UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANTONIO QUEIRÓZ NETO

DB-FUZZY: UM NOVO MÉTODO PARA BUSCAS EM BANCOS DE DADOS

MOSSORÓ-RN

2020

ANTONIO QUEIRÓZ NETO

DB-FUZZY: UM NOVO MÉTODO PARA BUSCAS EM BANCOS DE DADOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação - Associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Profª Dra. Angélica Félix de Castro
Coorientador: Profº Dr. Leandro Carlos de Souza

MOSSORÓ-RN

2020

Q3{ Queiróz Neto, Antonio.
{DB-FUZZY: UM NOVO MÉTODO PARA BUSCAS EM BANCOS DE DADOS / Antonio Queiróz Neto. - 2020.
63 f. : il.

Orientadora: Angélica Félix de Castro.
Coorientadora: Leandro Carlos de Souza.
Dissertação (Mestrado) - Universidade Federal Rural do Semi-árido, Programa de Pós-graduação em Ciência da Computação, 2020.

1. Bancos de dados. 2. Otimização de busca. 3. Lógica Fuzzy. I. Félix de Castro, Angélica, orient. II. Carlos de Souza, Leandro, co-orient. III. Título.

ANTONIO QUEIRÓZ NETO

DB-FUZZY: UM NOVO MÉTODO PARA BUSCAS EM BANCOS DE DADOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: 28 / 08 / 2020.

Profa. Dra. Angélica Félix de Castro
Orientadora e Presidente da Banca

Prof. Dr. Leandro Carlos de Souza
Co-orientador – UFPB

Profa. Dra. Renata Maria Cardoso Rodrigues de Souza
Examinadora Externa – UFPE

Prof. Dr. Bruno Almeida Pimentel
Examinador Externo – UFAL

Aos meus pais
Francisco Queiroz (In Memorian) e
Cosma Oliveira

Agradecimentos

Agradeço infinitamente aos meus pais Cosma Oliveira e Francisco Queiroz (*in memorian*), por terem me ensinado tantas lições simplesmente sendo quem são. Em especial, a minha mãe, que sempre me proporcionou um ambiente sem restrições a carinho, risadas e livros.

Agradeço à minha família, que sempre apoiou e torceu por mim. À minha irmã Camila, ao meu irmão Arthur. Às minhas tias Jacinta, Écia, Lúcia e Francisca. Às minhas primas Cássia, Rúbia, Anny, Carol, Clara, Cecília, Candice, Tarsila e Ilana. Aos meus primos Heitor e Bernardo.

Agradeço aos amigos que fiz durante o mestrado, principalmente a Otília Sousa, Júlio Cartier, Cynthia Maia e Thyago Gomes, que estavam comigo em todos os momentos difíceis e que me inspiram a continuar dando o meu melhor, o que PAA uniu nada vai separar. Também agradeço a Johnattan Douglas, Thaiza Medeiros e Virgínia Maia, saudades de "quintar"com vocês.

Agradeço aos meus amigos Andrezza Kharla, Jefferson Lins, Maria Luzia e Glacilene Damásio. Minhas palavras parecem nunca ser suficientes ou significativas o bastante para dizer o quanto vocês significam para mim, agradeço todos os dias por tê-los como amigos. Agradeço também a Alisson Maurício, você entende todas as referências, e Amanda Costa; Agradeço a todos do Convento, em especial a Isabela Sousa, Ana Raiza, Vitor Hugo, Priscilla Souza e Isabela Romão.

Agradeço aos professores que fazem parte do PPgCC por seus ensinamentos e, em especial, aos meus orientadores Angélica Félix e Leandro Sousa, pela disponibilidade e, principalmente, pela paciência.

Agradeço à CAPES, pelo apoio financeiro que viabilizou esta pesquisa. E a todos que contribuíram para a realização deste sonho. **Muito obrigado.**

"Perhaps the greatest faculty our minds possess is the ability to cope with pain."
[*Patrick Rothfuss*]

Resumo

Atualmente, os dados armazenados em sistemas crescem rapidamente. Esses dados tem uma finalidade variada, porém o padrão para armazenamento na indústria de software é em bancos de dados. O tempo de recuperação de um registro específico em um banco de dados é diretamente proporcional ao volume de dados armazenado. Dessa maneira, bases de dados com um grande volume de registros demoram mais tempo na busca do que bases menores. Para resolver esse problema, esse trabalho propõe um novo método para buscas em bancos de dados, chamado DB-Fuzzy. Esse método usa conceitos de *fuzzy* para localizar dados individualmente. A busca demora um número constante de passos, independentemente do tamanho do banco de dados. Uma comparação com outro método presente na literatura mostra a eficiência do método proposto.

Palavras-chave: Banco de Dados, Otimização de Busca, Lógica Fuzzy.

Abstract

Nowdays, systems stored data increases rapidly. It may have different purposes, however, the standard in software industry is to store the data in databases. The time to retrieve a specific piece of data from a database is directly related to the volume of data stored. In this way, datasets with a significant volume of data would take longer to retrieve a specific tuple than smaller datasets. This paper proposes a new a novel method for search in databases, called DB-Fuzzy. It uses fuzzy concepts to locate individual data. It takes a constant number of steps for each searching, independently of the database size. Comparisons with literature method shows the efficiency of the proposed method.

Keywords: Database, Search Optimization, Fuzzy Logic.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BD	*Banco de Dados*
SGBD	*Sistema Gerenciador de Banco de Dados*
RSL	*Revisão Sistemática de Literatura*
QP	*Questão de Pesquisa*
QS	*Questão Secundária*
CI	*Critério de Inclusão*
CE	*Critério de Exclusão*
CQ	*Critério de Qualidade*
BMHT	*Boosting Multi Hash Tables*
GSS	*Graph Similarity Search*
MSGS	*Multi-Start Grid Search*
DLG	*Double Layer neighborhood Graph*
BFS	*Breadth First Search*
CBMIR	*(Content-Based Music Information Retrieval*
EMD	*Earth Mover's Distance*
TFI	*Temporal Focus of Interest*
QoS	*Quality of Service*
BNL	*Block Name Label*

Lista de símbolos

Γ	Letra grega Gama
θ	Letra grega Theta
σ	Letra grega Sigma

Sumário

1 INTRODUÇÃO

1.1 Contexto de Pesquisa

O volume de dados gerados cresce rapidamente, seja na indústria ou nas redes sociais. Gantz e Reinsel (2012) previram que mais de 40.000 exabytes de dados serão produzidos esse ano. Esses dados são produzidos em vários formatos e utilizados para vários fins, como tornar empresas mais eficientes, buscar a cura de doenças, avaliar os efeitos das mudanças climáticas, prever catástrofes ou qualquer empreendimento que demande uma enorme quantidade de dados (AMARAL, 2016).

Não existe um padrão único para a difusão ou divulgação dos dados, podendo ser apresentados de diversas maneiras (OLIVEIRA; GUERRA; MCDONNELL, 2018). Porém, de forma generalizada, esses dados podem ser separados em duas classes de acordo com suas estruturas: dados estruturados e não-estruturados. Os dados estruturados têm um formato pré-estruturado, como planilhas e arquivos com campos de tamanho, e são armazenados, por padrão, em bancos de dados relacionais. Os dados não-estruturados não possuem um modelo de dados, como imagens e vídeos, sendo mais adequados para serem armazenados nos bancos de dados não relacionais (ANAND; RAO, 2016).

Bancos de dados são coleções de dados que têm relação entre si e possuem sentido. Assim, bancos de dados representam uma abstração do mundo real com um propósito específico, logo devem ser construídos de forma ordenada. Sistemas Gerenciadores de Bancos de Dados (SGBD) ajudam a criar e manter bancos de dados computadorizados de forma ordenada.

Apesar do advento dos bancos de dados não relacionais, os bancos de dados relacionais ainda são o padrão universal da indústria de *software* por quase tanto tempo quanto existem (VYAWAHARE H. R.; KARDE; THAKARE, 2018). O modelo relacional modela os dados de forma que eles sejam percebidos como tabelas pelo usuário, em que cada linha representa um fato que corresponde a uma entidade ou relacionamento do mundo real.

Algumas características se destacam em um banco de dados, como o volume e a variedade dos dados armazenados. A velocidade da recuperação dos dados é, naturalmente, um dos aspectos fundamentais para o sucesso de um sistema de bancos de dados (YU; MENG, 1994). No entanto, o estado da arte na busca em bancos de dados relacionais não teve muitas mudanças nas últimas décadas.

1.2 Motivação

O tempo gasto para recuperar um registro específico em um banco de dados é diretamente proporcional ao volume de dados armazenado. Dessa forma, grandes bases de dados exigem mais tempo para recuperar o registro específico do que bases menores. Isso se torna um problema em uma realidade onde bancos de dados na ordem de bilhões de registros são frequentes. Conjectura-se que dividir uma tabela em várias outras menores diminuirá o tempo de busca. Contudo, nas últimas décadas, o estado da arte na busca em banco de dados relacionais não obteve avanços nesse sentido. Isso se deu porque os esforços científicos se voltaram para a aplicação e aperfeiçoamento dos bancos não-relacionais, uma vez que os dados não estruturados, gerados com mais frequência nos últimos anos, são melhor processados nessa outra modalidade de bancos de dados. No entanto, os bancos de dados relacionais ainda são usados para uma gama de aplicações que necessitam de grandes bases de dados. Nessa perspectiva, a motivação desta pesquisa está relacionada com a importância de investigar alternativas para diminuir o tempo de busca nessas bases.

1.3 Objetivos

O objetivo geral desta pesquisa é especificar um método baseado em tabelas de dispersão e conjuntos *fuzzy* para diminuição do tempo de busca em bancos de dados relacionais.

Objetivos específicos desta pesquisa incluem:

- Investigar o estado da arte relacionado à busca em bancos de dados;
- Definir o método para diminuição do tempo de busca em bancos de dados;
- Validar o método proposto.

1.4 Metodologia

Para alcançar os objetivos dessa pesquisa, foram escolhidos os seguintes passos:

- Revisão de Literatura: Uma revisão sistemática de literatura tem como objetivo identificar, avaliar e interpretar os estudos relevantes para uma questão de pesquisa em particular. Para este trabalho foi utilizado o processo proposto em Biolchini *et al.* (2007) e junto com algumas diretrizes definidas por Kitchenham (2004) com o objetivo de levantar o estado da arte relacionado à busca em bancos de dados;
- Prototipação: Para Sommerville (2011), a prototipação consiste no processo de implementação do artefato solicitado pelo cliente. Assim, a prototipação desse

trabalho consiste primeiramente em definir as partes que compõem o método, como a função de mapeamento. A segunda etapa considera a adequação dos procedimentos básicos de um banco de dados para a natureza do método.

- Validação: Para validação do método, é feita uma análise estatística comparando os tempos de busca do método proposto com o de outro presente na literatura.

