

---

---

# Banco de Dados II

## Trabalho Final

— Amanda Oliveira, Wolgan Ens —

---

---

# Agenda

- Características do PostgreSQL
- Ambiente experimental
- Cenário de testes
- Consultas utilizadas
- Metodologia
- Gráficos obtidos
- Resultados

# Características do PostgreSQL

O PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional, desenvolvido na University of California at Berkeley, com versão estável desde 1996. Possui uma comunidade de 33 desenvolvedores ativa, distribuída mundialmente, encarregada da sua evolução e resolução de problemas encontrados.

- Desenvolvido na Linguagem de programação C
- Executado nos principais sistemas operacionais: Linux, Unix, Windows
- Possui as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade)

# Características do PostgreSQL

Possui recursos comuns aos tradicionais SGBDs comerciais:

- Índices (index) - (B-Tree, Hash, Gist, Sp-Gist, Gin)
- Gatilhos (triggers)
- Visões (views)
- Chaves estrangeiras (foreign keys)
- Controle de concorrência



Figura 1 - [3]

Atualmente o PostgreSQL é conhecido como o SGBD de código aberto mais avançado e com o mais completo conjunto de recursos.

# Ambiente experimental

Os experimentos foram realizados em um computador com:

- Processador Intel Core i5-4200U a 1.6 GHz
- 4GB de memória SDRAM
- Disco Rígido de 500GB
- Sistema Operacional: Windows 8.1

Foi usada a versão 9.5.3 do SGBD PostgreSQL.

# Cenário de testes

Foram feitas cinco tabelas distintas com a mesma estrutura, variando na quantidade de registros.

```
1  create table table1(  
2      ip serial primary key,  
3      "is" integer not null,  
4      ic1 integer not null,  
5      ic2 integer not null,  
6      ni integer not null  
7  );  
8  create table table2(  
9      ip serial primary key,  
10     "is" integer not null,  
11     ic1 integer not null,  
12     ic2 integer not null,  
13     ni integer not null  
14 );
```

```
15 create table table3(  
16     ip serial primary key,  
17     "is" integer not null,  
18     ic1 integer not null,  
19     ic2 integer not null,  
20     ni integer not null  
21 );  
22 create table table4(  
23     ip serial primary key,  
24     "is" integer not null,  
25     ic1 integer not null,  
26     ic2 integer not null,  
27     ni integer not null  
28 );
```

```
29 create table table5(  
30     ip serial primary key,  
31     "is" integer not null,  
32     ic1 integer not null,  
33     ic2 integer not null,  
34     ni integer not null  
35 );
```

# Cenário de testes

Para o cenário de experimento foi exigido que a estrutura de banco de dados possuísse:

- um atributo com índice primário (ip),
- um atributo com índice secundário (is),
- dois atributos com índice secundário composto (ic1, ic2) e
- um atributo sem índices (ni).

Todos esses atributos deveriam ser numéricos inteiros.

# Cenário de testes

```
36 create index primary_index_ip on table1 (ip);
37 create index secondary_index_is on table1 ("is");
38 create index multicolumn_index_ic1_ic2 on table1 (ic1,ic2);
39
40 create index primary_index_ip_t2 on table2 (ip);
41 create index secondary_index_is_t2 on table2 ("is");
42 create index multicolumn_index_ic1_ic2_t2 on table2 (ic1,ic2);
43
44 create index primary_index_ip_t3 on table3 (ip);
45 create index secondary_index_is_t3 on table3 ("is");
46 create index multicolumn_index_ic1_ic2_t3 on table3 (ic1,ic2);
47
48 create index primary_index_ip_t4 on table4 (ip);
49 create index secondary_index_is_t4 on table4 ("is");
50 create index multicolumn_index_ic1_ic2_t4 on table4 (ic1,ic2);
51
52 create index primary_index_ip_t5 on table5 (ip);
53 create index secondary_index_is_t5 on table5 ("is");
54 create index multicolumn_index_ic1_ic2_t5 on table5 (ic1,ic2);
```



