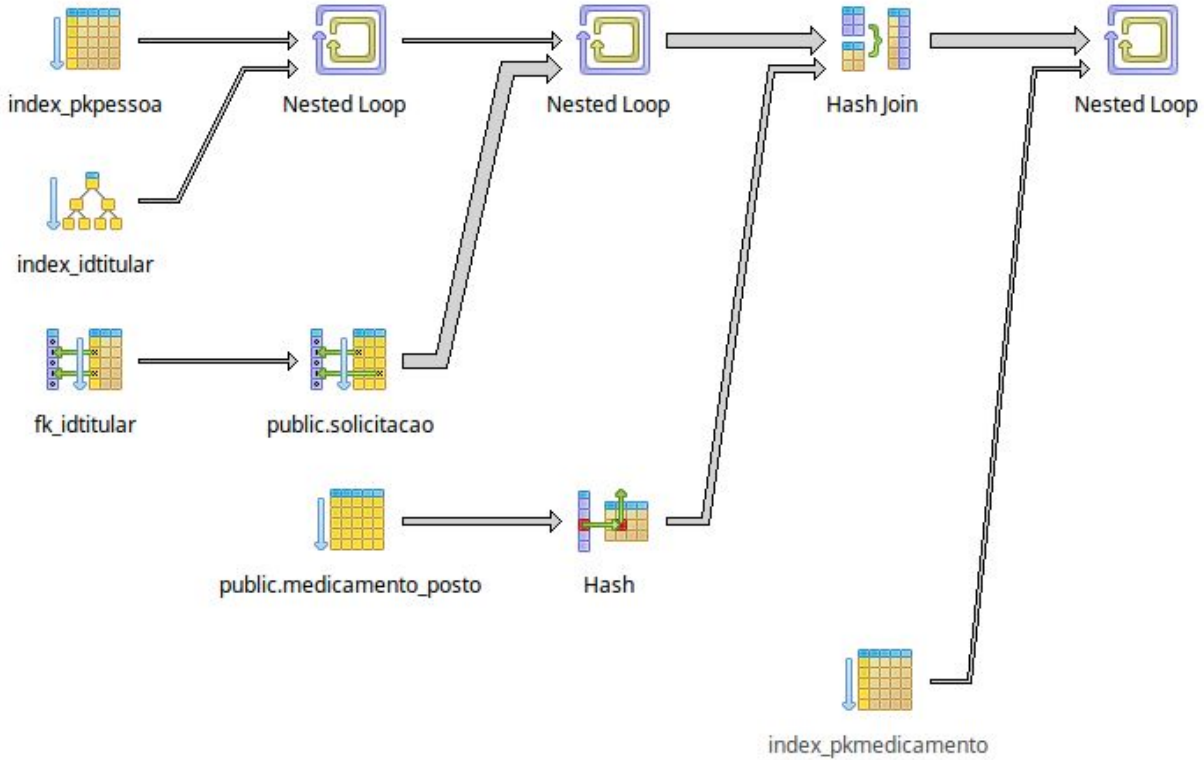
Detalhamento do explain

	QUERY PLAN text
1	Nested Loop (cost=0.58..33874.76 rows=19 width=52)
2	Output: pe.nome, me.nome, mp.quantidade, so.estadosolicitacao
3	Join Filter: (mp.idmedicamento = me.idmedicamento)
4	-> Seq Scan on public.medicamento me (cost=0.00..2.91 rows=91 width=25)
5	Output: me.idmedicamento, me.nome, me.bula
6	-> Materialize (cost=0.58..33845.96 rows=19 width=35)
7	Output: pe.nome, so.estadosolicitacao, mp.quantidade, mp.idmedicamento
8	-> Nested Loop (cost=0.58..33845.86 rows=19 width=35)
9	Output: pe.nome, so.estadosolicitacao, mp.quantidade, mp.idmedicamento
10	Join Filter: (so.idmedicamentoposto = mp.idmedicamentoposto)
11	-> Seq Scan on public.medicamento_posto mp (cost=0.00..18.00 rows=1000 width=12)
12	Output: mp.idmedicamentoposto, mp.estadomedicamento, mp.quantidade, mp.datavencimento, mp.idposto, mp.idmedicamento
13	-> Materialize (cost=0.58..33542.91 rows=19 width=31)
14	Output: pe.nome, so.estadosolicitacao, so.idmedicamentoposto
15	-> Nested Loop (cost=0.58..33542.82 rows=19 width=31)
16	Output: pe.nome, so.estadosolicitacao, so.idmedicamentoposto
17	-> Nested Loop (cost=0.58..12.63 rows=1 width=27)
18	Output: pe.nome, ti.idpessoa
19	-> Index Scan using "PK_103e8f7f4ebf0ca871bbe4f8a3c" on public.pessoa pe (cost=0.29..8.31 rows=1 width=27)
20	Output: pe.idpessoa, pe.nome, pe.datanascimento, pe.cpf, pe.sexo, pe.rg
21	Index Cond: (pe.idpessoa = 1000)
22	-> Index Only Scan using "PK_52a860fd27254e95d135301b989" on public.titular ti (cost=0.29..4.31 rows=1 width=4)
23	Output: ti.idpessoa
24	Index Cond: (ti.idpessoa = 1000)
25	-> Seq Scan on public.solicitacao so (cost=0.00..33530.00 rows=19 width=12)
26	Output: so.idsolicitacao, so.data_hora, so.quantidademedicamento, so.estadosolicitacao, so.idtitular, so.idmedicamentoposto
27	Filter: ((so.idtitular = 1000) AND (so.estadosolicitacao = 2))

Criação dos Indices

```
create index fk_idtitular on solicitacao (idtitular)
create index fk_idmedicamentoposto on solicitacao (idmedicamentoposto)
create index fk_pessoa on titular (idpessoa)
create index index_idmedicamentoposto on recebimento (idmedicamentoposto);
create index index_idpessoa on recebimento (idpessoa);
create index index_estadosolicitacao on solicitacao (estadosolicitacao);
```

Com Index


index_pkpessoa
Nested Loop
Nested Loop
Hash Join
Nested Loop
index_idtitular
fk_idtitular
public.solicitacao
public.medicamento_posto
Hash
index_pkmedicamento