1.5 Organização do documento

- **Capítulo 1** apresenta o contexto e motivação da pesquisa. Em seguida, são apresentados os objetivos e a metodologia seguida para alcançar os objetivos.
- **Capítulo 2** fornece os conceitos de tabelas de dispersão. Define o que se entende por conjuntos nebulosos e a representação geométrica desses conjuntos. Em seguida, apresenta os conceitos referentes à equalização.
- **Capítulo 3** investiga o estado da arte em métodos para redução de tempo em consultas de banco de dados no formato de uma revisão sistemática de literatura. O protocolo e detalhes dos estudos encontrados são mostrados no decorrer do Capítulo.
- **Capítulo 4** explica os detalhes para construção do método proposto, como a definição da função de mapeamento e do método de equalização, e as modificações necessárias nas operações básicas de um banco de dados para se ajustarem à natureza do método.
- **Capítulo 5** descreve os experimentos feitos para testar o método proposto e a base de dados utilizadas, assim como os resultados desses experimentos e compara o método proposto com outro método presente na literatura.
- **Capítulo 6** apresenta as considerações finais sobre a pesquisa, dá uma visão das limitações presentes e dos trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Introdução

Este capítulo apresenta a fundamentação teórica, que tem o objetivo de prover os principais conceitos relacionados ao método proposto. Tais conceitos são fundamentais para o seu total entendimento. Este capítulo está dividido em três seções. A primeira seção aborda os conceitos para manipulação de dados em uma tabela de dispersão. A segunda aborda a teoria dos conjuntos nebulosos, compreendendo algumas definições básicas e sua representação geométrica. Ao final, são apresentados os conceitos sobre equalização.

2.2 Bancos de Dados

Bancos de Dados são coleções de informações, também chamadas de dados, devidamente relacionadas e que têm sentido. Os dados podem ser armazenados e possuem um significado implícito. Machado (2018) afirma que um Banco de Dados deve possuir as seguintes propriedades:

- Ser uma coleção lógica coerente de dados com um significado inerente; deste modo, uma disposição desordenada dos dados não pode ser referenciada como Banco de Dados;
- Ser projetado, construído e preenchido com valores de dados para um propósito específico; um Banco de Dados possui um conjunto predefinido de usuários e de aplicações;
- Apresentar algum aspecto do mundo real, o qual é chamado de minimundo; qualquer alteração efetuada no minimundo é automaticamente refletida no Banco de Dados.

Os bancos de dados computadorizados são normalmente criados e mantidos pelos Sistemas Gerenciadores de Bancos de Dados (SGBDs). O SGBD é um sistema de software de uso geral que facilita o processo de definição, construção, manipulação e compartilhamento de bancos de dados entre diversos usuários e aplicações (DATE, 2004).

Definir um banco de dados envolve especificar os tipos, estruturas e restrições dos dados a serem armazenados (ELMASRI; NAVATHE, 2011). De maneira semelhante à engenharia de software, existem modelos para a criação de Bancos de Dados, chamados paradigmas. Dentre esses paradigmas, pode ser destacado o relacional que modela os dados de forma que eles sejam percebidos como tabelas pelo usuário.

Para que o usuário tenha uma melhor visão dessa abordagem relacional, o banco de dados é mostrado como um conjunto de tabelas bidimensionais. Cada linha na tabela representa uma coleção de dados relacionais; uma linha representa um fato que normalmente corresponde a uma entidade ou relacionamento do mundo real. Os nomes da tabela são usados para ajudar a interpretar o significado dos valores em cada linha enquanto os nomes das colunas representam os valores de dados em cada linha, com base na coluna em que cada valor se encontra; e todos os valores em uma coluna são do mesmo tipo de dado (OLIVEIRA *et al.*, 2015).

Segundo Machado (2018), a Teoria Relacional possui premissas que definem uma tabela de dados:

- Cada uma das tabelas é chamada de relação;
- O conjunto de uma linha e suas colunas é chamado de tupla;
- Cada coluna dessa tabela tem um nome e representa um domínio da tabela;
- A ordem das linhas é irrelevante;
- Não há duas linhas iguais;
- Usamos nomes para fazer referência às colunas;
- A ordem das colunas também é irrelevante;
- Cada tabela tem um nome próprio distinto de qualquer outra tabela no banco de dados.

É importante a compreensão de que cada linha de uma tabela representa um objeto, um assunto que é descrito pelos valores de cada uma dessas colunas. Um exemplo de uma relação típica de um banco de dados é mostrado na Tabela 1.

Tabela 1 – Exemplo de relação de um banco de dados.

Código	Nome	Sexo	Idade
001	Maria Silva	Feminino	11
002	José Oliveira	Masculino	30
003	Joana Fonseca	Feminino	23
004	Calor Costa	Masculino	11

Fonte: Autoria Própria (2020).

O exemplo da tabela pessoa mostra que cada tupla (linha) da relação deve ser interpretada como um fato dessa afirmação. De acordo com Oliveira *et al.* (2015), existem quatro operações que podem ser aplicadas nos estados das relações de um banco de dados:

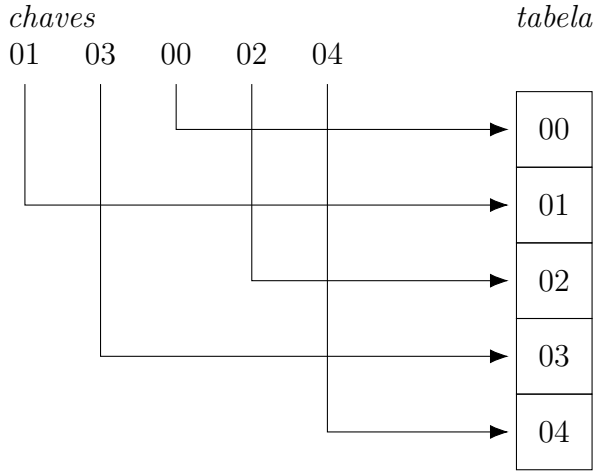
inserção, utilizada para adicionar novos dados à tabela; busca, utilizada para resgatar as informações de uma única tupla ou de várias ao mesmo tempo; deleção, para excluir uma linha da tabela; e atualização, utilizada para alterar as informações de uma tupla.

2.3 Tabelas de Dispersão

Uma tabela de dispersão, também conhecida como tabela *hash*, é uma estrutura de dados. A tabela de dispersão considera somente o valor absoluto de uma determinada chave, interpretado como um valor numérico, ao invés de ser organizada segundo o valor relativo de cada chave em relação às demais (SZWARCFITER; MARKENZON, 2010). Isto é, dado um item x, deve ser possível determinar diretamente sua posição de armazenamento no vetor (PREISS, 2000).

Outros métodos, como a busca binária e a pesquisa em uma árvore, demandam a aplicação de outros testes preliminares para determinar a posição de uma chave. Quando a chave é conhecida, sua posição na tabela de dispersão pode ser acessada diretamente (DROZDEK, 2005), como é ilustrado na tabela de acesso direto na Figura 1, em que a chave x é armazenada no compartimento x da tabela de dispersão. Na realidade, o método aproveita a possibilidade de acesso randômico à memória para alcançar uma complexidade média por operação de $O(1)$, sendo o pior caso, entretanto, $O(n)$ (SZWARCFITER; MARKENZON, 2010).

Figura 1 – Tabela de dispersão de acesso direto.



Fonte: Adaptado de Szwarcfiter e Markenzon (2010).

Para que o objetivo de performance seja alcançado, é necessária uma estratégia para realizar as operações de inserção e localização de um item sem fazer uma busca no vetor (PREISS, 2000). Essa estratégia é chamada de função de dispersão e sua função é mapear uma chave para um dos compartimentos da tabela de dispersão.

2.3.1 Funções de Dispersão

Uma função de dispersão é uma estratégia para o mapeamento de uma chave qualquer x em um dos compartimentos da tabela de dispersão. Essas funções precisam garantir que o valor retornado seja um índice válido para um dos compartimentos da tabela (DROZDEK, 2005). Assim, a função deve transformar cada chave em um valor no intervalo $[0, m-1]$ (SZWARCFITER; MARKENZON, 2010), sendo m o número de compartimentos da tabela de dispersão. Existem diversas funções de dispersão com graus de complexidade diferentes. Algumas dessas funções serão apresentadas no decorrer dessa Seção.

No método da divisão, a chave x é dividida pela dimensão da tabela m e o resto da divisão é usado como endereço-base (SZWARCFITER; MARKENZON, 2010). Pode-se calcular o valor dessa função de dispersão pela Equação 2.1. Em geral, essa abordagem é satisfatória para quase todo valor de m (PREISS, 2000). Existem alguns critérios que têm sido aplicados com bons resultados práticos, como escolher m de modo que seja um número primo não próximo a uma potência de 2 (SZWARCFITER; MARKENZON, 2010). A Figura 2 ilustra uma tabela de dispersão de tamanho 23. A chave 44 é mapeada para o compartimento 21 da tabela, a chave 46 para o compartimento 0, e assim por diante.

$$h(x) = x \bmod m. \tag{2.1}$$

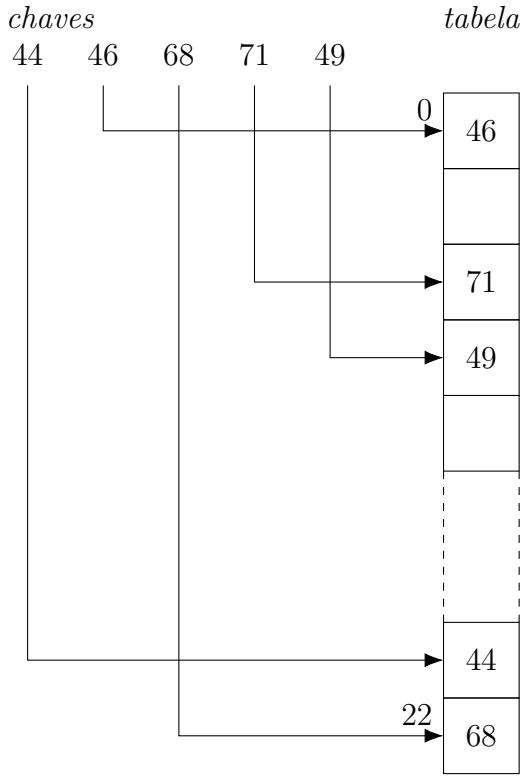
No método da dobra, a chave é dividida em diversas partes, que são processadas usando uma operação simples, tal como a soma, para combiná-las em certo modo (DROZDEK, 2005). A Figura 3 ilustra um exemplo de aplicação do método da dobra, em que os dígitos decimais da chave são d_1, ..., d_k e que uma dobra é realizada após o j-ésimo dígito à esquerda (SZWARCFITER; MARKENZON, 2010).

O método da multiplicação se beneficia do fato de uma multiplicação ser mais rápida que uma divisão inteira, sendo possível melhorar potencialmente o tempo de processamento do algoritmo de dispersão (PREISS, 2000). A chave é multiplicada por ela mesma, ou, alternativamente, por uma constante, e o resultado é armazenado numa palavra de memória de b bits (SZWARCFITER; MARKENZON, 2010).

No método da análise de dígitos é necessário um conhecimento prévio do tipo das chaves. A função de dispersão consiste em selecionar, de forma conveniente, alguns dos dígitos decimais que formam a chave para compor o seu endereço-base (SZWARCFITER; MARKENZON, 2010). É feita uma análise para cada dígito, até que os k melhores tenham sido encontrados. Assim, o endereço-base de uma chave só contém os dígitos escolhidos.

No método da extração, somente uma parte da chave é usada para calcular o endereço (DROZDEK, 2005). É importante notar que a porção excluída das chaves não deve ser suficientemente significativa para o cálculo do endereço-base.

Figura 2 – Método da divisão.



Fonte: Adaptado de Szwarcfiter e Markenzon (2010).

