



★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



Pesquisa MySQL por similaridade entre 2 colunas diferentes



Kaléu Caminha · [Follow](#)

3 min read · Feb 15, 2019



Share



More

Recentemente, precisei comparar endereços em uma base de dados para identificar endereços muito parecidos, como por exemplo “Avenida João da Silva” e “Av. João da Silva”.

As Tentativas

Inicialmente, fui procurar por funções com comportamento parecido ao da função *similar_text* do PHP que retorna o percentual de similaridade entre duas strings.

Cheguei na função *Soundex* do MySQL, que retorna uma string referente ao som da palavra em inglês. Não resolveria, porque minha necessidade é em PT_br e tenho problemas além do som, como pontuação e erros de grafia.

Pensei que poderia usar o índice *fulltext*, mas também não daria certo porque as pesquisas *fulltext* funcionam com uma comparação com uma string literal, mas minha necessidade é comparar duas colunas diferentes.

A Solução

Após algumas tentativas, encontrei no Stack Overflow o algoritmo *levenshtein* (usado frequentemente para o tipo de problema que eu tinha) implementado como uma function do MySQL.

Aparentemente a empresa Artful Software Development, que lançou o ebook Get It Done With MySQL 5&Up, disponibilizou essa solução no site deles com a função

pronta para ser usada:

<http://www.artfulsoftware.com/infotree/qrytip.php?id=552> (propositalmente, não vou copiar o código aqui, para dar esse crédito da visita ao site da Artful).

São duas funções disponibilizadas: 1) **levenshtein**, que calcula o número de operações necessárias para transformar uma string em outra e a 2) **levenshtein_ratio**, que usa a função acima para calcular um percentual de similaridade entre as duas strings, exatamente o que eu precisava.

O Exemplo

Segue abaixo uma reprodução em pequena escala do problema que a função resolveu. Destaco que não fiz testes de performance. Trabalhei com consultas na casa de centenas de linhas apenas.

Para o exemplo, considere que a empresa está fazendo uma migração de sistemas e precisa verificar a base de endereços dos clientes com uma base antiga. Para este exemplo fictício, vamos usar a seguinte estrutura:

```
create database exemplo_similaridade;

create table enderecos_cliente (
    id integer not null primary key,
    id_cliente integer not null,
    endereco varchar(200) not null
) ENGINE=innodb CHARSET=utf8 COLLATE=utf8_general_ci;

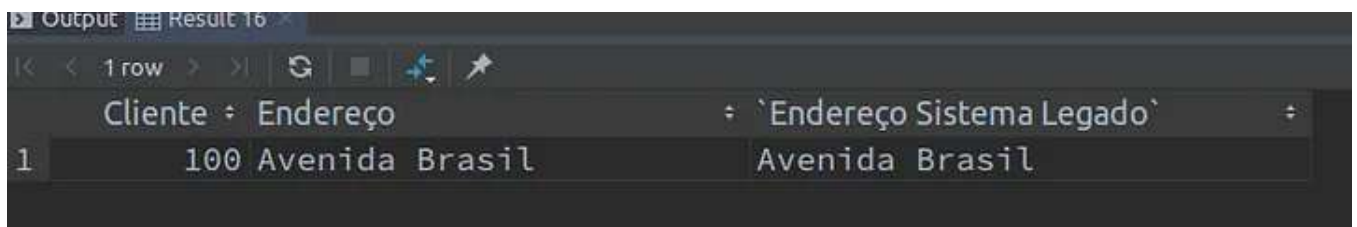
insert into enderecos_cliente values
    (1, 100, 'Avenida Brasil'),
    (2, 100, 'Avenida Santo Antônio'),
    (3, 200, 'Rua Richard Filho'),
    (4, 200, 'Servidão Arlindo Silva Cruz');

create table enderecos_cliente_sistema_legado (
    id integer not null primary key,
    id_cliente integer not null,
    endereco varchar(200) not null
) ENGINE=innodb CHARSET=utf8 COLLATE=utf8_general_ci;

insert into enderecos_cliente_sistema_legado values
    (1000, 100, 'Avenida Brasil'),
    (2000, 100, 'Avenida Santo Antônio'),
    (3000, 200, 'R. Richard Filho'),
    (4000, 200, 'Servidão Aurélio Machado');
```