Detalhamento do explain

	QUERY PLAN text
1	Nested Loop (cost=35.94..197.71 rows=19 width=52)
2	Output: pe.nome, me.nome, mp.quantidade, so.estadosolicitacao
3	-> Hash Join (cost=35.80..194.17 rows=19 width=35)
4	Output: pe.nome, so.estadosolicitacao, mp.quantidade, mp.idmedicamento
5	Hash Cond: (so.idmedicamentoposto = mp.idmedicamentoposto)
6	-> Nested Loop (cost=5.30..163.41 rows=19 width=31)
7	Output: pe.nome, so.estadosolicitacao, so.idmedicamentoposto
8	-> Nested Loop (cost=0.58..12.63 rows=1 width=27)
9	Output: pe.nome, ti.idpessoa
10	-> Index Scan using index_pkpessoa on public.pessoa pe (cost=0.29..8.31 rows=1 width=27)
11	Output: pe.idpessoa, pe.nome, pe.datanascimento, pe.cpf, pe.sexo, pe.rg
12	Index Cond: (pe.idpessoa = 1000)
13	-> Index Only Scan using index_idtitular on public.titular ti (cost=0.29..4.31 rows=1 width=4)
14	Output: ti.idpessoa
15	Index Cond: (ti.idpessoa = 1000)
16	-> Bitmap Heap Scan on public.solicitacao so (cost=4.72..150.60 rows=19 width=12)
17	Output: so.idsolicitacao, so.data_hora, so.quantidademedicamento, so.estadosolicitacao, so.idtitular, so.idmedicamentoposto
18	Recheck Cond: (so.idtitular = 1000)
19	Filter: (so.estadosolicitacao = 2)
20	-> Bitmap Index Scan on fk_idtitular (cost=0.00..4.71 rows=38 width=0)
21	Index Cond: (so.idtitular = 1000)
22	-> Hash (cost=18.00..18.00 rows=1000 width=12)
23	Output: mp.quantidade, mp.idmedicamentoposto, mp.idmedicamento
24	-> Seq Scan on public.medicamento_posto mp (cost=0.00..18.00 rows=1000 width=12)
25	Output: mp.quantidade, mp.idmedicamentoposto, mp.idmedicamento
26	-> Index Scan using index_pkmedicamento on public.medicamento me (cost=0.14..0.18 rows=1 width=25)
27	Output: me.idmedicamento, me.nome, me.bula
28	Index Cond: (me.idmedicamento = mp.idmedicamento)

Teste de Performance

Query 1 - Retirar Medicamentos Solicitados		
Núm. Vezes	**Planning Time (ms)**	**Execution Time (ms)**
1	1,093	0,153
2	0,718	0,159
3	0,733	0,057
4	0,989	0,243
5	0,942	0,092
Média	**0,895**	**0,1408**

Com Index

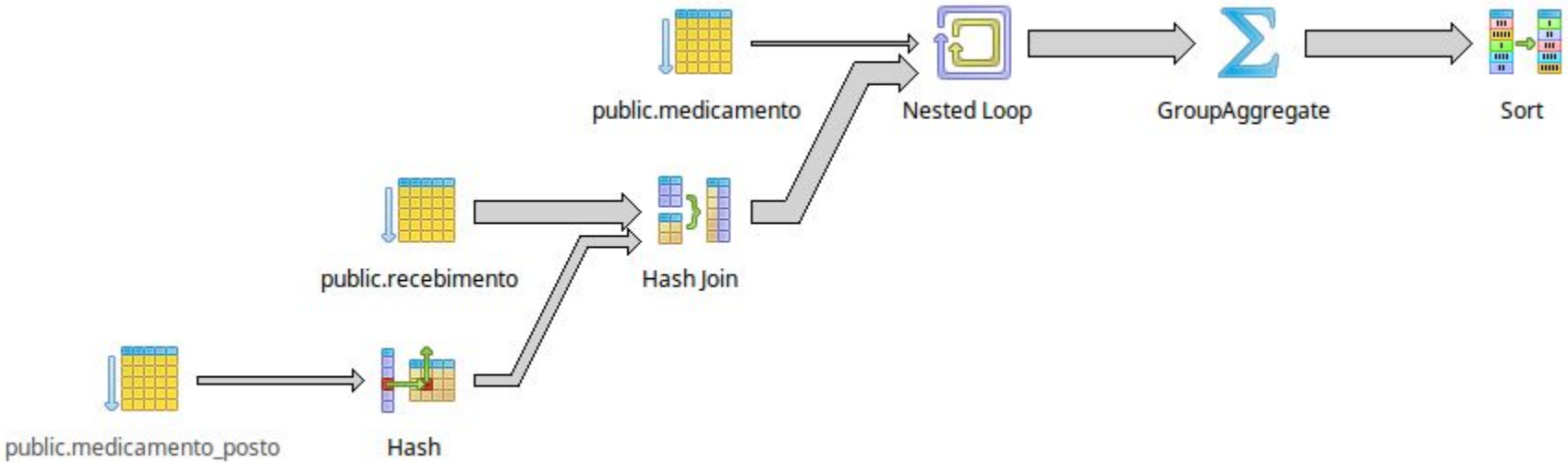
Query 1 - Retirar Medicamentos Solicitados		
Núm. Vezes	**Planning Time (ms)**	**Execution Time (ms)**
1	0,780	0,607
2	0,593	0,357
3	0,587	0,593
4	1,114	0,567
5	0,421	0,360
Média	**0,699**	**0,4968**

Sem Index

Query 2 - Medicamentos mais retirados por quantidade

```
SELECT me.nome, SUM(re.quantidademedicamentos) AS "Quantidade" FROM medicamento
AS me
INNER JOIN medicamento_posto AS mp ON (mp.idmedicamento = me.idmedicamento)
INNER JOIN recebimento AS re ON (re.idmedicamentoposto = mp.idmedicamentoposto)
GROUP BY me.idmedicamento
HAVING SUM(re.quantidademedicamentos) >= 8500 AND me.idmedicamento = 46
ORDER BY "Quantidade" DESC;
```

Sem Index


public.medicamento
Nested Loop
GroupAggregate
Sort
public.recebimento
Hash Join
public.medicamento_posto
Hash