No método de transformação de raiz, a chave K é transformada em outro número base; K é expressado em outro sistema numérico que usa outra raiz (DROZDEK, 2005). Esse método pode ser aplicado em conjunto com outro método para melhorar sua eficiência.

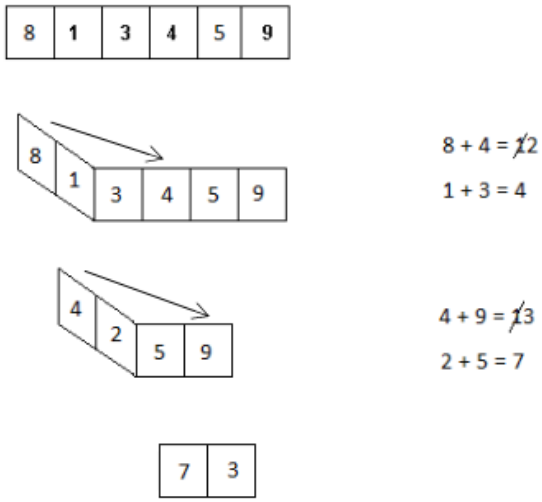
A Tabela 2 sumariza as definições das funções de dispersão apresentadas nessa Seção.

Tabela 2 – Funções de dispersão e suas descrições.

Função	**Descrição**
Método da Divisão	É usado o resto da divisão do valor da chave pelo tamanho da tabela como endereço-base.
Método da Dobra	A chave é dividida em diversas partes que são combinadas usando uma operação simples.
Método da Multiplicação	A chave é multiplicada por uma constante e parte do resultado é usada como endereço.
Método da Análise de Dígitos	Apenas alguns dígitos da chave são escolhidos para para compor o endereço-base.
Método da Extração	Uma parte que não influencie significativamente no mapeamento é extraída e o restante é usado para encontrar o endereço-base.
Método da Transformação de Raiz	A chave convertida para outro sistema de numeração.

Fonte: Autoria Própria (2020).

Figura 3 – Método da dobra.



Fonte: Szwarcfiter e Markenzon (2010).

Szwarcfiter e Markenzon (2010) definem três características que uma boa função de dispersão deve ter. Elas são: evitar colisões; espalhar as chaves pelo vetor; e, ser fácil de computar. Quando dois elementos são mapeados, através da função de dispersão, para o mesmo compartimento, é dito que aconteceu uma colisão. Algumas técnicas que lidam com essas colisões são explicadas na próxima seção.

2.3.2 Tratamento de colisões

Os métodos que tratam as colisões podem ser separados em duas categorias: por encadeamento, que consiste em armazenar as chaves com mesmo endereço-base em listas encadeadas, e por endereçamento aberto, em que as chaves com mesmo endereço-base são armazenadas na própria tabela, sem acrescentar nenhuma informação adicional. Algumas técnicas para tratamento de colisões são definidas a seguir.

O método de encadeamento exterior consiste em manter um conjunto de listas encadeadas, uma lista para cada compartimento da tabela. Cada compartimento da tabela de dispersão guardará somente o nó-raiz da lista encadeada. Em geral, a chave x é incluída na última posição da lista do compartimento $h(x)$. A ideia de incluir x no final decorre do fato de que a lista terá que ser percorrida de qualquer maneira, para assegurar que x não pertence à mesma, independentemente da posição de inclusão de x (SZWARCFITER; MARKENZON, 2010).

O encadeamento interior prevê a divisão da tabela T em duas zonas, uma de endereços-base, de tamanho p, e outra reservada aos sinônimos, de tamanho s (SZWARCFITER; MARKENZON, 2010). Cada compartimento da tabela tem dois campos, um para guardar a chave e outro para guardar um ponteiro que indica o próximo elemento da lista.

Assim, a lista encadeada ocupa o mesmo espaço de memória que a tabela de dispersão.

O endereçamento aberto por tentativa linear (CORMEN *et al.*, 2009) usa uma função de dispersão como a ilustrada na Equação 2.2, em que $i = 0, 1, ..., n-1$, para a alocação de entradas. Caso duas chaves distintas sejam mapeadas para o mesmo endereço-base, o novo nó, da chave x, será armazenado no endereço consecutivo $h'(x) + 1$ (SZWARCFITER; MARKENZON, 2010). Caso esse compartimento já esteja ocupado, a chave é armazenada no compartimento seguinte. Apesar de essa função ser de fácil implementação, ela pode causar longas tiragens de compartimentos ocupados, o que aumenta o tempo de busca.

$$h(k, i) = (h'(k) + i) \ mod \ n \tag{2.2}$$

A técnica por tentativa quadrática (CORMEN *et al.*, 2009) usa uma função como a ilustrada pela Equação 2.3, em que c_1 e c_2 são constantes auxiliares e $i = 0, 1, ..., n-1$. O objetivo desse método é mapear duas chaves próximas para endereços dispersos. Os valores de c_1 e c_2 devem ser escolhidos de tal forma que os endereços-base $h(x, k)$ correspondam a varrer toda a tabela para $k = 0, ..., m-1$ (SZWARCFITER; MARKENZON, 2010). Esse método funciona melhor que a sondagem linear, mas dependendo dos valores escolhidos para c_1 e c_2, pode levar a formas mais brandas das tiragens de compartimentos ocupados.

$$h(k, i) = (h'(k) + c_1 i + c_2 i^2) \ mod \ n \tag{2.3}$$

A técnica de dispersão dupla (CORMEN *et al.*, 2009) usa uma função como a ilustrada pela Equação 2.4, em que ambas as funções h' e h" sãos funções hash auxiliares. Esse método é capaz de gerar um número maior de possíveis sequências de tentativas distintas do que os métodos anteriores (SZWARCFITER; MARKENZON, 2010), já que as permutações causadas por essa função são similares a permutações escolhidas aleatoriamente.

$$h(k, i) = (h'(k) + ih''(k)) mod \ n \tag{2.4}$$

A Tabela 3 sumariza as técnicas para tratamento de colisões apresentadas nessa Seção.

Tabela 3 – Métodos para tratamento de colisões e suas descrições.

Método	**Descrição**
Encadeamento Exterior	Os compartimentos da tabela de dispersão contém a raiz de lista encadeada. Assim, os endereços-base das chaves são uma referência para uma lista.
Encadeamento Interior	É acrescentado um novo campo a cada compartimento da tabela de dispersão. Esse novo campo indica em qual compartimento na tabela está o próximo elemento da lista.
Endereçamento Aberto por Tentativa Linear	Calcula a sequência de tentativas de acordo com a seguinte equação: $h(x,k) = h'(x) + k$.
Endereçamento Aberto por Tentativa Quadrática	Calcula a sequência de tentativas de acordo com a seguinte equação: $h(x,k) = (h'(x) + c_1 k + c_2 k^2) \bmod m$.
Endereçamento Aberto por Dispersão Dupla	Calcula a sequência de tentativas de acordo com a seguinte equação, para a k-ésima tentativa: $h(x,k) = (h'(x) + k \cdot h''(x)) \bmod m$, $0 \leq k < m$, onde h', h'' são funções de dispersão.

Fonte: Autoria Própria (2020).

Apesar de suas vantagens, sendo a principal o baixo custo por operação, as tabelas de dispersão tendem a acessar o mesmo local da memória repetidamente e, a medida que as colisões aumentam, a sua eficiência diminui.

2.4 Conjuntos Nebulosos

Na teoria clássica dos conjuntos, a relação de pertinência compreende apenas dois casos, quanto ao relacionamento entre um elemento e um conjunto: ou o elemento faz parte do conjunto ou não faz parte. Na teoria dos conjuntos nebulosos, ou conjuntos fuzzy, é incluída uma nova possibilidade à relação de pertinência: um elemento pode pertencer parcialmente ao conjunto. A persistência em um conjunto nebuloso não é uma questão de afirmação ou negação, mas sim uma questão de grau (KLIR; YUAN, 1995).

Em muitos casos, é necessário medir os graus de incerteza e/ou certeza associados com uma dada situação ou experimento (LAZZERINI; JAIN; DUMITRESCU, 2000). A imprecisão é mensurada com uma medida de entropia de incerteza (KOSKO, 1991).

Cox (1994) define conjuntos nebulosos como funções que mapeiam um valor que pode ser membro de um conjunto para um número entre zero e um que indica seu grau de pertinência. Para Klir e Yuan (1995), os conjuntos nebulosos não só proveem uma representação significativa e poderosa de medida de incerteza, como também para conceitos vagos da linguagem natural.

Lazzerini, Jain e Dumitrescu (2000) definem um conjunto nebuloso como sendo um par (X, A), em que $A : X \to I$ e $I = [0, 1]$. X é um conjunto não-vazio considerado o

universo de discussão e o conjunto A é chamado de relação de pertinência.

O grau de pertinência de x em A, representado por A(x), é um valor entre 0 e 1. Assim, existem infinitos valores . Pode ser interpretado como o grau de plausibilidade da afirmação "x pertence a A"(LAZZERINI; JAIN; DUMITRESCU, 2000). Valores maiores denotam maiores graus de pertinência ao conjunto (KLIR; YUAN, 1995). Logo, quando A(x) = 1, entende-se como o elemento x definitivamente estando em A e quando A(x) = 0, como o elemento x definitivamente não estando em A.

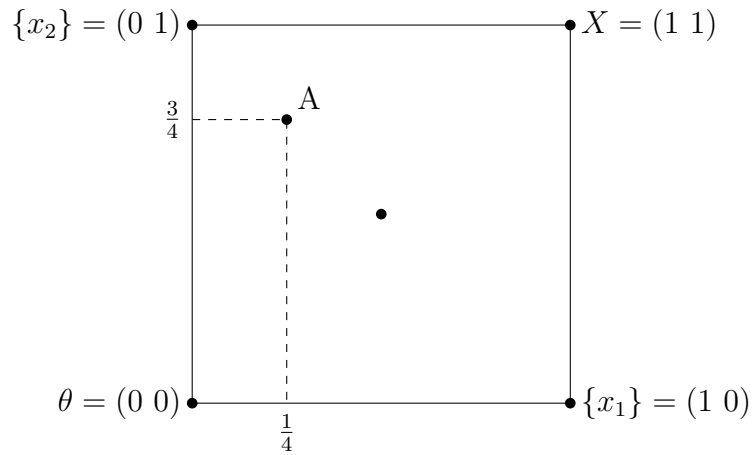
Na Subseção a seguir serão apresentadas algumas definições sobre conjuntos nebulosos que claramente são extensões de suas correspondentes para conjuntos normais, tal como a operação de complemento.

2.4.1 Geometria de um Conjunto Nebuloso

Alguns conceitos do paradigma nebuloso são mais facilmente entendidos quando vistos geometricamente. A geometria dos conjuntos nebulosos envolve tanto o domínio $X = \{x_1, ..., x_n\}$ quanto o conjunto das imagens $[0, 1]$ dos mapeamentos $f : X \longrightarrow [0, 1]$ (KOSKO, 1991). Geometricamente, o conjunto das partes de X, $F(2^X)$, se assemelha a uma unidade de hipercubo $I^n = [0, 1]^n$ e um conjunto nebuloso, a um ponto no cubo.

Considerando que o conjunto X tem apenas dois elementos, x_1 e x_2, a Figura 4 ilustra a representação geométrica do conjunto das partes de X como uma unidade de hipercubo I^2 e do seu subconjunto nebuloso A. Nesse exemplo, o grau de pertinência em A do elemento x_1, $A(x_1)$, é $\frac{1}{3}$ e o do elemento x_2, $A(x_2)$, é $\frac{3}{4}$. Então, o subconjunto nebuloso A tem coordenadas $\left(\frac{1}{3}\ \frac{3}{4}\right)$.