Considerando que o id do cliente está corretamente identificado, se tentarmos pelo método tradicional encontrar endereços iguais, somente a “Avenida Brasil” será retornada. Os endereços da “Avenida Santo Antônio” e “Rua Richard Filho” que ao olho humano são certamente iguais, não serão identificados:

```
select
    e.id_cliente as 'Cliente',
    e.endereco as 'Endereço',
    e_legado.endereco as 'Endereço Sistema Legado'
from enderecos_cliente e
    inner join enderecos_cliente_sistema_legado e_legado
    on e.id_cliente = e_legado.id_cliente
    and e.endereco = e_legado.endereco;
```

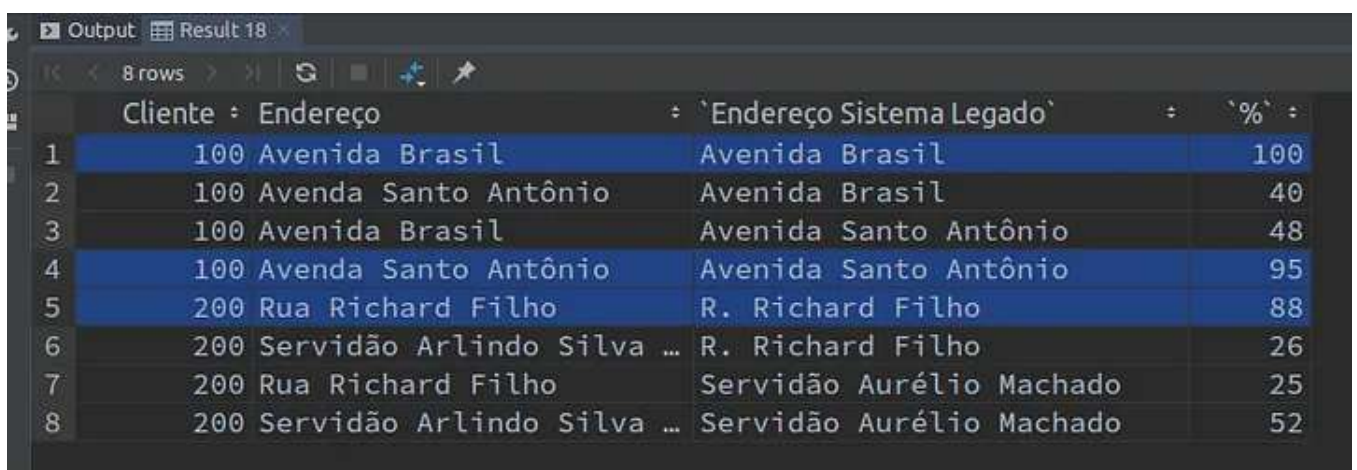


Output Result 16

	Cliente : Endereço	: 'Endereço Sistema Legado'	:
1	100 Avenida Brasil	Avenida Brasil	

Agora, vamos dar uma olhada no retorno da função de similaridade:

```
select
    e.id_cliente as 'Cliente',
    e.endereco as 'Endereço',
    e_legado.endereco as 'Endereço Sistema Legado',
    levenshtein_ratio(e.endereco, e_legado.endereco) as '%'
from enderecos_cliente e
    inner join enderecos_cliente_sistema_legado e_legado
    on e.id_cliente = e_legado.id_cliente;
```