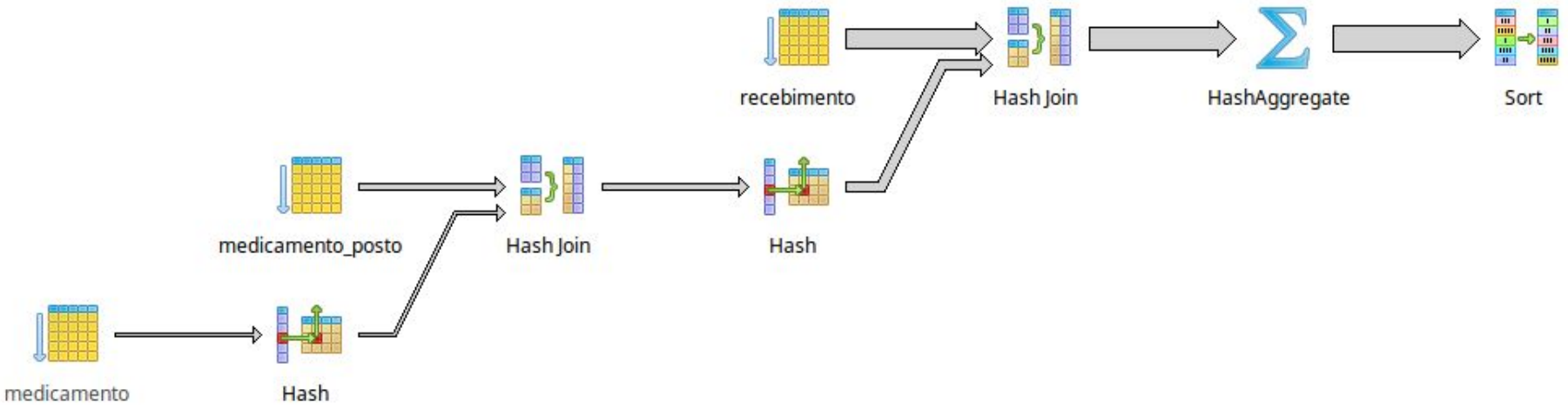
Detalhamento do explain

	QUERY PLAN text
1	Sort (cost=2192.67..2192.68 rows=1 width=29)
2	Output: me.nome, (sum(re.quantidademedicamentos)), me.idmedicamento
3	Sort Key: (sum(re.quantidademedicamentos)) DESC
4	-> GroupAggregate (cost=20.76..2192.66 rows=1 width=29)
5	Output: me.nome, sum(re.quantidademedicamentos), me.idmedicamento
6	Group Key: me.idmedicamento
7	Filter: (sum(re.quantidademedicamentos) >= 8500)
8	-> Nested Loop (cost=20.76..2176.90 rows=2100 width=29)
9	Output: me.idmedicamento, me.nome, re.quantidademedicamentos
10	-> Seq Scan on public.medicamento me (cost=0.00..3.14 rows=1 width=25)
11	Output: me.idmedicamento, me.nome, me.bula
12	Filter: (me.idmedicamento = 46)
13	-> Hash Join (cost=20.76..2152.76 rows=2100 width=8)
14	Output: mp.idmedicamento, re.quantidademedicamentos
15	Hash Cond: (re.idmedicamentoposto = mp.idmedicamentoposto)
16	-> Seq Scan on public.recebimento re (cost=0.00..1736.00 rows=100000 width=8)
17	Output: re.idrecebimento, re.quantidademedicamentos, re.data_hora, re.idpessoa, re.idatendente, re.idmedicamentoposto
18	-> Hash (cost=20.50..20.50 rows=21 width=8)
19	Output: mp.idmedicamento, mp.idmedicamentoposto
20	-> Seq Scan on public.medicamento_posto mp (cost=0.00..20.50 rows=21 width=8)
21	Output: mp.idmedicamento, mp.idmedicamentoposto
22	Filter: (mp.idmedicamento = 46)

Criação dos Indices

```
create index fk_idmedicamentoposto on solicitacao (idmedicamentoposto)
create index index_idmedicamentoposto on recebimento (idmedicamentoposto);
create index index_pkmedicamentoposto on medicamento_posto (idmedicamentoposto);
create index index_pkmedicamento on medicamento (idmedicamento);
create index index_pkrecebimento on recebimento (idrecebimento);
```

Com Index


recebimento
Hash Join
HashAggregate
Sort
medicamento_posto
Hash Join
Hash
medicamento
Hash

Detalhamento do explain

	QUERY PLAN text
1	Sort (cost=2184.87..2184.88 rows=1 width=29)
2	Output: me.nome, (sum(re.quantidademedicamentos)), me.idmedicamento
3	Sort Key: (sum(re.quantidademedicamentos)) DESC
4	-> GroupAggregate (cost=12.96..2184.86 rows=1 width=29)
5	Output: me.nome, sum(re.quantidademedicamentos), me.idmedicamento
6	Group Key: me.idmedicamento
7	Filter: (sum(re.quantidademedicamentos) >= 8500)
8	-> Nested Loop (cost=12.96..2169.10 rows=2100 width=29)
9	Output: me.idmedicamento, me.nome, re.quantidademedicamentos
10	-> Seq Scan on public.medicamento me (cost=0.00..3.14 rows=1 width=25)
11	Output: me.idmedicamento, me.nome, me.bula
12	Filter: (me.idmedicamento = 46)
13	-> Hash Join (cost=12.96..2144.96 rows=2100 width=8)
14	Output: mp.idmedicamento, re.quantidademedicamentos
15	Hash Cond: (re.idmedicamentoposto = mp.idmedicamentoposto)
16	-> Seq Scan on public.recebimento re (cost=0.00..1736.00 rows=100000 width=8)
17	Output: re.idrecebimento, re.quantidademedicamentos, re.data_hora, re.idpessoa, re.idatendente, re.idmedicamentoposto
18	-> Hash (cost=12.70..12.70 rows=21 width=8)
19	Output: mp.idmedicamento, mp.idmedicamentoposto
20	-> Bitmap Heap Scan on public.medicamento_posto mp (cost=4.44..12.70 rows=21 width=8)
21	Output: mp.idmedicamento, mp.idmedicamentoposto
22	Recheck Cond: (mp.idmedicamento = 46)
23	-> Bitmap Index Scan on fk_idmedicamento (cost=0.00..4.43 rows=21 width=0)
24	Index Cond: (mp.idmedicamento = 46)