Figura 4 – Representação geométrica do subconjunto A como um ponto em I^2.



Fonte: Kosko (1990).

Os vértices da unidade de hipercubo I^2 têm coordenadas (0 0), (0 1), (1 0) e (1 1). O subconjunto cujas coordenadas são (0 0) é o conjunto vazio, $\emptyset$, uma vez que os elementos

x_1 e x_2 têm grau de pertinência igual a 0, ou seja, x_1 e x_2 definitivamente não pertencem a esse conjunto. Analogamente, o subconjunto cujas coordenadas são (1 1) é o próprio X, o subconjunto de coordenadas (1 0) é o conjunto $\{x_1\}$ e o subconjunto de coordenadas (0 1) é o conjunto $\{x_2\}$. Esses quatro conjuntos não são nebulosos já que não há incerteza quanto a pertinência dos elementos x_1 e x_2 neles.

2.5 Equalização

O processo de equalização tem como objetivo distribuir uniformemente as partes de um todo, seja unindo duas ou mais partes em uma única ou separando uma das partes em várias outras. Uma função de distribuição pode uniformizar dados contínuos (YANG, 2005).

Em Processamento Digital de Imagens, a equalização de histogramas é bastante comum. A equalização de imagens melhora seu contraste (AZEVEDO; CONCI; LETA, 2018), já que uma imagem com histograma estreito possui baixa qualidade visual. O objetivo da equalização do histograma é encontrar e aplicar uma operação pontual tal que o histograma da imagem modificada se aproxime de distribuição uniforme (BURGER; BURGE, 2009).

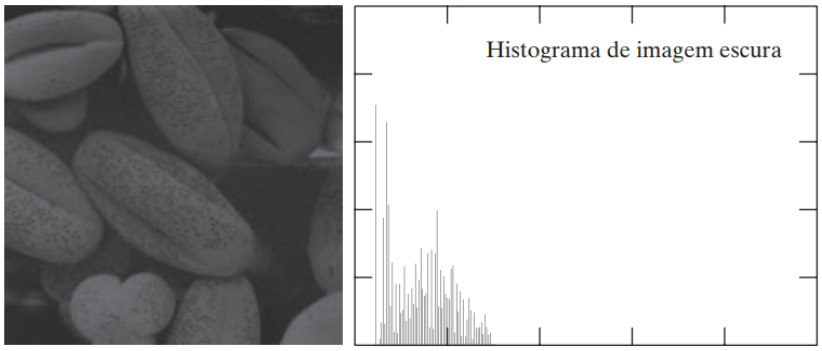
Gonzalez e Woods (2010) afirmam que, para valores discretos, são usados probabilidades (valores do histograma) e somatórios, em vez de funções de densidade de probabilidade e integrais, como acontece para valores contínuos. Assim, a forma discreta para equalização de um histograma é:

$$T\left(r_k\right) = \frac{(L-1)}{N} \sum_{j=0}^{k} n_j \tag{2.5}$$

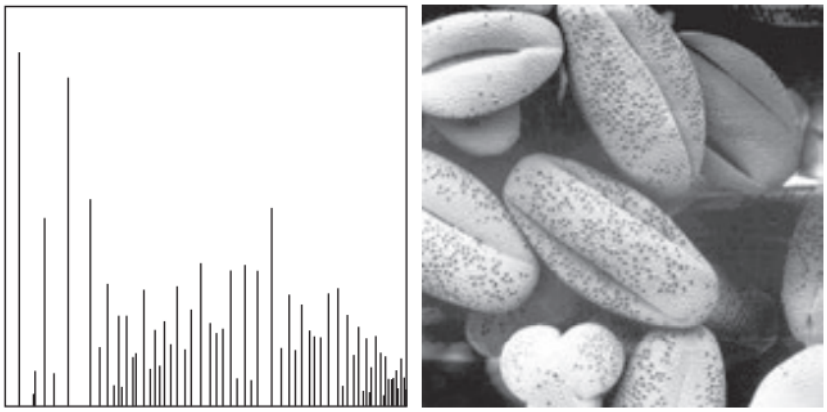
Um exemplo de equalização de histograma é mostrado na Figura 5. A Subfigura 5a ilustra uma imagem escura e seu histograma cujas barras se encontram mais à esquerda. A Subfigura 5b ilustra o histograma depois do processo de equalização. Apesar de as barras do histograma estarem mais espalhadas horizontalmente, não estão totalmente uniformes.

Figura 5 – Exemplo de histograma.

(a) Imagem original e seu histograma.



(b) Histograma e imagem após o processo de equalização.



Fonte: Adaptado de Gonzalez e Woods (2010).

Para Burger e Burge (2009), quando se trabalha de forma discreta, acontecerá somente mudanças e união das entradas do histograma, o que proporciona uma solução aproximada. Esse comportamento no processo de equalização justifica o histograma da Subfigura 5b não ser totalmente uniforme.

Este capítulo teve como objetivo fornecer uma base dos principais conceitos utilizados para o desenvolvimento do método proposto, esse entendimento é inerente à compreensão do funcionamento do método. De fato, conjuntos nebulosos e tabelas de dispersão fazem parte da própria natureza do método. A equalização é um processo fundamental que garante a eficiência do produto final.

3 MÉTODOS PARA REDUÇÃO DE TEMPO EM CONSULTAS DE BANCOS DE DADOS: UMA REVISÃO SISTEMÁTICA

3.1 Introdução

O registro de informações sempre se mostrou indispensável ao longo da história. A criação dos computadores aumentou ainda mais essa necessidade, visto que a medida que a complexidade das aplicações aumenta, maior deve ser a robustez do sistema de armazenamento. Assim, surgiram os Bancos de Dados (BD) e os Sistemas Gerenciadores de Bancos de Dados (SGBD - do inglês *Database Management System*).

É indiscutível que a sociedade está cada vez mais conectada. Seja na ciência ou nas redes sociais, a quantidade de informação sendo gerada é gigantesca. Levando em consideração essa informação, surgiu o questionamento de quais abordagens são usadas na recuperação de dados.

Algumas abordagens são amplamente utilizadas em buscas com chave única. A abordagem das tabelas de dispersão precisa de uma entrada que consiste de uma chave associada a um conteúdo. As técnicas de codificação por dispersão podem ser aplicadas a tabelas ou arquivos em que não há ordem previsível para acessar as entradas, sendo fundamental que a chave associada com o conteúdo desejado seja usada para localizar a entrada no armazenamento (MORRIS, 1968).

Ademais, árvores também são utilizadas em grandes medidas para realizar uma busca. Árvores B especificamente, ou suas variantes, são usadas em sistemas de bancos de dados para armazenar informação (CORMEN *et al.*, 2009). Sendo k um número natural, uma árvore T é da classe das árvores B se T estiver vazia ou obedecer às seguintes propriedades (BAYER; MCCREIGHT, 1972): 1) O caminho da raiz para qualquer uma das folhas tem o mesmo número de nós. 2) A raiz é uma folha ou tem pelo menos dois filhos. 3) Cada nó, exceto as folhas, tem pelo menos k + 1 filhos. 4) Cada nó tem no máximo 2k + 1 filhos. Como as árvores B têm muitos filhos, sua altura cresce logaritmicamente com o número de nós que contém. O procedimento de busca em uma árvore B leva O($log_d n$) (COMER, 1979), onde d é o número de filhos que cada nó tem.

Apesar dessas duas abordagens poderem ser usadas tanto em bancos de dados

relacionais quanto não-relacionais, novas técnicas foram desenvolvidas. Assim, o objetivo deste trabalho é, por meio de uma revisão sistemática de literatura, conhecer o estado da arte para recuperação de dados em bancos de dados, identificando os métodos, algoritmos, técnicas e abordagens em geral utilizadas.

O trabalho está organizado da seguinte maneira. Na Seção 3.2, é mostrado o processo usado na realização dessa revisão sistemática. Na Seção 3.3, é mostrada a condução da revisão sistemática. Na Seção 3.4, são mostradas as respostas das questões de pesquisa levantadas e é feita uma breve discussão dos trabalhos analisados. Por fim, as considerações finais deste trabalho são apresentadas na Seção 3.5.

3.2 Processo Sistemático

Kitchenham (2004) afirma que por meio de uma revisão sistemática de literatura é possível identificar, avaliar e interpretar todas as pesquisas relevantes para uma questão de pesquisa em particular, ou domínio, ou fenômeno de interesse de maneira repetível e imparcial.

Esta revisão sistemática seguiu o processo proposto em Biolchini *et al.* (2007) e usa algumas diretrizes definidas por Kitchenham (2004). Quatro pesquisadores participaram no processo de revisão: um estudante de mestrado com o papel principal de revisor e três pesquisadores seniores com a responsabilidade de validar a pesquisa.

Primeiro, definiu-se o protocolo de revisão, no qual foram estabelecidos os objetivos, questões de pesquisa, escopo, estratégia de pesquisa, critério para seleção de estudos e definição da *string* de busca como detalhado a seguir.

3.2.1 Objetivos e Escopo

A motivação para essa revisão sistemática surgiu de uma proposta de projeto de mestrado, cujo objetivo é utilizar uma estratégia de manipulação de dados para melhorar o tempo de consulta em tabelas de bancos de dados. Entretanto, antes de propor alguma estratégia é importante compreender o estado da arte do tema abordado. Assim, o escopo dessa revisão sistemática foi definido a partir dos seguintes objetivos:

- **Objetivo 1:** Identificar e analisar os métodos, algoritmos, técnicas ou abordagens que têm sido propostas para minimização do tempo de consultas em bancos de dados.
- **Objetivo 2:** Analisar a eficiência dos métodos, algoritmos, técnicas ou abordagens que têm sido propostas para minimização do tempo de consultas em bancos de dados.

- **Objetivo 3:** Identificar a forma de avaliação dos métodos, algoritmos, técnicas ou abordagens que têm sido propostas para minimização do tempo de consultas em bancos de dados.

Algumas especificidades dos objetivos desta revisão são definidas a seguir:

Intervenção. Esta revisão sistemática deve observar os métodos, algoritmos, técnicas ou abordagens que têm sido propostos para consultas a bancos de dados.

Controle. Alguns estudos primários devem ser encontrados durante as buscas nas bases de dados. A presença desses estudos garantem a eficiência desta revisão sistemática.

1. T. Bonny and B. Soudan, "Computation time reduction to speed-up the database searching process", 2015;
2. P. S. Patil, S. R. Patil, S. Rao, and S. B. Patil, "Customised approach for efficient data storing and retrieving from university databaseusing repetitive frequency indexing"; 2012.

População. Nesta revisão sistemática, serão observados somente os trabalhos que apresentem ou analisem métodos para diminuição do tempo nas operações de consultas em bancos de dados.

Resultados. Como resultado, espera-se um panorama dos métodos utilizados para diminuição do tempo nas operações de consultas em bancos de dados.

Aplicação. Pesquisadores que trabalham com otimização de operações em bancos de dados.