# Cenário de testes

- A primeira versão deveria conter 100 registros.
- A segunda versão deveria conter 1.000 registros.
- A terceira versão deveria conter 10.000 registros.
- A quarta versão deveria conter 100.000 registros.
- A quinta versão deveria conter 1.000.000 de registros.

```
56 insert into table1 values ( generate_series(1,100),ceil(random() * 100),ceil(random() * 100),ceil(random() * 100),ceil(random() * 100));
57 insert into table2 values ( generate_series(1,1000),ceil(random() * 1000),ceil(random() * 1000),ceil(random() * 1000),ceil(random() * 1000));
58 insert into table3 values ( generate_series(1,10000),ceil(random() * 10000),ceil(random() * 10000),ceil(random() * 10000),ceil(random() * 10000));
59 insert into table4 values ( generate_series(1,100000),ceil(random() * 100000),ceil(random() * 100000),ceil(random() * 100000),ceil(random() * 100000));
60 insert into table5 values ( generate_series(1,1000000),ceil(random() * 1000000),ceil(random() * 1000000),ceil(random() * 1000000),ceil(random() * 1000000));
```

# Consultas utilizadas

```
1 SELECT ip          FROM table5 WHERE ip      = 70;
2 SELECT "is"        FROM table5 WHERE "is"    = 70;
3 SELECT ip          FROM table5 WHERE ip      < 70;
4 SELECT "is"        FROM table5 WHERE "is"    < 70;
5 SELECT ip          FROM table5 WHERE ip      > 70;
6 SELECT "is"        FROM table5 WHERE "is"    > 70;
7 SELECT ip, "is"    FROM table5 WHERE ip      = 70 OR "is" = 70;
8 SELECT ip, ni      FROM table5 WHERE ip      = 70 OR ni = 70;
9 SELECT ip, "is"    FROM table5 WHERE ip      < 70 OR "is" < 70;
10 SELECT ip, ni     FROM table5 WHERE ip      < 70 OR ni < 70;
11 SELECT ip, "is"   FROM table5 WHERE ip      > 70 OR "is" > 70;
12 SELECT ip, ni     FROM table5 WHERE ip      > 70 OR ni > 70;
```

# Metodologia

A metodologia utilizada para os testes apresenta como principal característica, a execução de cada experimento, 10 vezes. Para calcular o resultado final entre os 10 experimentos, é aplicada a média aritmética, excluindo o melhor e pior resultado.

Foram executados os comandos `\r` e `DISCARD ALL` para limpar o buffer de consulta e limpar a sessão.