Teste de Performance

Query 2 - Medicamentos mais Retirados por quantidade		
Núm. Vezes	**Planning Time (ms)**	**Execution Time (ms)**
1	0,643	20,284
2	0,467	18,639
3	0,741	19,242
4	0,461	21,030
5	0,559	21,620
Média	**0,5742**	**20,163**

Com Index

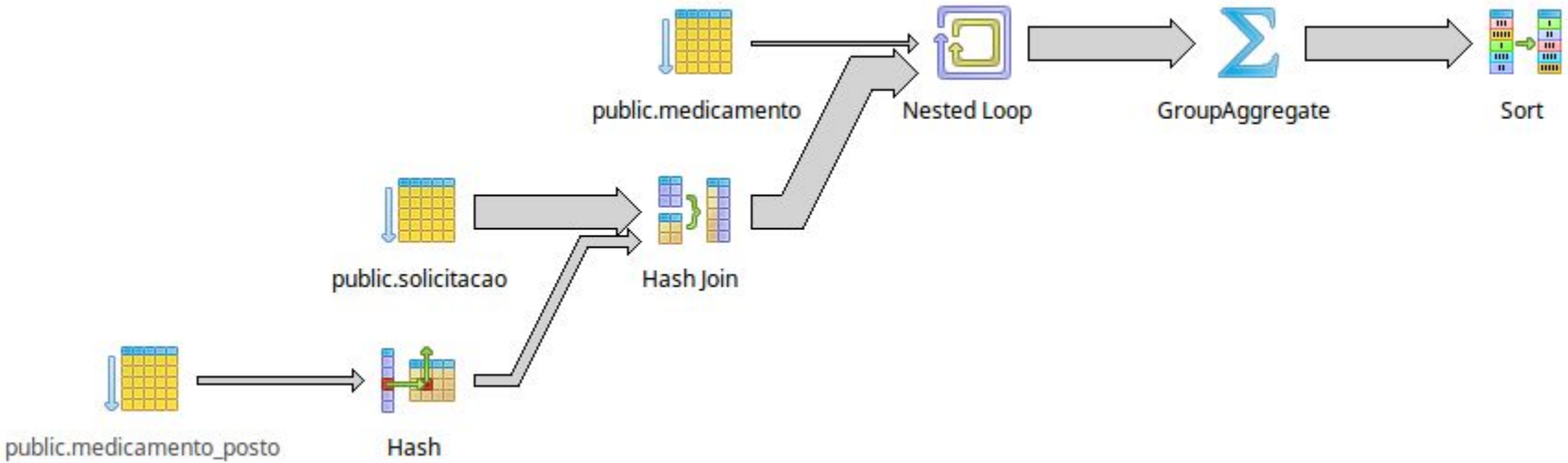
Query 2 - Medicamentos mais Retirados por quantidade		
Núm. Vezes	**Planning Time (ms)**	**Execution Time (ms)**
1	0,320	22,490
2	0,456	21,633
3	0,475	21,505
4	0,595	22,097
5	0,687	21,972
Média	**0,5066**	**21,9394**

Sem Index

Query 3 - Medicamentos mais solicitados por quantidade

```
SELECT me.nome, SUM(me.idmedicamento) AS "Quantidade" FROM medicamento AS me
INNER JOIN medicamento_posto AS mp ON (mp.idmedicamento = me.idmedicamento)
INNER JOIN solicitacao AS so ON (so.idmedicamentoposto = mp.idmedicamentoposto)
GROUP BY me.idmedicamento
HAVING SUM(me.idmedicamento) >= 1600000 AND me.idmedicamento = 88
ORDER BY "Quantidade" DESC;
```

Sem Index


public.medicamento
Nested Loop
GroupAggregate
Sort
public.solicitacao
Hash Join
public.medicamento_posto
Hash

