# Alquimia com buscas textuais!


Métodos para acelerar query's que utilizam colunas do tipo texto

**Alquimista 1:** Antônio Carlos D.
**Alquimista 2:** Elimar Macena

# 1º: Opte por LIKE invés de ILIKE

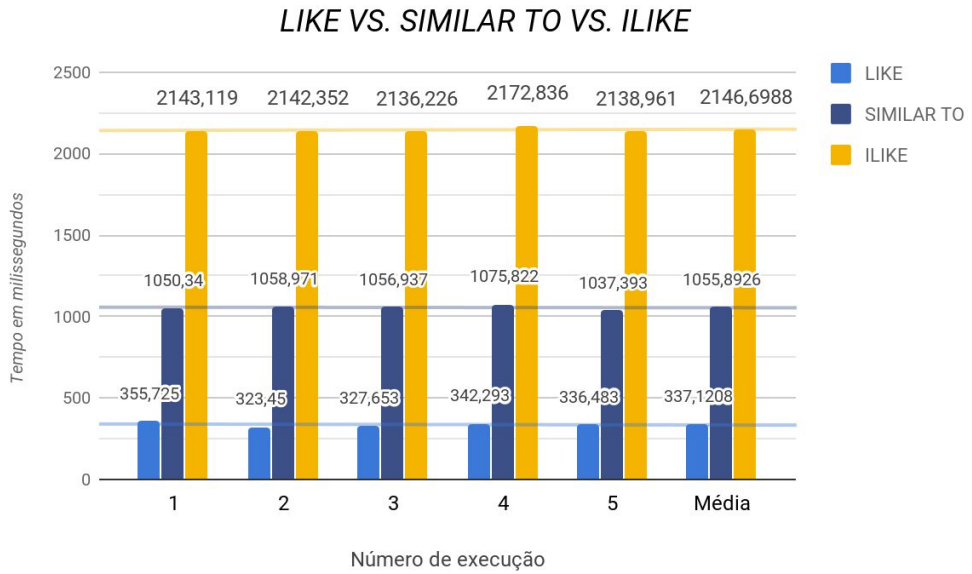
Sempre que possível, use LIKE para trabalhar com buscas textuais. *Don't LIKE ILIKE*, SIMILAR TO ou expressões regulares.

**LIKE** equivale a ~~
**ILIKE** equivale a ~~*

**Quantidade registros:** 3 000 489

**Query:**

```
EXPLAIN ANALYZE
SELECT * FROM contato
WHERE descricao [OPERADOR] 'maria%';
```



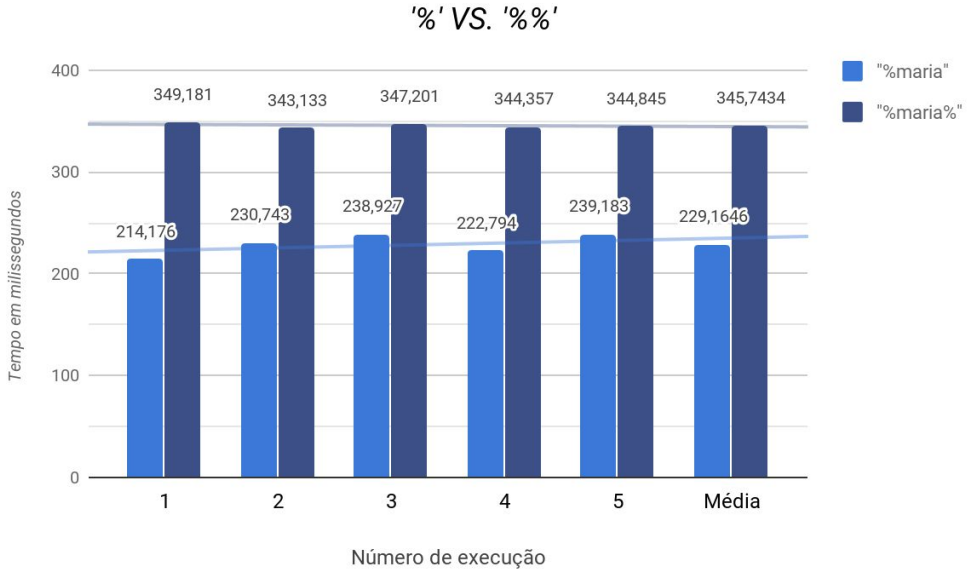
# 2º: Somente use ‘%’ quando necessário

Ao usar **LIKE** ou **ILIKE**, evite colocar o operador ‘%’ sem verificar sua necessidade.

**Quantidade registros:** 1 500 000

**Query:**

```
EXPLAIN ANALYZE
SELECT * FROM usuario
WHERE email LIKE '%maria%';
```



# 3º: Use índices, use trigrama, seja feliz!

**Problema:**
O tipo B-Tree de índice não apresenta benefício ao utilizar operadores **LIKE** ou **ILIKE**.

Para contornar tal empecilho utilize índices do tipo GIN ou GiST.

Para criar os índices acima em campos textuais (**CHAR** e **VARCHAR**) utilize a extensão de **TRIGRAMAS** do Postgre.

**“Santa Query! O que seria um trigrama?”**
*“Um trigrama é um grupo de três caracteres consecutivos retirados de uma string.”* ~PostgreSQL

**“Tá, belê, e quais as magias têm nessa extensão?”**
*“O módulo fornece funções e operadores para determinar a similaridade de texto alfanumérico com base na correspondência de trigramas, bem como classes de operadores de índice que suportam a pesquisa rápida de cadeias semelhantes.”* ~PostgreSQL

# E a magia, como acontece?

**Exemplo de trigramas:**

Palavra: “word”

Trigramas: { “ w”, “ wo”, “wor”, “ord”, "rd" }

**“Tá, mas e o índice?”**

Segundo a doc. do Postgre, o pg_trgm fornece meios criar índices rápidos para consultas textuais usando operadores **LIKE** e **ILIKE**, graças às funções que calculam similaridade e geram os trigramas.