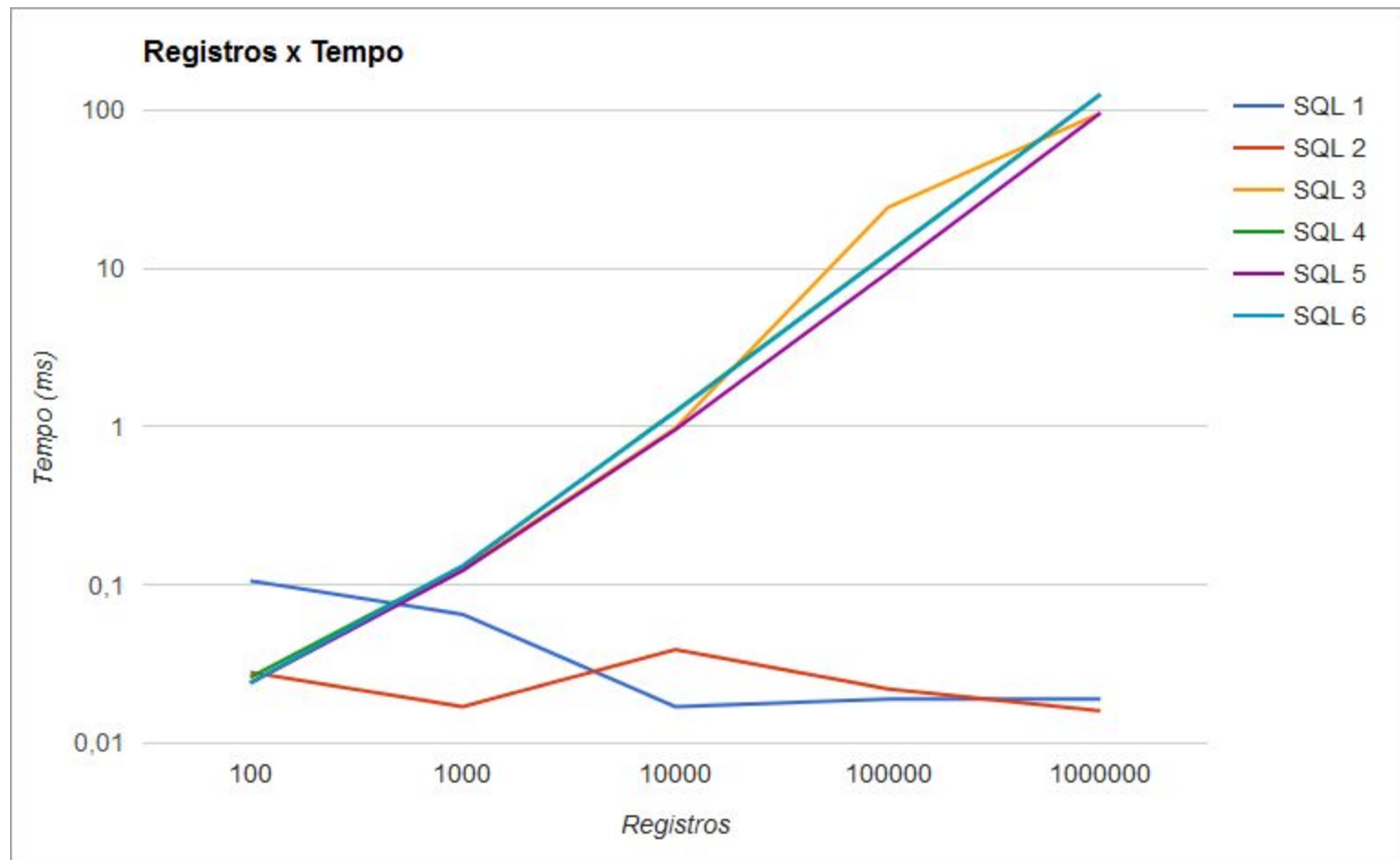










Gráfico 1

# Resultados

As consultas SQL1 e SQL2 demonstram inicialmente uma mudança no comportamento devido as mudanças no plano de execução nas tabelas iniciais. Ambas iniciam utilizando uma busca sequencial (devido ao número pequeno de registros?) e a partir da segunda tabela passam a fazer uso dos índices em seus campos

# Resultados

<b>Seq Scan</b> on public.table1  Output: ip Filter: (table1.ip = 500) Rows Removed by Filter: 100 (cost=0.00..2.25 rows=1 width=4) (actual time=0.026..0.026 rows=0 loops=1)	<b>Index Only Scan</b> using primary_index_ip_t2 on public.table2  Output: ip Index Cond: (table2.ip = 500) Heap Fetches: 1 (cost=0.28..8.29 rows=1 width=4) (actual time=0.034..0.036 rows=1 loops=1)	<b>Index Only Scan</b> using primary_index_ip_t3 on public.table3  Output: ip Index Cond: (table3.ip = 5000) Heap Fetches: 1 (cost=0.29..8.30 rows=1 width=4) (actual time=0.013..0.014 rows=1 loops=1)	<b>Index Only Scan</b> using primary_index_ip_t4 on public.table4  Output: ip Index Cond: (table4.ip = 5000) Heap Fetches: 1 (cost=0.29..8.31 rows=1 width=4) (actual time=0.022..0.023 rows=1 loops=1)
 public.table1	 primary_index_ip_t2	 primary_index_ip_t3	 primary_index_ip_t4
 Planning time: 0.095 ms	 Planning time: 0.158 ms	 Planning time: 0.089 ms	 Planning time: 0.120 ms