3.2.2 Questões de pesquisa

De modo a atender os objetivos, a questão de pesquisa (QP) que norteia esta revisão sistemática é "Qual o estado da arte para diminuição do tempo de consultas em bancos de dados?". A questão de pesquisa pode ser decomposta em uma série de questões mais específicas:

- **Questão primária (QP1):** Quais os métodos utilizados para diminuição do tempo nas operações de consultas em bancos de dados ?
- **Questão secundária (QS1):** Qual a eficiência desses métodos?
- **Questão secundária (QS2):** Como esses métodos têm sido avaliados e validados?

3.2.3 Estratégia de Busca

Algumas especificidades da busca devem ser definidas, como o critério de seleção das fontes, estratégia de pesquisa, palavras-chaves entre outras. Estas especificidades são apresentadas a seguir.

3.2.3.1 Critério para Seleção das Fontes

Com o objetivo de executar uma busca exaustiva por estudos primários, decidiu-se fazer uma busca *online* nas bibliotecas digitais mais comumente utilizadas na ciência da computação, além de consultas com especialistas.

3.2.3.2 Bases de Dados

As bibliotecas digitais consideradas neste trabalho são:

- IEEE Xplore.
- ACM Digital Library
- Science Direct.
- Scopus.

3.2.3.3 Estratégia de Pesquisa

Será utilizada a opção de busca avançada nas bibliotecas digitais escolhidas para um maior controle sobre os resultados apresentados.

3.2.3.4 Palavras-chave

Procurar por palavras-chaves adequadas é muito importante para a qualidade e cobertura dos resultados retornados pela busca nas bibliotecas digitais. A *string* de busca geral utilizada é composta de três termos, cada um combinado com seus respectivos sinônimos. Os três termos escolhidos e seus sinônimos são apresentados na Tabela 4. A *string* de busca geral definida precisa ser adaptada para atender às particularidades das bibliotecas digitais utilizadas nesta revisão, as *strings* adaptadas para cada biblioteca digital são apresentadas na Tabela 5.

Tabela 4 – Termos, palavras-chaves e sinônimos usados para criar a string de busca.

Palavra-chave	**Sinônimos em inglês**
banco de dados	"database", "DB", "BD", "data", "big data"
consulta	"retrieval", "recovery", "search", "query"
Redução de tempo	"time reduction,"time decrease", "time diminution", "lowering time"

Fonte: Autoria Própria (2020).

É importante fazer algumas observações sobre como foi realizada a pesquisa em cada biblioteca digital:

- **IEEE Xplore:** a busca foi restringida ao título e *abstract* das publicações ao selecionar a opção "*Metadata Only*" na busca avançada; aplicou-se também um filtro nativo da biblioteca para retornar artigos mais recentes, publicados de 2008 a 2018.
- **ACM Digital Library:** a busca foi restringida ao *abstract* das publicações ao adicionar o termo *recordAbstract* antes da *string* de busca; os artigos foram recuperados do conjunto *The ACM Guide to Computing Literature*; por fim, aplicou-se um filtro nativo da biblioteca para retornar artigos mais recentes, publicados de 2008 a 2018.
- **Science Direct:** a busca foi restringida ao título, *abstract* e palavras-chaves dos trabalhos; aplicou-se um filtro nativo da biblioteca para retornar artigos mais recentes, publicados de 2008 a 2018; somente dois tipos de artigos foram considerados: *review articles* e *research articles*.
- **Scopus:** a busca foi restringida ao título e *abstract* ao adicionar as palavras *TITLE-ABS-KEY* antes da string de busca. A busca também foi restringida à área de Ciência da Computação ao adicionar a restrição LIMIT-TO (SUBJAREA, "COMP"); por fim, aplicou-se um filtro nativo da biblioteca para retornar artigos mais recentes, publicados de 2008 a 2018.

Tabela 5 – Strings de busca para as bibliotecas digitais.

Biblioteca digital	**String de busca**
IEEE	((database OR data OR DB OR "big data") AND (recovery OR retrieval OR search OR query) AND ("time reduction" OR "time decrease" OR "time diminution" OR "lowering time"))
ACM digital library	recordAbstract:(database data "big data") AND recordAbstract:(recovery retrieval search query) AND recordAbstract:("time reductiontime decreasetime diminution lowering time")
Science direct	Title, abstract, keywords: (database OR data OR "Big Data") AND (recovery OR retrieval OR Search OR query) AND ("Time Reduction"OR "time decrease"OR "time diminution"OR "lowering time")
Scopus	TITLE-ABS-KEY ((database OR data OR db OR "Big Data") AND (recovery OR retrieval OR search OR query) AND ("Time Reduction"OR "time decrease"OR "time diminution"OR "lowering time")) AND (LIMIT-TO (SUBJAREA , "COMP"))

Fonte: Autoria Própria (2020).

3.2.4 Estratégia de Seleção dos Estudos

Alguns critérios devem ser usados para a seleção de estudos primários. Assim, foram definidos critérios para inclusão e exclusão. Os estudos primários incluídos devem pertencer a um dos critérios de inclusão a seguir:

- Critério de Inclusão (CI1): Estudos que definem algum método, algoritmo, técnica ou abordagem para consultas em bancos de dados que não sejam distribuídos.
- Critério de Inclusão (CI2): Estudos que aplicam algum método, algoritmo, técnica ou abordagem para consultas em bancos de dados que não sejam distribuídos.
- Critério de Inclusão (CI3): Estudos que analisam algum método, algoritmo, técnica ou abordagem método consultas em bancos de dados que não sejam distribuídos.

Observe que serão considerados apenas estudos escritos em inglês, por ser a língua internacionalmente aceita para redação de trabalhos científicos. Os estudos deverão ser excluídos durante a seleção de acordo com os critérios de exclusão a seguir:

- Critério de Exclusão (CE1): Estudos que não tratem de métodos para consultas em bancos de dados, ou que o façam em bancos de dados distribuídos.
- Critério de Exclusão (CE2): Estudos que são versões resumidas de trabalhos completos já encontrados na RSL.
- Critério de Exclusão (CE3): Estudos incompletos, não disponíveis e/ou duplicados.

Além dos critérios de exclusão, foram definidos critérios de qualidade que também serão utilizados para filtrar estudos.

- Critério de Qualidade (CQ1): Estudos que tenham pelo menos quatro páginas.
- Critério de Qualidade (CQ2): Estudos tenham sido publicados em revistas com qualis.
- Critério de Qualidade (CQ3): Estudos tenham sido publicados nos últimos dez anos.

3.2.5 Procedimento para Seleção dos Estudos

O processo de seleção de estudos foi organizado em quatro fases diferentes, descritas a seguir:

- **Fase 1 (Coleta de trabalhos):** Nesta fase, é feita a busca, utilizando a *string* específica, em todas as bibliotecas digitais e coleta-se os trabalhos encontrados.

- **Fase 2:** Determina-se quais estudos encontrados na fase anterior são relevantes baseado em seus títulos, *abstracts* e palavras-chaves. Nesta fase, são utilizados os critérios de inclusão e exclusão. Os estudos relevantes serão mantidos para uma inspeção futura.

- **Fase 3:** Para cada um dos estudos relevantes da fase passada, revisa-se cuidadosamente o artigo completo para determinar se este é um estudo primário relevante baseado nos critérios de inclusão e exclusão.

- **Fase 4:** Um especialista avalia e valida os estudos selecionados, com a possibilidade de inclusão ou exclusão de estudos.

3.2.6 Extração de dados

A extração de informação visa resumir a informação dos estudos primários selecionados. Preparou-se um formulário para extração e sintetização dos estudos a fim de responder as questões de pesquisa definidas no protocolo dessa RSL. O formulário usado é ilustrado na Figura 6. Os itens do formulário e suas descrições são definidos na Tabela 6.

Figura 6 – Formulário usado para extração dos dados.

ID:		Título:	
Ano:		Fonte:	
Autores:			
Motivo de Inclusão:			
Objetivo:			
Abordagem:			
Validação:			
Considerações:			

Fonte: Autoria Própria (2020).

Tabela 6 – Itens e descrições do formulário de extração de dados.

Itens comuns	**Descrições**
ID	Índice para identificação do artigo.
Título	O título do estudo primário
Ano	O ano quando o estudo primário foi publicado.
Fonte	A conferência ou revista onde o trabalho foi publicado.
Autores	Pesquisadores que escreveram o artigo.
Motivo de Inclusão	Critério pelo qual o artigo foi inserido na RSL.
Objetivo	Objetivo do artigo.
Abordagem	Qual método a solução proposta utiliza.
Validação	Como a solução proposta foi validada.
Considerações	Espaço reservado para alguma consideração adicional sobre o artigo.

Fonte: Autoria Própria (2020).

3.3 Condução da Revisão Sistemática

O processo de revisão definido na Seção 3.2 resultou em 323 artigos encontrados nas quatro bases. Depois da primeira fase, 37 estudos foram selecionados para a segunda fase, dos quais 13 foram identificados como estudos primários relevantes. Os detalhes da busca e seleção dos estudos são mostrados na Tabela 7 por base de dados e na Tabela 8 por fase.

Tabela 7 – Detalhes da busca e seleção dos estudos por base de dados.

Base de Dados	**Resultados da Busca**
IEEE Xplore	176
ACM Digital Library	42
Science Direct	27
Scopus	78
Total de artigos	**323**
Excluídos	5
Artigos para a segunda fase	**318**

Fonte: Autoria Própria (2020).

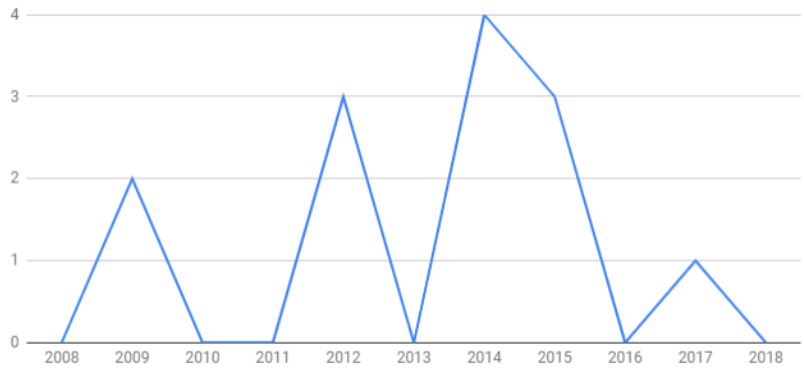
Tabela 8 – Detalhes da busca e seleção dos estudos por fase.

	Descrição	**Incluídos**	**Excluídos**
Fase 1	Resultados das buscas	323	0
Fase 2	Seleção por título e *abstract*	37	286
Fase 3	Seleção por leitura completa	13	24
Fase 4	Validação da seleção por um especialista	13	0
Total		13	310

Fonte: Autoria Própria (2020).

Dentre esses estudos, a maioria foi incluído pelo CI1, definição de um algoritmo, técnica ou abordagem para consultar em bancos de dados; nenhum artigo foi incluído pelo CI2 e somente um foi incluído pelo CI3, análise de método, algoritmo, técnica ou abordagem consultas em bancos de dados que não sejam distribuídos. Esses estudos foram publicados entre os anos de 2008 e 2018. Apresenta-se a distribuição de estudos por ano, na Figura 7. Nota-se que foi encontrado apenas 1 estudo em 2017; 2 estudos em 2009; 3 estudos em 2012 e 2015; 4 estudos em 2014 e nenhum estudo foi encontrado em 2008, 2010, 2011, 2013, 2016 e 2018.

Figura 7 – Distribuição de estudos primários por ano.