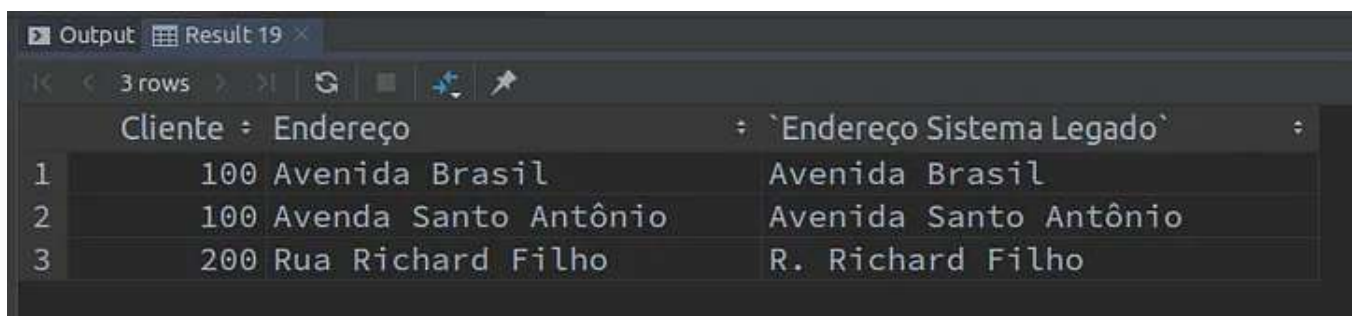
Output Result 18

	Cliente : Endereço	: 'Endereço Sistema Legado'	:	'%'	:
1	100 Avenida Brasil	Avenida Brasil		100	
2	100 Avenda Santo Antônio	Avenida Brasil		40	
3	100 Avenida Brasil	Avenida Santo Antônio		48	
4	100 Avenda Santo Antônio	Avenida Santo Antônio		95	
5	200 Rua Richard Filho	R. Richard Filho		88	
6	200 Servidão Arlindo Silva ...	R. Richard Filho		26	
7	200 Rua Richard Filho	Servidão Aurélio Machado		25	
8	200 Servidão Arlindo Silva ...	Servidão Aurélio Machado		52	

Perceba que claramente, os percentuais das strings que representam de fato o mesmo endereço ficam muito “destoantes” das demais.

Com essa estratégia é possível, dependendo do domínio, estabelecer um corte neste percentual do que vai ser considerado igual ou no mínimo marcar estes endereços para uma posterior verificação humana já muito bem direcionada:

```
select
  e.id_cliente as 'Cliente',
  e.endereco as 'Endereço',
  e_legado.endereco as 'Endereço Sistema Legado'
from enderecos_cliente e
  inner join enderecos_cliente_sistema_legado e_legado
    on e.id_cliente = e_legado.id_cliente
where levenshtein_ratio(e.endereco, e_legado.endereco) > 80;
```



The screenshot shows a database query result with three rows. The columns are 'Cliente', 'Endereço', and 'Endereço Sistema Legado'. The rows show the following data:

	Cliente	Endereço	Endereço Sistema Legado
1	100	Avenida Brasil	Avenida Brasil
2	100	Avenda Santo Antônio	Avenida Santo Antônio
3	200	Rua Richard Filho	R. Richard Filho

Abraços e caso você conheça outras técnicas, só adicionar aos comentários.

MySQL

Sql

Similaridade



Follow

Written by Kaléu Caminha

76 Followers · 116 Following

Responses (1)



Marco Aurélio Barbiero

What are your thoughts?



Fagner Gonçalves

Mar 13, 2021



Parabéns pelo artigo, será muito útil !! Realmente a gente sofre um pouco com essas comparações mas esta função parece resolver bastante o caso.



1

[Reply](#)