# Resultados

## Seq Scan

on public.table1

Output: "is" Filter: (table1."is" = 50) Rows Removed by Filter: 99  
(cost=0.00..2.25 rows=1 width=4)  
(actual time=0.023..0.026 rows=1 loops=1)



public.table1



Planning time: 0.091 ms

## Index Only Scan

using secondary\_index\_is\_t2 on public.table2

Output: "is" Index Cond: (table2."is" = 500) Heap Fetches: 2  
(cost=0.28..8.29 rows=1 width=4)  
(actual time=0.017..0.020 rows=2 loops=1)



secondary\_index\_is\_t2



Planning time: 0.095 ms

## Bitmap Index Scan

on secondary\_index\_is\_t3

Index Cond: (table3."is" = 500)  
(cost=0.00..4.30 rows=2 width=0)  
(actual time=0.054..0.054 rows=2 loops=1)



secondary\_index\_is\_t3



public.table3



Planning time: 0.133 ms

## Bitmap Index Scan

on secondary\_index\_is\_t4

Index Cond: (table4."is" = 500)  
(cost=0.00..4.31 rows=2 width=0)  
(actual time=6.962..6.962 rows=1 loops=1)



secondary\_index\_is\_t4



public.table4



Planning time: 0.132 ms

## Bitmap Heap Scan

on public.table4

Output: "is" Recheck Cond: (table4."is" = 500) Heap Blocks: exact=1  
(cost=4.31..12.00 rows=2 width=4)  
(actual time=6.990..6.991 rows=1 loops=1)