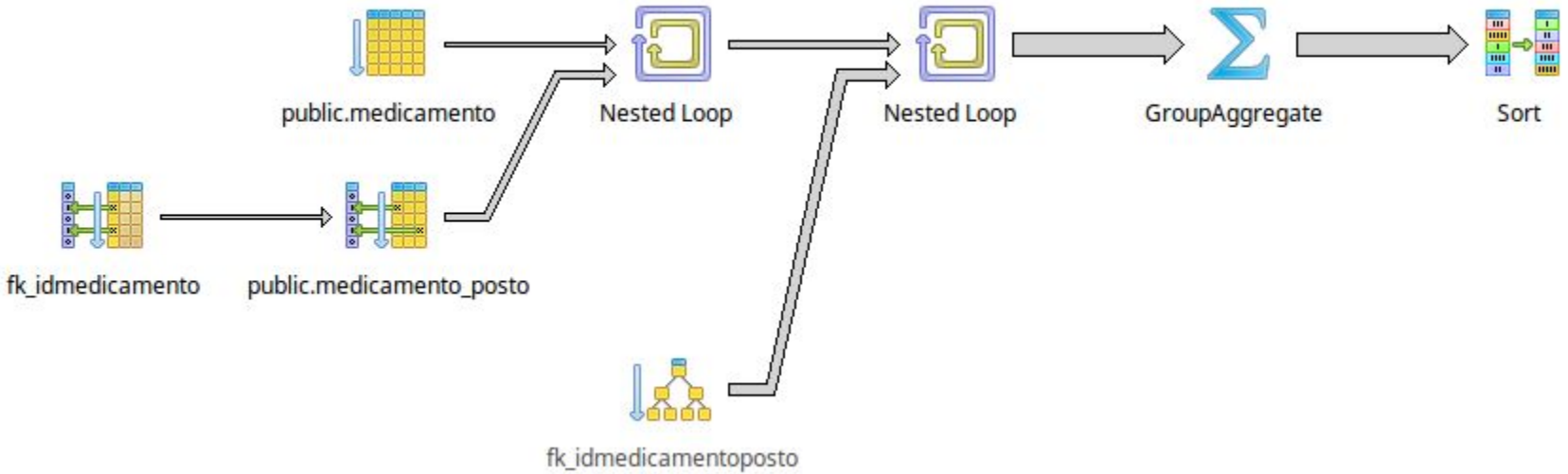
Detalhamento do explain

	QUERY PLAN text
1	Sort (cost=32297.60..32297.60 rows=1 width=25)
2	Output: me.nome, (sum(me.idmedicamento)), me.idmedicamento
3	Sort Key: (sum(me.idmedicamento)) DESC
4	-> GroupAggregate (cost=20.69..32297.59 rows=1 width=25)
5	Output: me.nome, sum(me.idmedicamento), me.idmedicamento
6	Group Key: me.idmedicamento
7	Filter: (sum(me.idmedicamento) >= 1600000)
8	-> Nested Loop (cost=20.69..32128.83 rows=22500 width=25)
9	Output: me.idmedicamento, me.nome
10	-> Seq Scan on public.medicamento me (cost=0.00..3.14 rows=1 width=25)
11	Output: me.idmedicamento, me.nome, me.bula
12	Filter: (me.idmedicamento = 88)
13	-> Hash Join (cost=20.69..31900.69 rows=22500 width=4)
14	Output: mp.idmedicamento
15	Hash Cond: (so.idmedicamentoposto = mp.idmedicamentoposto)
16	-> Seq Scan on public.solicitacao so (cost=0.00..26030.00 rows=1500000 width=4)
17	Output: so.idsolicitacao, so.data_hora, so.quantidademedicamento, so.estadosolicitacao, so.idtitular, so.idmedicamentoposto
18	-> Hash (cost=20.50..20.50 rows=15 width=8)
19	Output: mp.idmedicamento, mp.idmedicamentoposto
20	-> Seq Scan on public.medicamento_posto mp (cost=0.00..20.50 rows=15 width=8)
21	Output: mp.idmedicamento, mp.idmedicamentoposto
22	Filter: (mp.idmedicamento = 88)

Criação dos Indices

```
create index fk_idmedicamentoposto on solicitacao (idmedicamentoposto)
create index index_idmedicamentoposto on recebimento (idmedicamentoposto);
create index index_estadosolicitacao on solicitacao (estadosolicitacao);
create index index_pkmedicamento on medicamento (idmedicamento);
create index fk_idtitular on solicitacao (idmedicamentoposto)
```

Com Index


public.medicamento
Nested Loop
Nested Loop
GroupAggregate
Sort
fk_idmedicamento
public.medicamento_posto
fk_idmedicamentoposto

Detalhamento do explain

	QUERY PLAN text
1	Sort (cost=1109.80..1109.81 rows=1 width=25)
2	Output: me.nome, (sum(me.idmedicamento)), me.idmedicamento
3	Sort Key: (sum(me.idmedicamento)) DESC
4	-> GroupAggregate (cost=4.82..1109.79 rows=1 width=25)
5	Output: me.nome, sum(me.idmedicamento), me.idmedicamento
6	Group Key: me.idmedicamento
7	Filter: (sum(me.idmedicamento) >= 1600000)
8	-> Nested Loop (cost=4.82..941.03 rows=22500 width=25)
9	Output: me.idmedicamento, me.nome
10	-> Nested Loop (cost=4.39..15.87 rows=15 width=29)
11	Output: me.idmedicamento, me.nome, mp.idmedicamentoposto
12	-> Seq Scan on public.medicamento me (cost=0.00..3.14 rows=1 width=25)
13	Output: me.idmedicamento, me.nome, me.bula
14	Filter: (me.idmedicamento = 88)
15	-> Bitmap Heap Scan on public.medicamento_posto mp (cost=4.39..12.58 rows=15 width=8)
16	Output: mp.idmedicamentoposto, mp.estadomedicamento, mp.quantidade, mp.datavencimento, mp.idposto, mp.idmedicamento
17	Recheck Cond: (mp.idmedicamento = 88)
18	-> Bitmap Index Scan on fk_idmedicamento (cost=0.00..4.39 rows=15 width=0)
19	Index Cond: (mp.idmedicamento = 88)
20	-> Index Only Scan using fk_idmedicamentoposto on public.solicitacao so (cost=0.43..46.68 rows=1500 width=4)
21	Output: so.idmedicamentoposto
22	Index Cond: (so.idmedicamentoposto = mp.idmedicamentoposto)

Teste de Performance

Query 3 - Medicamentos mais Solicitados por quantidade		
Núm. Vezes	**Planning Time (ms)**	**Execution Time (ms)**
1	0,632	8,874
2	0,401	8,409
3	o,400	8,938
4	0,860	7,983
5	0,760	8,054
Média	**0,5306**	**8,4516**

Com Index

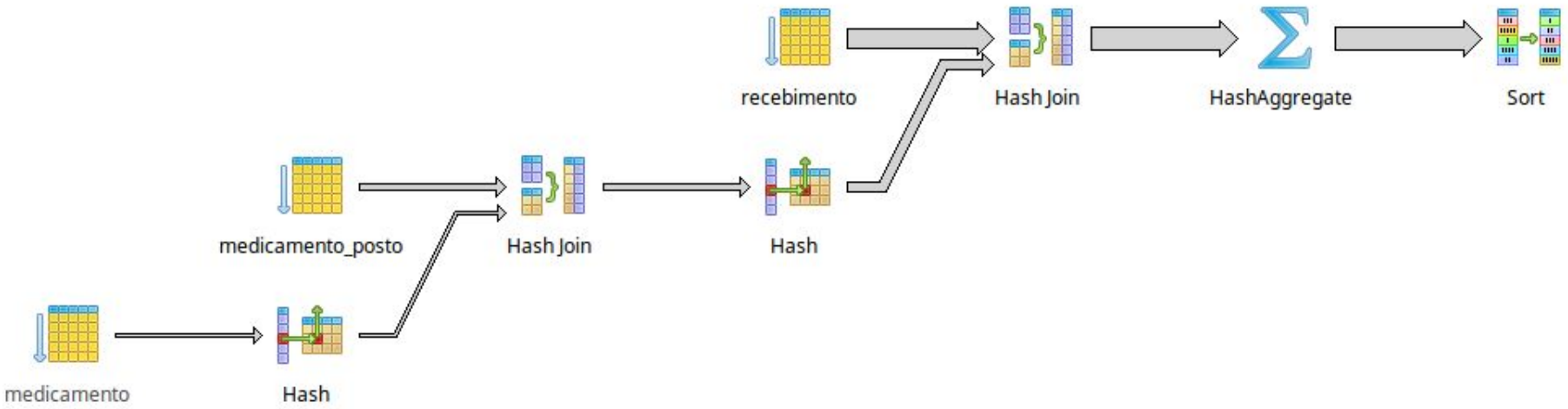
Query 3 - Medicamentos mais Solicitados por quantidade		
Núm. Vezes	**Planning Time (ms)**	**Execution Time (ms)**
1	0,462	307,574
2	0,602	317,208
3	0,451	316,297
4	0,266	315,784
5	0,221	310,819
Média	**0,4004**	**313,5364**