Fonte: Autoria Própria (2020).

Na Tabela 9, são apresentados os treze estudos primários relevantes encontrados nesta revisão sistemática.

3.4 Resultados e discussão

Nesta seção, as três questões de pesquisa que nortearam esta revisão sistemática são respondidas e os estudos primários relevantes são discutidos.

Tabela 9 – Estudos relevantes encontrados na revisão sistemática.

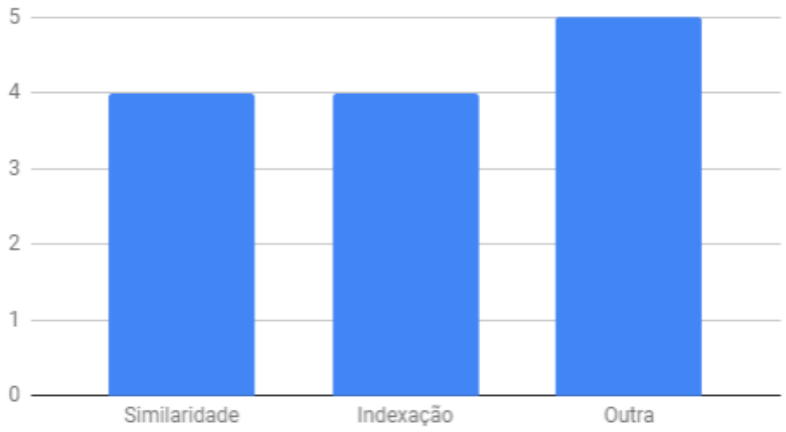
ID	**Ano**	**Título**	**Autores**
01	2009	An Adaptive Search System using Heterogeneous Document Vector Spaces	Kosuke Takano, Shuichi Kurabayashi, Xing Chen, Yasishi Kiyoki
02	2012	Boosting Multiple Hash Tables to Search	Jin-Cheng Li
03	2012	Customised Approach for Efficient Data Storing and Retrieving from University Database Using Repetitive Frequency Indexing	Preeti S Patil, Sunita R Patil, Srikantha Rao, Suryakant B Patil
04	2012	MapReduce Skyline Query Processing with a New Angular Partitioning Approach	Liang Chen, Kai Hwang, Jian Wu
05	2014	ADAM - A Database and Information Retrieval System for Big Multimedia Collections	Ivan Giangreco, Ihab Al Kabary, Heiko Schuldt
06	2014	Caching Support for Skyline Query Processing with Partially Ordered Domains	Yu-Ling Hsueh, Tristan Hascoet
07	2014	Fast Music Information Retrieval with Indirect Matching	Takahiro Hayashi, Nobuaki Ishii, Masato Yamaguchi
08	2014	Improvements to Efficient Retrieval of Very Large Temporal Datasets with the Travel-Light Method	Alexandre Valle de Carvalho, Marco Amaro Oliveira, Artur Rocha
09	2014	Integrating Spatial Information into Inverted Index for Large-Scale Image Retrieval	Bien-Van Nguyen, Duy Pham, Tanh Duc Ngo, Duy-Dinh Le, Duc Anh Duong
10	2015	Computation Time Reduction to Speed-up the Database Searching Process	Talal Bonny, Bassel Soudan
11	2015	Double-layer Neighborhood Graph Based Similarity Search for Fast Query-by-Example Spoken Term Detection	Kazuo Aoyama, Atsunori Ogawa, Takashi Hattori, Takaaki Hori
12	2015	Evaluation of Parallel Indexing Scheme for Big Data	Kenta Funaki, Teruhisa Hochin, Hiroki Nomiya, Hideya Nakanishi
13	2017	Optimizing the Query Performance of Block Index Through Data Analysis and I/O Modeling	Tzuhsien Wu, Jerry Chou, Shyng Hao, Bin Dong, Scott Klasky, Kensheng Wu

Fonte: Autoria Própria (2020).

3.4.1 QP1: Quais os métodos utilizados para diminuição do tempo nas operações de consultas em bancos de dados?

Analisando os 13 estudos primários, duas abordagens se sobressaem quanto ao número de vezes que aparecem. Essas abordagens são: similaridade e indexação. Pode-se observar na Figura 8 que ambas as abordagens aparecem quatro vezes nos estudos, caracterizando aproximadamente 30% dos trabalhos.

Figura 8 – Abordagens usadas nos estudos primários.



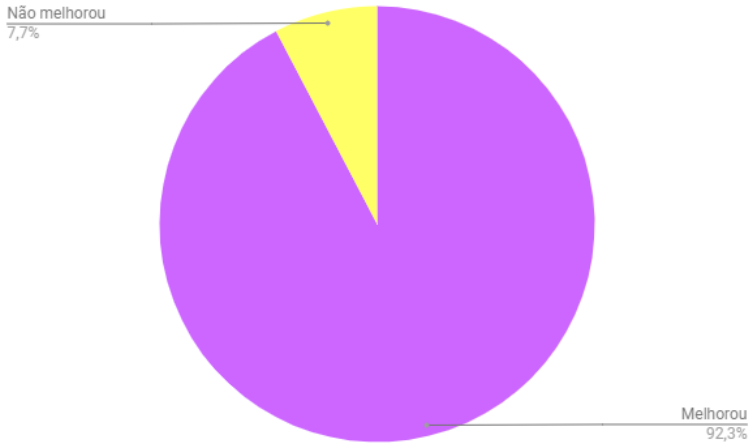
Fonte: Autoria Própria (2020).

Em uma abordagem de indexação, o banco de dados é dividido em seções disjuntas, em que cada uma tem um índice. De modo que, quando se quer recuperar um registro, só é necessário procurar por tal registro numa seção identificada por um índice específico. Em uma abordagem de similaridade, são definidas características para uma mídia e, então, são feitas comparações entre o que está na *string* de busca e os registros do banco de dados para encontrar mídias similares.

3.4.2 QS1: Qual a eficiência desses métodos?

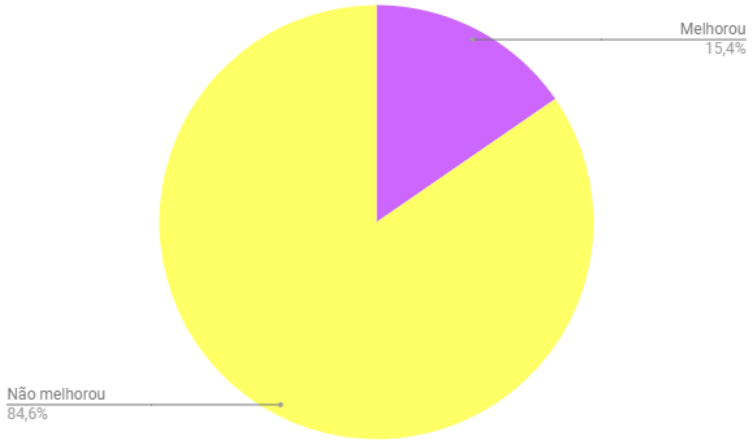
Na Figura 9, observa-se que, aproximadamente, 92,3% dos trabalhos reduziram o tempo de recuperação nas consultas aos bancos de dados quando comparados às abordagens usadas. Porém, como pode-se observar na Figura 10, apenas, aproximadamente, 15,4% dos trabalhos aumentaram a precisão. Somente um dos estudos reduziu o tempo e aumentou a precisão.

Figura 9 – Trabalhos que mostraram melhora no tempo.



Fonte: Autoria Própria (2020).

Figura 10 – Trabalhos que mostraram melhora na precisão.



Fonte: Autoria Própria (2020).

Na Tabela 10, são especificados quais trabalhos melhoraram o tempo e a precisão nas consultas em bancos de dados.

3.4.3 QS2: Como esses métodos têm sido avaliados e validados?

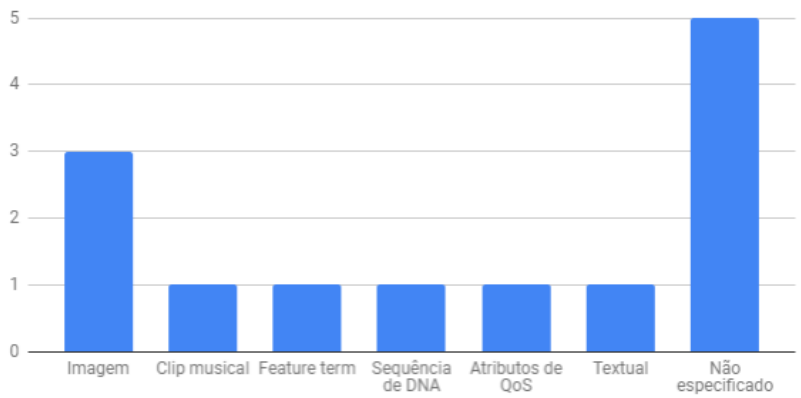
Em sua maioria, as validações foram feitas por meio de testes em uma base de dados. As abordagens propostas foram comparadas com outros métodos encontrados na literatura. Na Figura 11, observa-se que, apesar de cinco trabalhos não especificarem o tipo de dados com que trabalharam, a maior parte dos estudos trabalha com tipos de dados específicos.

Tabela 10 – Tempo e precisão.

ID	Tempo	Precisão
01	Sim	Sim
02	Sim	Sim
03	Sim	Não
04	Sim	Não
05	Sim	Não
06	Sim	Não
07	Sim	Não
08	Sim	Não
09	Não	Não
10	Sim	Não
11	Sim	Não
12	Sim	Não
13	Sim	Não

Fonte: Autoria Própria (2020).

Figura 11 – Tipos de dados usados nas validações.



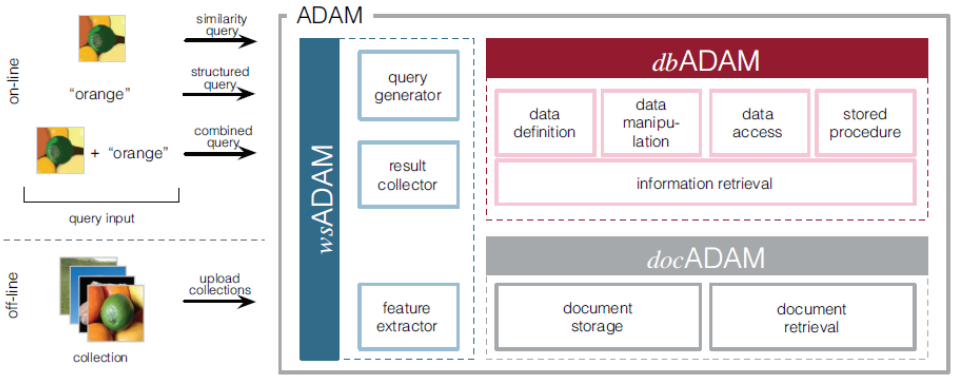
Fonte: Autoria Própria (2020).