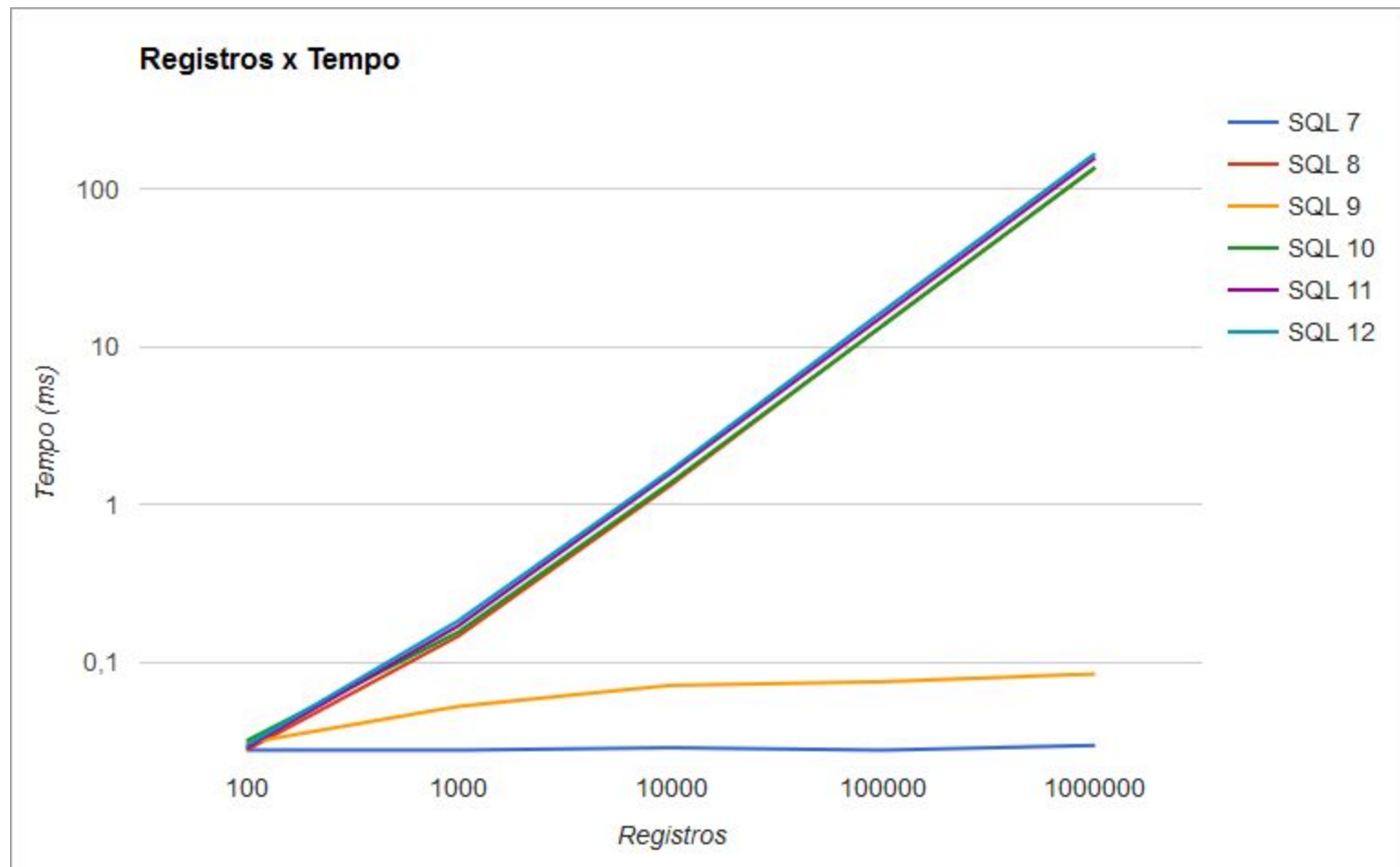


Gráfico 2



# Resultados

## Seq Scan on table1

Filter: ((ip = 50) OR ("is" = 50)) Rows Removed by Filter: 99  
(cost=0.00..2.50 rows=2 width=8)  
(actual time=0.056..0.075 rows=1 loops=1)



table1



Planning time: 0.243 ms

## Bitmap Index Scan on primary\_index\_ip\_t2

Index Cond: (ip = 500)  
(cost=0.00..4.28 rows=1 width=0)  
(actual time=0.017..0.017 rows=1 loops=1)



primary\_index\_ip\_t2



secondary\_index\_is\_t2

## Bitmap Index Scan on secondary\_index\_is\_t2

Index Cond: ("is" = 500)  
(cost=0.00..4.28 rows=1 width=0)  
(actual time=0.013..0.013 rows=1 loops=1)

## BitmapOr

(cost=8.57..8.57 rows=2 width=0)  
(actual time=0.035..0.035 rows=0 loops=1)



BitmapOr

## Bitmap Heap Scan on table2

Recheck Cond: ((p = 500) OR ("is" = 500)) Heap Blocks: exact=2  
(cost=8.57..13.39 rows=2 width=8)  
(actual time=0.047..0.051 rows=2 loops=1)



table2



Planning time: 0.196 ms

## Bitmap Index Scan on primary\_index\_ip\_t3

Index Cond: (ip = 5000)  
(cost=0.00..4.29 rows=1 width=0)  
(actual time=0.019..0.019 rows=1 loops=1)



primary\_index\_ip\_t3



secondary\_index\_is\_t3

## Bitmap Index Scan on secondary\_index\_is\_t3

Index Cond: ("is" = 5000)  
(cost=0.00..4.30 rows=2 width=0)  
(actual time=0.013..0.013 rows=2 loops=1)

## BitmapOr

(cost=8.59..8.59 rows=3 width=0)  
(actual time=0.034..0.034 rows=0 loops=1)



BitmapOr

## Bitmap Heap Scan on table3

Recheck Cond: ((p = 5000) OR ("is" = 5000)) Heap Blocks: exact=3  
(cost=8.59..18.69 rows=3 width=8)  
(actual time=0.046..0.052 rows=3 loops=1)