More from Kaléu Caminha

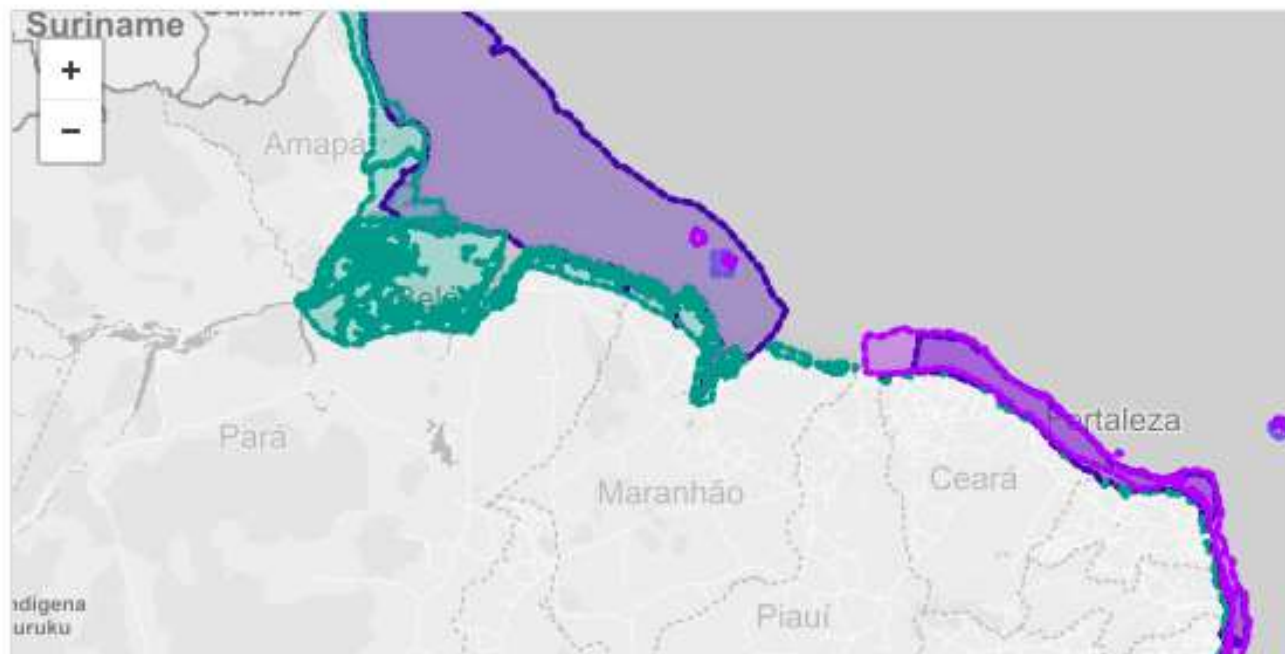


Kaléu Caminha

Os Diferentes Tipos de Testes

Edit: Este material fazia parte de um curso de testes em PHP do qual estou dividindo os materiais e publicando isoladamente por aqui. ;-)

Jan 22, 2018 🖱 5



In Datapedia Tech by Kaléu Caminha

Nota Técnica: Mapa de Zona Costeira e Marítima

A visualização de Zonas Costeiras do Datapedia apresenta Áreas Prioritárias para Conservação mapeadas pelo Ministério do Meio Ambiente que...

Aug 30, 2018 🖱 3



 Kaléu Caminha

Referências em Transparência Dados de Eleições

O Brasil (TSE) disponibiliza uma grande quantidade de dados sobre as eleições do país, desde o número de votos de cada candidato por zona...