Sem Index

Query 4 - Medicamento mais Retirados por número retiradas

```
SELECT me.nome, COUNT(me.idmedicamento) AS "Número de Retiradas" FROM
medicamento AS me
INNER JOIN medicamento_posto AS mp ON (mp.idmedicamento = me.idmedicamento)
INNER JOIN recebimento AS re ON (re.idmedicamentoposto = mp.idmedicamentoposto)
GROUP BY me.idmedicamento
HAVING COUNT(me.idmedicamento) >= 1600 and nome ilike 'A%'
ORDER BY "Número de Retiradas" DESC;
```

Sem Index


recebimento
Hash Join
HashAggregate
Sort
medicamento_posto
Hash Join
Hash
medicamento
Hash

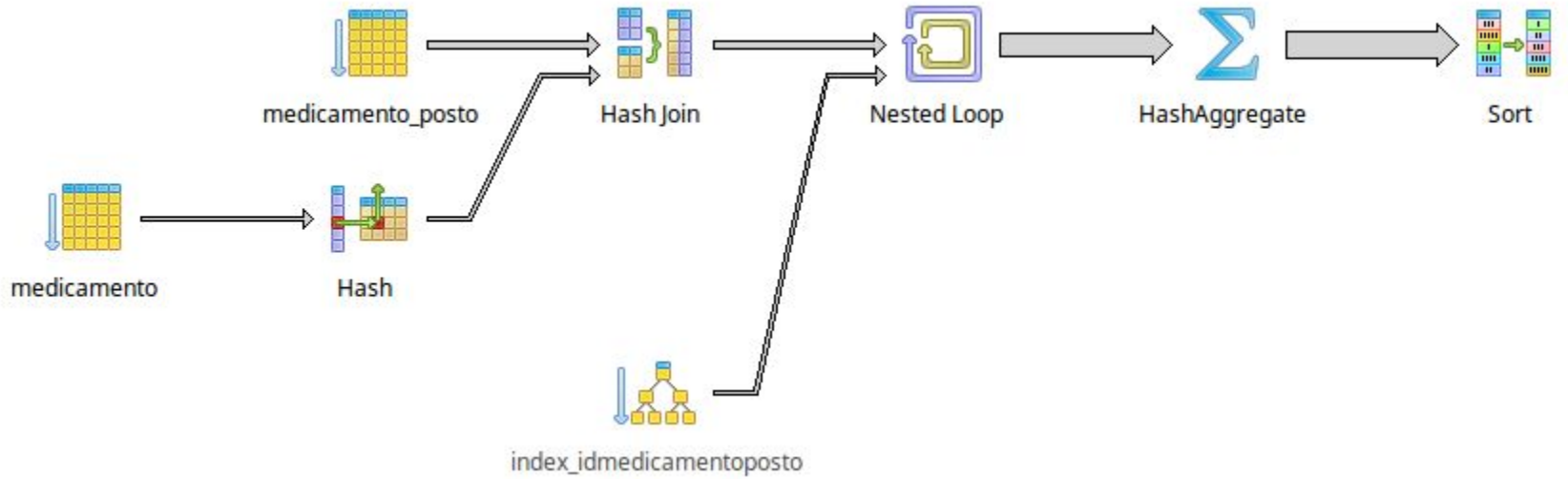
Detalhamento do resultado

	QUERY PLAN text
1	Sort (cost=2546.32..2546.37 rows=21 width=25)
2	Output: me.nome, (count(me.idmedicamento)), me.idmedicamento
3	Sort Key: (count(me.idmedicamento)) DESC
4	-> HashAggregate (cost=2545.60..2545.86 rows=21 width=25)
5	Output: me.nome, count(me.idmedicamento), me.idmedicamento
6	Group Key: me.idmedicamento
7	Filter: (count(me.idmedicamento) >= 1600)
8	-> Hash Join (cost=30.35..2372.35 rows=23100 width=25)
9	Output: me.idmedicamento, me.nome
10	Hash Cond: (re.idmedicamentoposto = mp.idmedicamentoposto)
11	-> Seq Scan on public.recebimento re (cost=0.00..1736.00 rows=100000 width=4)
12	Output: re.idrecebimento, re.quantidademedicamentos, re.data_hora, re.idpessoa, re.idatendente, re.idmedicamentoposto
13	-> Hash (cost=27.46..27.46 rows=231 width=29)
14	Output: me.idmedicamento, me.nome, mp.idmedicamentoposto
15	-> Hash Join (cost=3.40..27.46 rows=231 width=29)
16	Output: me.idmedicamento, me.nome, mp.idmedicamentoposto
17	Hash Cond: (mp.idmedicamento = me.idmedicamento)
18	-> Seq Scan on public.medicamento_posto mp (cost=0.00..18.00 rows=1000 width=8)
19	Output: mp.idmedicamentoposto, mp.estadomedicamento, mp.quantidade, mp.datavencimento, mp.idposto, mp.idmedicamento
20	-> Hash (cost=3.14..3.14 rows=21 width=25)
21	Output: me.idmedicamento, me.nome
22	-> Seq Scan on public.medicamento me (cost=0.00..3.14 rows=21 width=25)
23	Output: me.idmedicamento, me.nome
24	Filter: ((me.nome)::text ~~* 'A%'::text)

Criação dos Indices

```
create index fk_idmedicamentoposto on solicitacao (idmedicamentoposto)
create index index_idmedicamentoposto on recebimento (idmedicamentoposto);
create index index_pkmedicamentoposto on medicamento_posto (idmedicamentoposto);
create index index_pkmedicamento on medicamento (idmedicamento);
create index index_pkrecebimento on recebimento (idrecebimento);
```

Com Index


medicamento_posto
Hash Join
Nested Loop
HashAggregate
Sort
medicamento
Hash
index_idmedicamentoposto