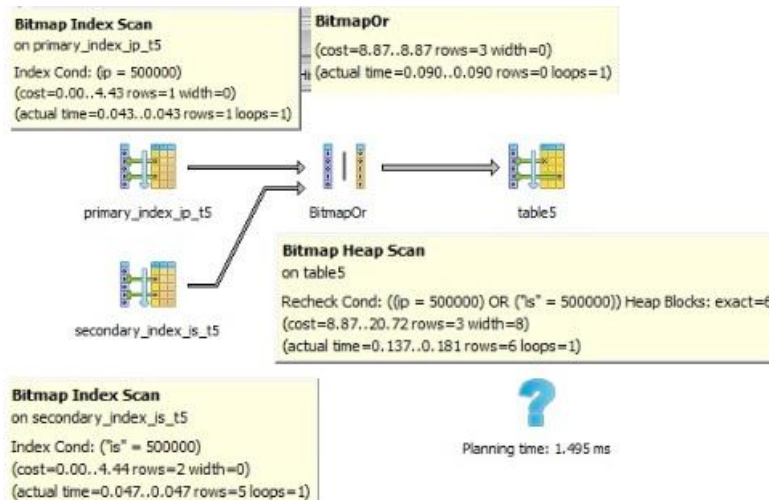
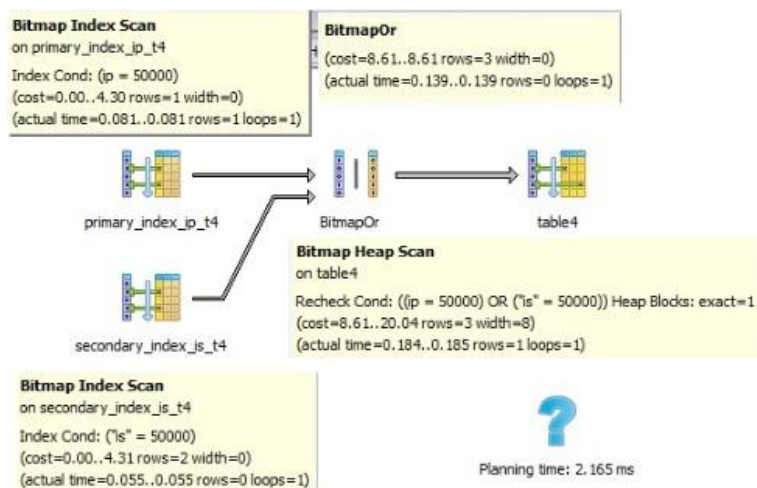


table3



Planning time: 0.201 ms

# Resultados



# Resultados

## Seq Scan

on table1

Filter: ((ip < 50) OR ("is" < 50)) Rows Removed by Filter: 21  
(cost=0.00..2.50 rows=78 width=8)  
(actual time=0.025..0.050 rows=79 loops=1)



table1



Planning time: 1.191 ms

## Seq Scan

on table2

Filter: ((ip < 500) OR ("is" < 500)) Rows Removed by Filter: 265  
(cost=0.00..22.00 rows=742 width=8)  
(actual time=0.090..0.436 rows=735 loops=1)



table2



Planning time: 2.054 ms

## Seq Scan

on table3

Filter: ((ip < 5000) OR ("is" < 5000)) Rows Removed by Filter: 2468  
(cost=0.00..214.00 rows=7554 width=8)  
(actual time=0.092..5.198 rows=7532 loops=1)



table3



Planning time: 1.663 ms

# Resultados

## Seq Scan

on table4

Filter: ((ip < 50000) OR ("is" < 50000)) Rows Removed by Filter: 25114  
(cost=0.00..2137.00 rows=75157 width=8)  
(actual time=0.088..40.369 rows=74886 loops=1)



table4



Planning time: 1.248 ms

## Seq Scan

on table5

Filter: ((ip < 500000) OR ("is" < 500000)) Rows Removed by Filter: 249670  
(cost=0.00..21370.00 rows=751955 width=8)  
(actual time=0.100..386.736 rows=750330 loops=1)



table5



Planning time: 1.492 ms

# Referências:

[1] “Comunidade Brasileira de PostgreSQL” - <https://www.postgresql.org.br/sobre>, acessado em 06/2014.

[2] Pedrozo, Wendel Góes. *“Arquitetura para seleção de índices em banco de dados relacionais, utilizando abordagem baseada em custos do otimizador”*, Curitiba, 2008.

[3] [https://www.runabove.com/images/new/2015/postgresql\\_1.png](https://www.runabove.com/images/new/2015/postgresql_1.png)