Feb 26, 2018



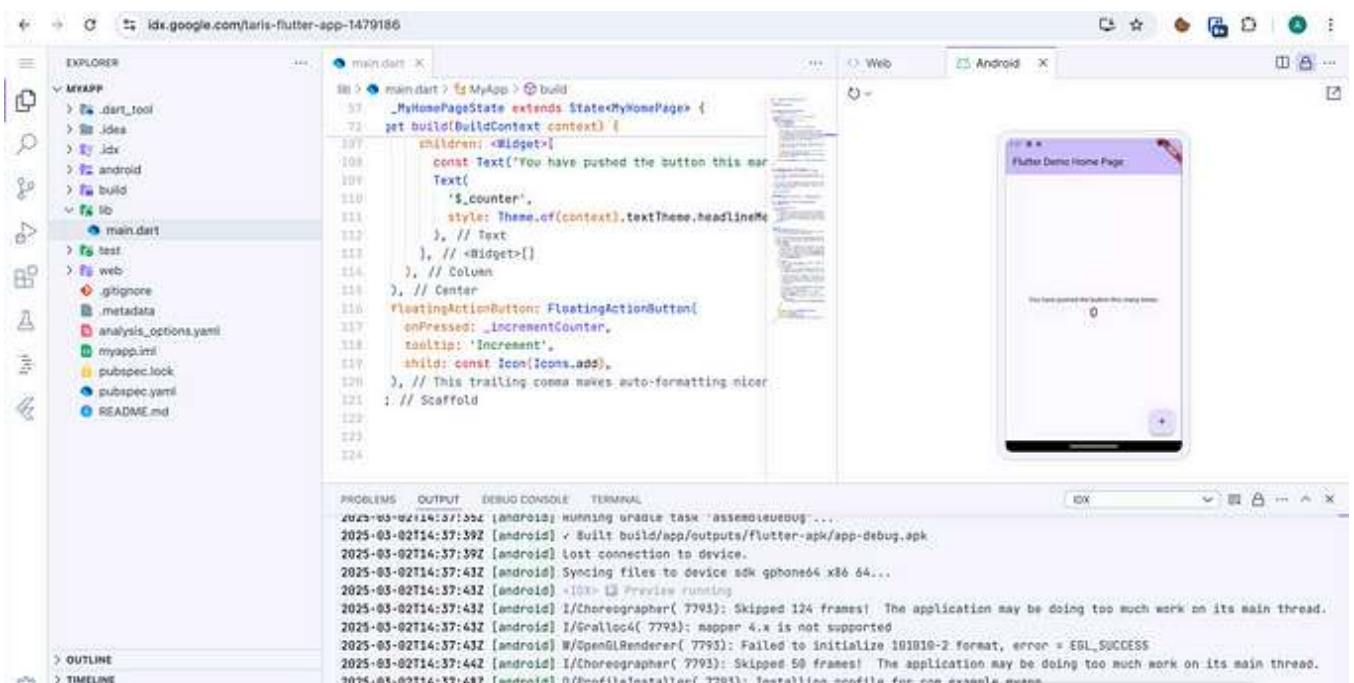
 In Datapedia Tech by Kaléu Caminha

Nota Técnica: Visualização de Unidades de Conservação

Logo na sequência do lançamento da visualização dos Rios nas cidades brasileiras, o Datapedia acaba de lançar a visualização de Unidades de...

[See all from Kaléu Caminha](#)

Recommended from Medium



 In Coding Beauty by Tari Ibaba

This new IDE from Google is an absolute game changer

This new IDE from Google is seriously revolutionary.