Detalhamento do resultado

	QUERY PLAN text
1	Sort (cost=1159.28..1159.33 rows=21 width=25)
2	Output: me.nome, (count(me.idmedicamento)), me.idmedicamento
3	Sort Key: (count(me.idmedicamento)) DESC
4	-> HashAggregate (cost=1158.55..1158.81 rows=21 width=25)
5	Output: me.nome, count(me.idmedicamento), me.idmedicamento
6	Group Key: me.idmedicamento
7	Filter: (count(me.idmedicamento) >= 1600)
8	-> Nested Loop (cost=3.69..985.30 rows=23100 width=25)
9	Output: me.idmedicamento, me.nome
10	-> Hash Join (cost=3.40..27.46 rows=231 width=29)
11	Output: me.idmedicamento, me.nome, mp.idmedicamentoposto
12	Hash Cond: (mp.idmedicamento = me.idmedicamento)
13	-> Seq Scan on public.medicamento_posto mp (cost=0.00..18.00 rows=1000 width=8)
14	Output: mp.idmedicamentoposto, mp.estadomedicamento, mp.quantidade, mp.datavencimento, mp.idposto, mp.idmedicamento
15	-> Hash (cost=3.14..3.14 rows=21 width=25)
16	Output: me.idmedicamento, me.nome
17	-> Seq Scan on public.medicamento me (cost=0.00..3.14 rows=21 width=25)
18	Output: me.idmedicamento, me.nome
19	Filter: ((me.nome)::text ~~* 'A%'::text)
20	-> Index Only Scan using index_idmedicamentoposto on public.recebimento re (cost=0.29..3.15 rows=100 width=4)
21	Output: re.idmedicamentoposto
22	Index Cond: (re.idmedicamentoposto = mp.idmedicamentoposto)

Teste de Performance

Query 4 - Medicamentos mais Retirados por número retiradas		
Núm. Vezes	**Planning Time (ms)**	**Execution Time (ms)**
1	1,129	15,130
2	1,865	18,007
3	0,772	23,220
4	0,748	19,799
5	1,257	22,482
Média	**1,1542**	**20**

Com Index

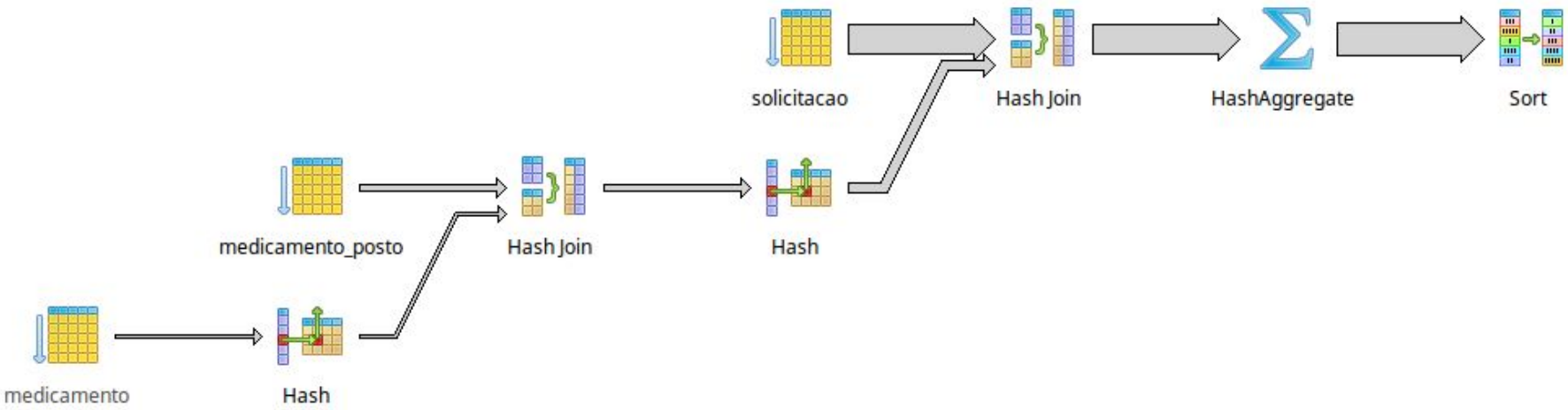
Query 4 - Medicamentos mais Retirados por número retiradas		
Núm. Vezes	**Planning Time (ms)**	**Execution Time (ms)**
1	0,508	33,517
2	0,386	34,868
3	0,456	33,136
4	0,722	43,007
5	0,722	35,527
Média	**0,5588**	**36,011**

Sem Index

Query 5 - Medicamentos mais Solicitados por número de solicitações

```
SELECT me.nome, COUNT(me.idmedicamento) AS "Número de Solicitações" FROM
medicamento AS me
INNER JOIN medicamento_posto AS mp ON (mp.idmedicamento = me.idmedicamento)
INNER JOIN solicitacao AS so ON (so.idmedicamentoposto = mp.idmedicamentoposto)
GROUP BY me.idmedicamento
HAVING COUNT(me.idmedicamento) >= 25000 and nome ilike 'A%'
ORDER BY "Número de Solicitações" DESC;
```

Sem Index


solicitacao
Hash Join
HashAggregate
Sort
medicamento_posto
Hash Join
Hash
medicamento
Hash