3.4.4 Discussão dos Estudos Primários

No trabalho apresentado por Giangreco, Kabary e Schuldt (2014), é introduzido o ADAM, uma nova abordagem para combinar a tecnologia dos bancos de dados e a recuperação de informação para dados de grande multimídia. A arquitetura do ADAM é dividida em três componentes, como mostrado na Figura 12. O *middleware* (1) *ws*ADAM, que atua como uma porta para todas as funções do ADAM. É responsável pela distribuição de novas informações e, em interação com o *db*ADAM, a orquestração de *queries.* O (2) *db*ADAM é o cerne do sistema que armazena e recupera os documentos do banco de dados. O sistema ADAM pode ser escalado para coleções de tamanhos maiores ao adicionar novos fragmentos de *db*ADAM que agem como trabalhadores para a camada *ws*ADAM. Por fim, (3) *doc*ADAM é responsável efetivamente pelo armazenamento e

recuperação dos documentos de multimídia para propósitos de exibição. O ADAM suporta tanto a recuperação booleana quanto a busca por aproximação de similaridade. A busca booleana pode ser aplicada a todos os campos estruturados. A recuperação baseada em similaridade, por outro lado, pode ser usada para procurar entre os arranjos de características extraídos para k documentos similares. Para aperfeiçoar a eficiência das *queries*, ADAM suporta uma versão adaptada de Vetores de Aproximação (VA) *File*, como introduzido em (WEBER; SCHEK; BLOTT, 1998), citado por (GIANGRECO; KABARY; SCHULDT, 2014), comprimindo vetores de características para uma assinatura simples e pequena e posteriormente consultar as assinaturas de uma maneira sequencial.

Figura 12 – Visão de sistema do ADAM.



Fonte: Giangreco, Kabary e Schuldt (2014).

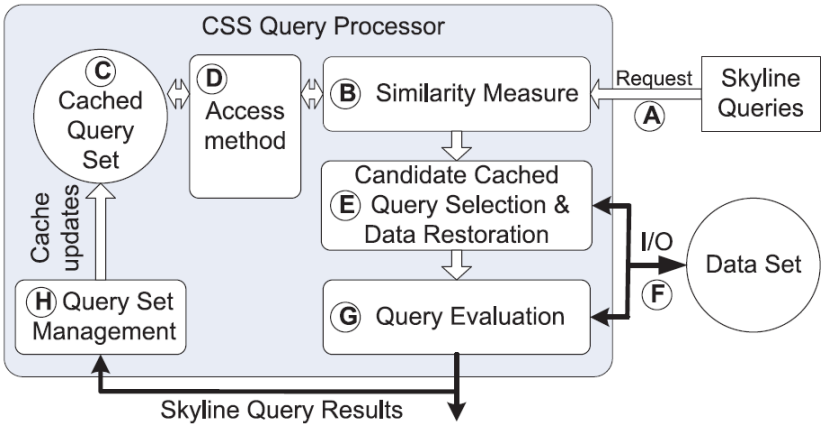
No estudo apresentado por Takano *et al.* (2009), é proposto um sistema de busca adaptativa usando espaços de vetores de documentos heterogêneos. O sistema provê uma função de busca adaptativa ao implementar duas funções: uma função de construção dinâmica para os componentes do motor de busca com diferentes domínios de busca; e, uma função de seleção no banco de dados. Para criar os componentes do motor de busca em um modelo de espaço de vetor, o sistema adaptavelmente seleciona conjuntos de termos de características entre diferentes domínios de busca de acordo com os documentos sendo vistos atualmente. Se um destes documentos incluir diferentes tópicos no seu contexto, o sistema constrói diferentes componentes do motor de busca de acordo com cada tópico. Além disso, o sistema exclui documentos que são ruídos dos resultados das buscas ao usar conjuntos de termos de características qualificados, tais como os termos dos índices de dicionários e enciclopédias. Para o propósito principal de reduzir o tempo de computação dos cálculos de similaridade em cada espaço do vetor de documentos, o sistema seleciona bancos de dados de documentos adequados de acordo com os termos da consulta feitas pelo usuário.

Li (2012) propõe um algoritmo de reforço que se apoia em múltiplas tabelas de dispersão que balanceiam a recuperação e a precisão. O método maximiza a precisão da dispersão de cada tabela de dispersão em uma distância de *hamming* menor que um pequeno

limite *d*. Dada uma *query*, os pontos de *mis-hash* não encontrados na tabela de dispersão atual têm uma alta probabilidade de serem encontrados na próxima tabela de dispersão. Na etapa de busca, só é preciso acessar os baldes cuja distância de *hamming* é menor que *d* da *query* de todas as tabelas de dispersão até que se obtenha um número de resultados relevantes (complexidade de tempo de busca constante). O algoritmo proposto melhora consideravelmente a recuperação enquanto preserva uma alta precisão. São apresentados um (1) algoritmo para reforçar múltiplas tabelas de dispersão (BMHT) e um (2) algoritmo de busca eficiente nas BMHT. Para o algoritmo (1), as tabelas de dispersão são aprendidas sequencialmente. Os pares de pontos similares em semântica devem cair nos mesmos baldes ou em baldes próximos (dentro de uma distância de *hamming d*) enquanto pares de pontos diferentes em semântica devem cair em baldes diferentes com uma grande distância de *hamming*. Em BMHT, uma amostra no banco de dados pode ter diferentes códigos de dispersão em diferentes tabelas. Então, no algoritmo (2), dada uma query, a pesquisa da dispersão na BMHT é feita retornando todas as amostras que tem uma distância de *hamming* menor que o limite *d* em todas as *L* tabelas de dispersão, isso é, obtém-se o conjunto da operação de união de todos os resultados retornados das *L* tabelas.

Hsueh e Hascoet (2014) conjecturam que *queries* que foram processadas previamente com preferências de usuário similares àquelas da *query* atual podem contribuir com pontos de candidatos a resultados úteis. Na Figura 13, é mostrado o *framework* do sistema que utiliza os seguintes passos: O processador de *queries* computa inicialmente os resultados de uma *skyline query* para uma *query* de requisição (A) e armazena o conjunto resultado com as preferências associadas (C). Subsequentemente, quando uma nova requisição de *query q* entra no sistema, uma *Task* (B) executa uma avaliação de similaridade ao computar pontuações de similaridade para *q* e para cada *query* armazenada em um conjunto selecionado por meio de um método de acesso (D). Após sua conclusão, a *Task* (B) encaminha um lista ordenada de *queries* de candidatos para outra *Task* (E), que por sua vez seleciona um conjunto de candidatos da lista. Se a nova *query q* não for completamente respondida pela cache, o componente de restauração dos dados acessa o conjunto dos dados (F) para executar *queries* menos restringidas para restaurar todas os possíveis pontos de resposta que faltaram. Finalmente, em uma nova *Task* (G), dada uma lista de *queries* candidatas e os pontos de resposta faltosos que foram restaurados, se houver algum, o processador de *queries* avalia os resultados baseados nas preferências da nova *query* para refinar a resposta final. Devido o espaço de armazenamento ser limitado, tem-se uma *Task* (H) responsável por expurgar a *cache* ao preservar as preferências mais populares; isso é, eliminando *queries* com as preferências menos recentes da *cache.*

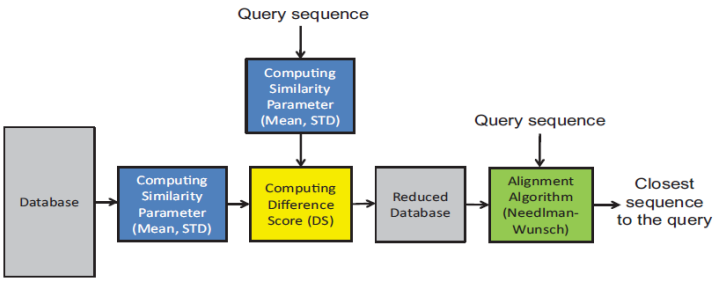
Figura 13 – Framework do sistema.



Fonte: Hsueh e Hascoet (2014).

Bonny e Soudan (2015) introduzem uma técnica nova e eficiente para reduzir o tempo de computação de aplicações de busca em bancos de dados ao usarem parâmetros de similaridade que são baseados na média e no desvio padrão dos códigos distribuídos de cada sequência no banco de dados. Essa técnica pode ser aplicada em conjunção com métodos prévios para melhorar seus tempo de execução. O banco de dados das aplicações de computação de sequência contém muitas sequências. Quando a extensão do banco de dados é reduzida, o tempo de computação requerido para as aplicações de sequenciação também é reduzido. Na Figura 14, é mostrado o diagrama de bloco da técnica introduzida para redução da amplitude do banco de dados procurado. A técnica proposta passa por várias etapas diferentes. Na primeira, os parâmetros de similaridade para cada sequência do banco de dados é computada. Na segunda etapa, é computado o *difference score* entre a busca e cada sequência do banco de dados. Depois que o *difference score* é computado, baseado no parâmetro de similaridade selecionado, as sequências do banco de dados são ordenadas de acordo com seus *difference scores*, do menor para o maior. A última etapa é aplicar o algoritmo de alinhamento (algoritmo de Needleman-Wunsch) nas sequências que tem um baixo *difference score*. Isso provê um alinhamento ótimo em um tempo razoável.

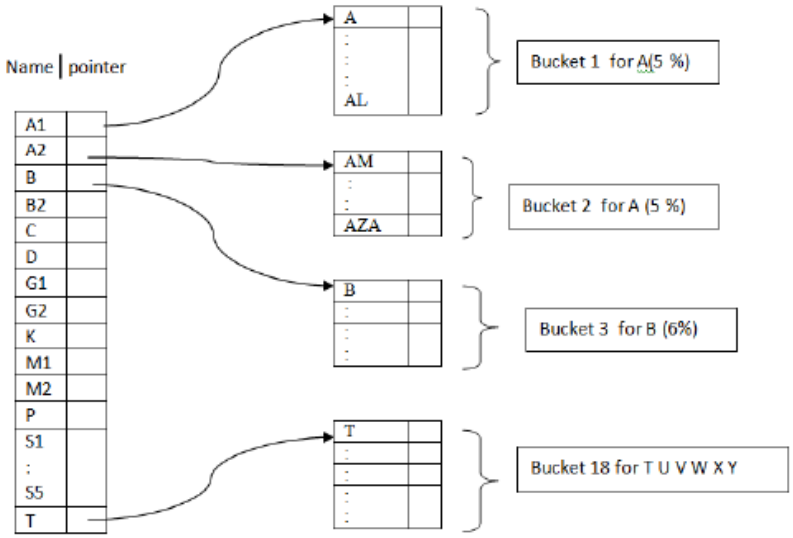
Figura 14 – Diagrama de bloco da técnica para reduzir a amplitude do banco de dados utilizado.



Fonte:Bonny e Soudan (2015).

No estudo feito por Patil *et al.* (2012), dois problemas dos bancos de dados indexados em 26 baldes, de acordo com as 26 letras do alfabeto, são tratados. São eles: (1) alguns baldes são deixados vazios como o balde dos nomes que começam com 'X'; (2) a procura nos baldes com uma alta frequência podem ter tempo de busca requerido bem alto. Na arquitetura proposta, primeiro é calculada a porcentagem da ocorrência que cada letra do alfabeto tem como a letra do início atributo do nome. Então, os baldes são alocados de acordo com essa porcentagem e os ponteiros não mapeados no arquivo de índices, como mostrado na Figura 15. Mais baldes são designados para as letras do alfabeto mais frequentes e menos baldes para as letras do alfabeto menos frequentes.

Figura 15 – Indexação de frequência repetitiva.



Fonte: Patil *et al.* (2012).