★ Mar 11 🤝 3.8K 💬 206



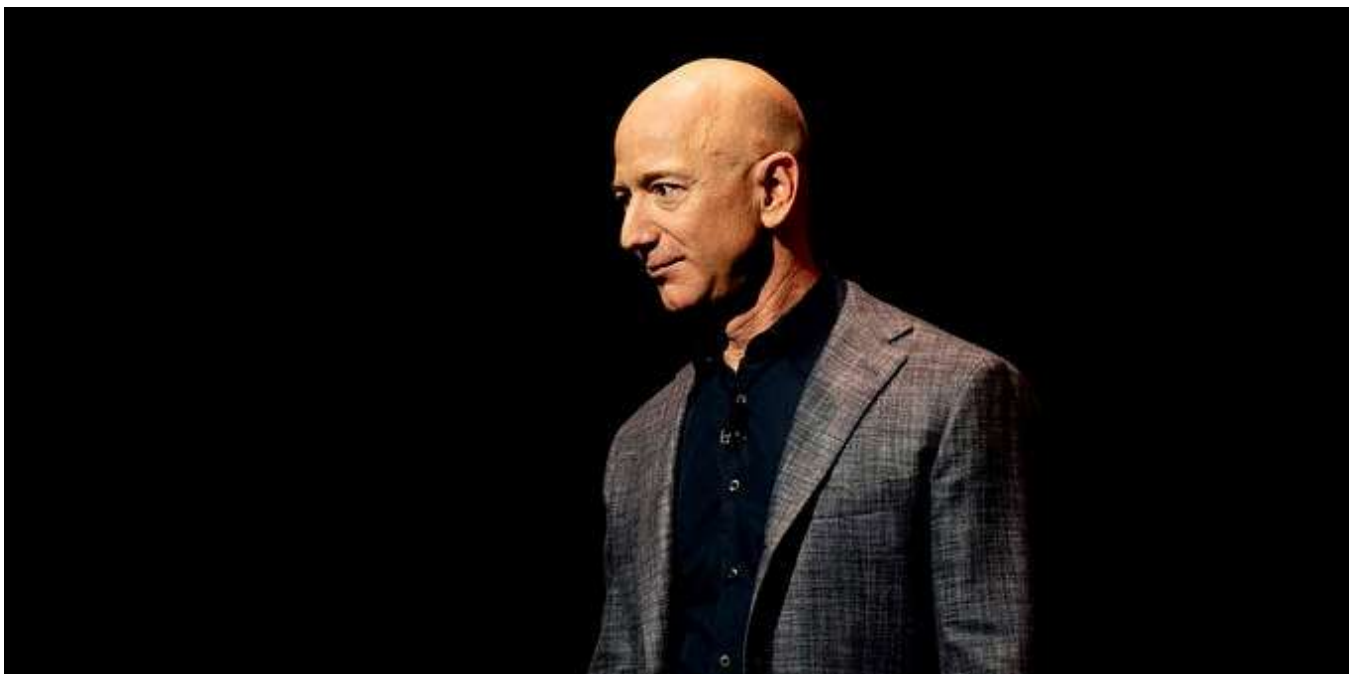
 In Hack the Stack by Coders Stop

9 Database Optimization Tricks SQL Experts Are Hiding From You

Most developers learn enough SQL to get by—SELECT, INSERT, UPDATE, DELETE, and maybe a few JOINS. They might even know how to create...

🌟 Mar 27 🖱 286 💬 7

🔖⁺ ...



 Jessica Stillman

Jeff Bezos Says the 1-Hour Rule Makes Him Smarter. New Neuroscience Says He's Right

Jeff Bezos's morning routine has long included the one-hour rule. New neuroscience says yours probably should too.

🌟 Oct 30, 2024 🖱 26K 💬 766



 Sebastian Carlos

Fired From Meta After 1 Week: Here's All The Dirt I Got

This is not just another story of a disgruntled ex-employee. I'm not shying away from the serious corporate espionage or the ethical...

🌟 Jan 8 🖱 20K 💬 450



```
80 func (l *Listener) listenToQueue(ctx context.Context, topic string, queue MessageQueue) {
81
82     // set the callback function for this event to commit the message to kafka
83     wrapper.SetCallback(func(c context.Context) {
84         commitToKafka(c, queue, msg)
85     })
86
87     l.workerPool.Submit(func() {
88         if err := l.eventHandler.HandleEvent(cctx, wrapper); err != nil {
89             log.Error(cctx, "Error handling event", log.Tags{"topic": topic, "error": err})
90             wrapper.DoCallback() // commit the message to kafka
91         }
92     })
93 }
94
95 // wait for an OS signal to exit
96 sigchan := make(chan os.Signal, 1)
97 signal.Notify(sigchan, syscall.SIGINT, syscall.SIGTERM)
98 <-sigchan
99
100 log.Info(cctx, "Received signal, shutting down...", log.Tags{"topic": topic})
```

I added this code block but copied it blindly from the example given in the segmentio/kafka-go docs. Is this the correct way to handle this? Or is there a better way?

The signal handling code you've added is problematic in its current location. It's inside the `listenToQueue` method which runs as a goroutine for each queue, meaning you'll have multiple signal handlers competing with each other. Additionally, this signal handling should be at a higher level in your application.

Here's how to fix it:

```
func (l *Listener) Run(ctx context.Context) error {
    for topic, queue := range l.queues {
        go l.listenToQueue(ctx, topic, queue)
    }
    <-ctx.Done()
    return nil
}

func (l *Listener) listenToQueue(ctx context.Context, topic string, queue MessageQueue) {
    defer queue.Close()

    for {
```



In Level Up Coding by Jacob Bennett

The 5 paid subscriptions I actually use in 2025 as a Staff Software Engineer

Tools I use that are cheaper than Netflix



Jan 7



12.3K



302



In Javarevisited by Rasathurai Karan

Java's Funeral Has Been Announced....💀💻

Oh, Java is outdated! Java is too verbose! No one uses Java anymore!



Mar 7



2.5K



132



See more recommendations