**Exemplo de busca usando o módulo:**

Palavra buscada: “maria”

Texto 2 - alvo: “maria_dev@icloud.com”

Trigramas 1: {" m", " ma", ari, "ia ", mar, ria}

Trigramas 2:

{" c", " d", " i", " m", " co", " de", " ic", " ma", ari, clo, com, dev, "ev ", "ia ", icl, lou, mar, "om ", oud, ria, "ud "}

Como o texto 2 contém trigramas que são iguais ou que englobam os do texto buscado, o endereço de email será incluso nos resultados da consulta.

# Será que houve diferença?!

Speedup médio aproximado: 66.80 vezes (6680%) .

**Query 1 - Sem otimização alguma:**

```
EXPLAIN ANALYZE
SELECT * FROM usuario
WHERE email ILIKE '%maria%';
```

**Query 2:**

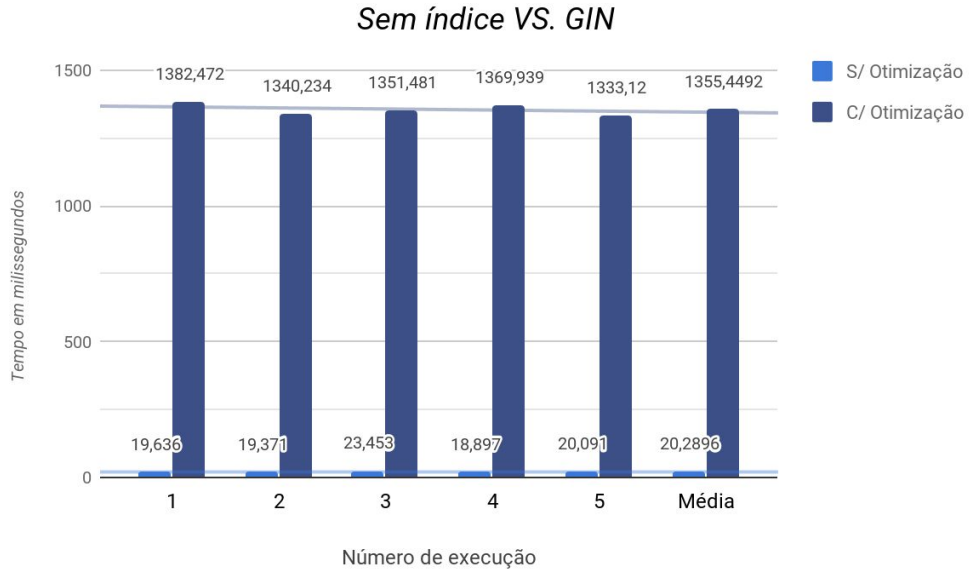
```
EXPLAIN ANALYZE
SELECT * FROM usuario
WHERE email LIKE '%maria%';
```

**Para criar a extensão:**

```
CREATE EXTENSION pg_trgm;
```

**Para criar o índice:**

```
CREATE INDEX nome_do_indice ON nome_tabela
USING gin (nome_da_coluna  gin_trgm_ops);
```

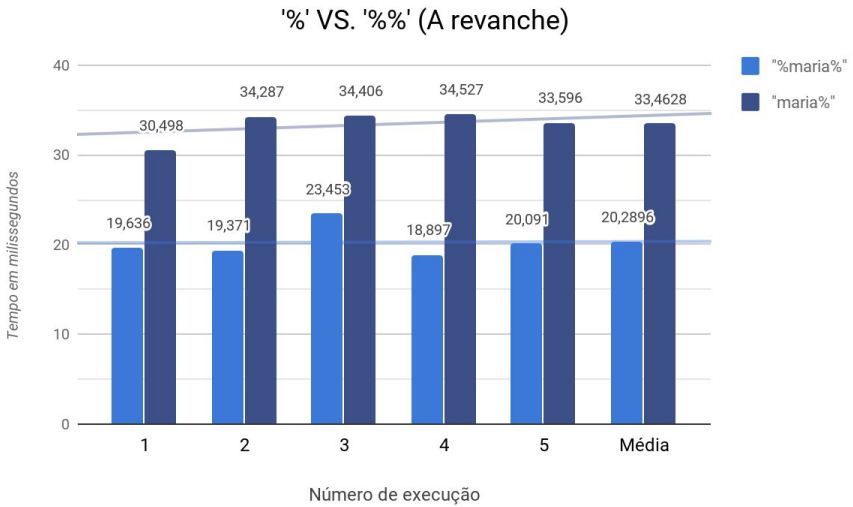


# Paradoxo dos ‘%’

Ao fazer uso do módulo pg_trgm para indexar as colunas textuais, algo curioso ocorre. As buscas feitas com o símbolo ‘%’ no início E no fim do texto são mais rápidas do que as feitas com apenas um ‘%’ no fim do texto.

**“Tu mentiu no começo dos slides?”**

Um pouquinho. “*Quanto mais* trigramas puderem ser extraídos da expressão regular, *mais eficaz será a pesquisa de índice*. Ao contrário das pesquisas baseadas na árvore B, a cadeia de pesquisa não precisa ser ancorada à esquerda.” ~PostgreSQL



# REFERÊNCIAS

Pattern Matching: https://www.postgresql.org/docs/9.0/functions-matching.html

pg_trgm: https://www.postgresql.org/docs/10/pgtrgm.html