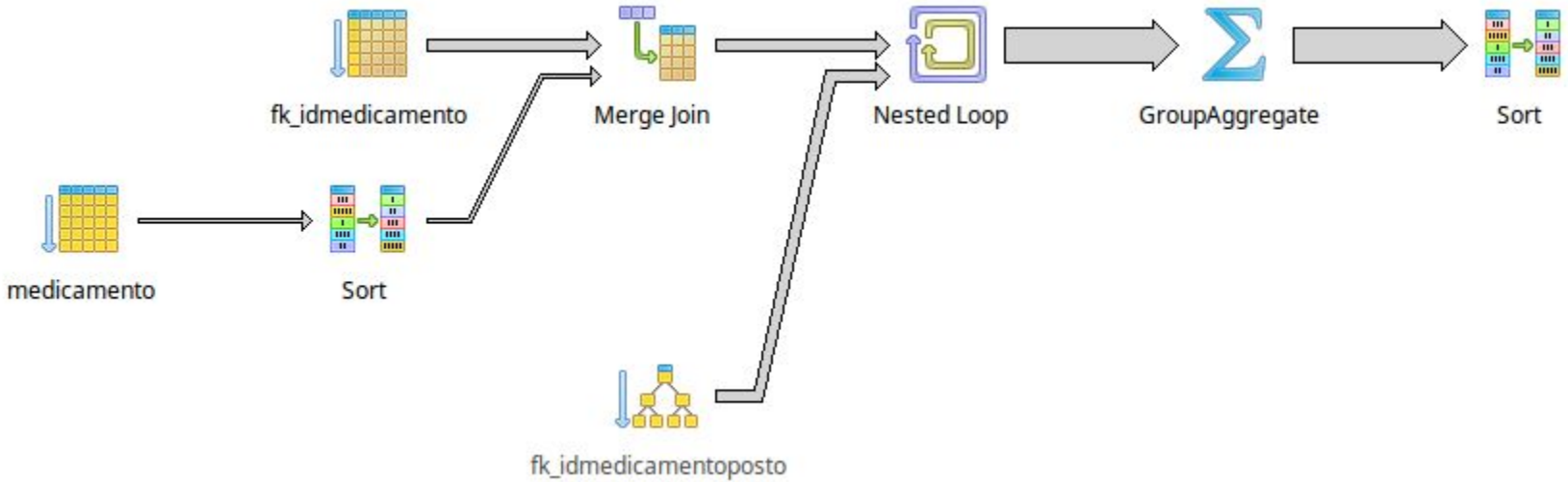
Detalhamento do resultado

	QUERY PLAN text
1	Sort (cost=37749.82..37749.87 rows=21 width=25)
2	Output: me.nome, (count(me.idmedicamento)), me.idmedicamento
3	Sort Key: (count(me.idmedicamento)) DESC
4	-> HashAggregate (cost=37749.10..37749.36 rows=21 width=25)
5	Output: me.nome, count(me.idmedicamento), me.idmedicamento
6	Group Key: me.idmedicamento
7	Filter: (count(me.idmedicamento) >= 25000)
8	-> Hash Join (cost=30.35..35150.35 rows=346500 width=25)
9	Output: me.idmedicamento, me.nome
10	Hash Cond: (so.idmedicamentoposto = mp.idmedicamentoposto)
11	-> Seq Scan on public.solicitacao so (cost=0.00..26030.00 rows=1500000 width=4)
12	Output: so.idsolicitacao, so.data_hora, so.quantidademedicamento, so.estadosolicitacao, so.idtitular, so.idmedicamentoposto
13	-> Hash (cost=27.46..27.46 rows=231 width=29)
14	Output: me.idmedicamento, me.nome, mp.idmedicamentoposto
15	-> Hash Join (cost=3.40..27.46 rows=231 width=29)
16	Output: me.idmedicamento, me.nome, mp.idmedicamentoposto
17	Hash Cond: (mp.idmedicamento = me.idmedicamento)
18	-> Seq Scan on public.medicamento_posto mp (cost=0.00..18.00 rows=1000 width=8)
19	Output: mp.idmedicamentoposto, mp.estadomedicamento, mp.quantidade, mp.datavencimento, mp.idposto, mp.idmedicamento
20	-> Hash (cost=3.14..3.14 rows=21 width=25)
21	Output: me.idmedicamento, me.nome
22	-> Seq Scan on public.medicamento me (cost=0.00..3.14 rows=21 width=25)
23	Output: me.idmedicamento, me.nome
24	Filter: ((me.nome)::text ~~* 'A%'::text)

Criação dos Indices

```
create index fk_idmedicamentoposto on solicitacao (idmedicamentoposto)
create index index_idmedicamentoposto on recebimento (idmedicamentoposto);
create index index_estadosolicitacao on solicitacao (estadosolicitacao);
create index index_pkmedicamento on medicamento (idmedicamento);
create index fk_idtitular on solicitacao (idmedicamentoposto)
```

Com Index


fk_idmedicamento
Merge Join
Nested Loop
GroupAggregate
Sort
medicamento
Sort
fk_idmedicamentoposto

Detalhamento do resultado

	QUERY PLAN text
1	Sort (cost=15129.00..15129.05 rows=21 width=25)
2	Output: me.nome, (count(me.idmedicamento)), me.idmedicamento
3	Sort Key: (count(me.idmedicamento)) DESC
4	-> HashAggregate (cost=15128.28..15128.54 rows=21 width=25)
5	Output: me.nome, count(me.idmedicamento), me.idmedicamento
6	Group Key: me.idmedicamento
7	Filter: (count(me.idmedicamento) >= 25000)
8	-> Nested Loop (cost=3.83..12529.53 rows=346500 width=25)
9	Output: me.idmedicamento, me.nome
10	-> Hash Join (cost=3.40..27.46 rows=231 width=29)
11	Output: me.idmedicamento, me.nome, mp.idmedicamentoposto
12	Hash Cond: (mp.idmedicamento = me.idmedicamento)
13	-> Seq Scan on public.medicamento_posto mp (cost=0.00..18.00 rows=1000 width=8)
14	Output: mp.idmedicamentoposto, mp.estadomedicamento, mp.quantidade, mp.datavencimento, mp.idposto, mp.idmedicamento
15	-> Hash (cost=3.14..3.14 rows=21 width=25)
16	Output: me.idmedicamento, me.nome
17	-> Seq Scan on public.medicamento me (cost=0.00..3.14 rows=21 width=25)
18	Output: me.idmedicamento, me.nome
19	Filter: ((me.nome)::text ~~* 'A%'::text)
20	-> Index Only Scan using fk_idmedicamentoposto on public.solicitacao so (cost=0.43..39.12 rows=1500 width=4)
21	Output: so.idmedicamentoposto
22	Index Cond: (so.idmedicamentoposto = mp.idmedicamentoposto)

Teste de Performance

Query 5 - Medicamentos mais Solicitados por número de solicitações		
Núm. Vezes	**Planning Time (ms)**	**Execution Time (ms)**
1	0,788	345,552
2	0,941	326,235
3	0,488	256,527
4	0,902	257,706
5	0,596	262,948
Média	**0,743**	**289,7936**

Com Index

Query 5 - Medicamentos mais Solicitados por número de solicitações		
Núm. Vezes	**Planning Time (ms)**	**Execution Time (ms)**
1	0,768	555,720
2	0,611	484,942
3	0,391	445,855
4	0,404	453,753
5	1,121	460,048
Média	**0,659**	**480,0636**

Sem Index

Github

- https://github.com/lukasg18/Topicos-Trabalho-BD2