O trabalho proposto por Aoyama *et al.* (2015) foca em melhoras no método de busca por similaridade baseado em grafos (GSS). São determinados os parâmetros estruturais $k^{(b)}$ e $k^{(u)}$ dos grafos, verifica-se propriedades estatísticas em k de um grafo ordinário; um nível médio, um tamanho de caminho mais curto médio, e o produto deles que relaciona o custo computacional na busca em gráficos. Nota-se que o valor k afeta tanto o custo computacional na busca e acurácia da busca. A acurácia da busca é particularmente controlada tanto pelo valor de k quanto pelo parâmetro de busca L, que é o número de vértices de início no algoritmo MSGS. Aplica-se um algoritmo de busca simples ao DLG, que é a combinação dos algoritmos MSGS e BFS. A chave da aceleração da busca de similaridade está na estrutura do grafo como o índice mais do que no algoritmo de busca.

No trabalho apresentado por Funaki *et al.* (2015), é implementado e avaliado um esquema de indexação paralela proposto pelos autores em um trabalho anterior. Uma árvore R é usada como uma estrutura de inserção multidimensional. O esquema é avaliado

pelo tempo de inserção e tempo de recuperação variando o número de dados inseridos, o número de nós-normais e os esquemas de distribuição.

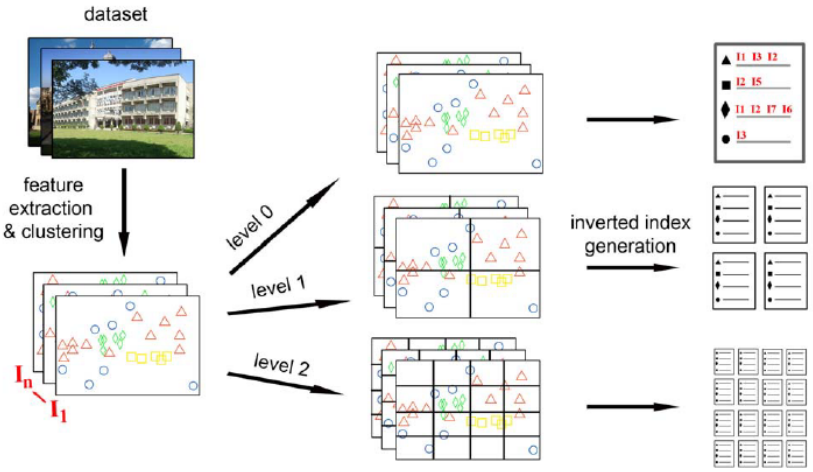
Em seu trabalho, Hayashi, Ishii e Yamaguchi (2014) propõe um método rápido de avaliação de similaridade chamado *indirect matching*. O método proposto é um *framework* geral que é independente de descritores de músicas individuais e métricas para avaliação de similaridade. O foco do método proposto é melhorar a execução de um sistema CBMIR existente mantendo os méritos dos descritores e da métrica usada no sistema. A abordagem proposta evita o alto custo computacional dos cálculos de EMD diretos entre uma *query* e cada clipe musical no banco de dados usando resultados de similaridade previamente avaliados com um pequeno número de *queries* de músicas pré-selecionadas chamadas de *queries* representativas. Antes da efetiva fase de recuperação *online* começar, usando *queries* representativas pré-selecionadas, o método proposto calcula com antecedências similaridades de cada clipe musical no banco de dados com as *queries* representativas. As similaridades calculadas são armazenados como uma tabela de similaridade. Na fase *online*, primeiramente, o vetor de similaridade entre a *query* atual (a *query* introduzida pelo usuário) e as *queries* representativas é calculada. Depois, a similaridade entre a *query* e cada clipe musical no banco de dados é indiretamente avaliada ao medir a similaridade dos vetores de similaridade entre a *query* e cada clipe musical no banco de dados.

No trabalho apresentado por Carvalho, Oliveira e Rocha (2014), são implementados melhoras ao método *TravelLight*, introduzido em (CARVALHO; OLIVEIRA; ROCHA, 2013), que age em bancos de dados temporais. Essas melhoras resultam da observação de empecilhos resultando do fato do método executar comandos diretamente no banco de dados: (1) a execução da primeira *query* temporal; (2) juntar a uma relação temporária persistida de uma coluna de estado adicional e índice correspondente; (3) atualização da coluna de estado; e (4) acesso concorrente. O algoritmo proposto é executado toda vez que o cliente requisitar informações. Quando invocado, o algoritmo primeiro determina se o TFI (Foco de Interesse Temporal) intersecta qualquer um dos intervalos atualmente armazenados. Se sim, calcula quais itens armazenados intersectam o TFI (*hits*). Em seguida, o algoritmo verifica se o TFI está inteiramente contido no conjunto de intervalos atualmente armazenados e, se sim, o subconjunto inteiro de *hits* armazenado já está disponível e portanto não há necessidade de recuperar informações adicionais. Nesse caso, o subconjunto de *hits* armazenado é retornado. Se o TFI atual parcialmente intersecta ou não intersecta o TFI atual, o procedimento continua calculando o conjunto de intervalos para o qual as informações são requeridas de serem recuperadas. Em seguida, os itens são buscados e carregados chamando a recuperação de itens que temporariamente intersectam o conjunto requerido de intervalos e que, ao mesmo tempo, não intersectam o conjunto de intervalos já completamente carregados na *cache*. Depois, a *cache* é aumentada se o número de *hits* já armazenados na *cache* mais os itens a serem carregados for maior do que o tamanho atual da *cache*. Aqui, o aumento considera um fator de 120% do

intervalo de crescimento requerido. Em seguida, o algoritmo calcula se o espaço livre disponível na *cache* é o suficiente para abranger o carregamento dos novos itens. Se não, uma estratégia é aplicada para recuperar o conjunto mínimo de intervalos (*toFree*) da *cache* que contém pelo menos um mínimo do número *toReleaseItems* de itens. A estratégia será aplicada considerando todos os itens armazenados que não estão na lista de *hits*. Todos os itens na *cache* que estão completamente contidos no conjunto de intervalos *toFree* serão deletados também e os intervalos correspondentes liberados da *cache*. Depois de liberar o espaço necessário, os itens carregados e os respectivos intervalos são adicionados na *cache*. Finalmente, a união dos intervalos é feita, para que seja mantida uma representação mínima para o conjunto de intervalos armazenados correspondentes aos itens armazenados. O algoritmo termina com o retorno de todos os itens que intersectam o TFI atual (*hits + load*).

É proposta porNguyen *et al.* (2014) uma abordagem que pode tirar proveito da informação espacial das características para aumentar a precisão enquanto mantém um curto tempo de recuperação. A ideia básica é construir múltiplos índices invertidos de palavras em diferentes regiões das imagens. Para tal, foi empregada uma pirâmide espacial para reforçar *voting* e a indexação de critérios da técnica original de indexação invertida. A ideia é dividir a imagem em células usando a pirâmide espacial com um dado nível máximo. Então, *visual words* são indexadas por célula em que pertencem. *Voting* é executado nas células em todos os níveis. Assim, para duas imagens que tenham o as mesmas *visual words* na mesma célula serão dados mais votos que aquelas que tenham as mesmas *visual words* distribuídas em células diferentes. Particularmente, a abordagem proposta usa índices multi-invertidos, que mantém a estrutura básica de índices invertidos mas divide em *multi-file*. Uma visão geral da abordagem proposta é mostrada na Figura 16. Depois de extrair as características de todas as imagens no banco de dados procurado, os descritores de características são quantizados para formar um vocabulário de *visual words*. Cada imagem contém um conjunto de *visual words*. Nesse momento, informações espaciais de todas as características são empregadas. Usa-se uma pirâmide espacial para particionar todas as imagens em células cada vez mais refinadas dependendo dos níveis da pirâmide definidos, e colecionando palavras encontradas dentro de cada célula. Em seguida, o conjunto de palavras encontrado em cada célula das imagens será usado para gerar um arquivo de índice invertido. O número de arquivos de índice invertido é igual ao número de células de pirâmide espacial. No processo de busca, primeiro se extrai características da imagem de busca. Então, *visual words* são computadas dessas características usando o dicionário construído anteriormente. Baseado na posição de cada *visual words*, elas são atribuídas nas redes de células criadas pela pirâmide espacial. Cada *visual word* é usada para acessar de forma imediata o índice invertido correspondente para simultaneamente obter e classificar a lista de imagens candidatas. É usado *voting* para classificação, então a lista da contagem de votos foi construída durante o processo de acessar os índices invertidos.

Figura 16 – Visão geral da abordagem proposta.



Fonte: Nguyen *et al.* (2014).

No trabalho proposto por Chen, Hwang e Wu (2012), dois problemas da adoção de uma abordagem de *Skyline* na resolução de problemas de *Quality of Service* (QoS) são tratados: (1) o crescimento exponencial da complexidade de *Skyline* no espaço de seleção e (2) como assegurar a QoS dos serviços selecionados de *Skyline*. Tais problemas são importantes na geração de uma solução *Skyline* com garantia de QoS em aplicações de *real-life web*/serviços na nuvem. Estendeu-se o modelo de *MapReduce* para a resolução do problema de seleção de *Skyline* em plataformas na nuvem. Para acelerar o processo de seleção de *Skyline* explorou-se o alto grau de paralelismo distribuído em *datacenters* automatizados ou nuvens de computação públicas. O espaço de busca de serviço é primeiro particionado em subespaços, então *skylines* locais de cada partição são calculadas em paralelo, e serviços de *skyline* globals são finalmente computados ao combinar todas as escolhas de *skyline* locais. Deve-se notar que a escolha de *skylines* locais podem não ser necessariamente ótima global. São geradas três versões de *MapReduce* do algoritmo de *Skyline* BNL (*Block Name Label*), baseadas em três esquemas de particionamento do espaço dos dados. Os algoritmos de *Skyline* propostos são denotados por *MR-Dim*, *MR-Grid* e *MR-Angle*. Especificamente, no *MR-Angle*, é proposto o uso de um algoritmo de particionamento angular para divisão do espaço dos dados, que se aumenta a eficiência do processo de busca *skyline* baseada em *MapReduce*.

Em seu trabalho, Wu *et al.* (2017) propõe a otimização da performance de busca por índices de bloco ao fornecer uma análise de um limite superior rígido nas requisições de I/O de resposta de uma *query* e o desenvolvimento de três técnicas de otimização baseadas nas características de *datasets* científicos e na modelagem de desempenho de sistemas de arquivos paralelos. A primeira técnica de otimização é chamada *merge read*. A ideia é maximizar o rendimento de entrada e saída no momento da busca ao ler múltiplos blocos de *hit* descontinuados de uma só vez quando suas localizações são perto o suficiente dentro

de uma distância de *treshold*. Para encontrar uma configuração de *treshold* apropriada, construiu-se a modelo de desempenho de *Lustre*, e demonstrou-se que a decisão baseada em modelo pode alcançar um desempenho perto do ótimo em experimentos em ambientes reais. A segunda técnica é chamada *adaptive dynamic schedule*. Essa técnica procura balancear o volume de trabalho da recuperação de dados entre os processos do leitor. Essa técnica supera o tráfego instável de entrada e saída de sistemas de arquivos compartilhados em paralelo cini *Lustre*, e assegura que a largura de banda de entrada e saída agregada dos processos de leitura seja totalmente utilizados. Finalmente, a terceira técnica é chamada *partial sort*. Ela explora a ideia de usar uma técnica de indexação mais custosa, como a indexação invertida, como indexação secundária em um bloco para compensar a limitação da indexação de blocos.

3.5 Conclusões e Limitações

Neste trabalho, propôs-se a apresentação de uma revisão sistemática de literatura com o propósito de obter o estado da arte dos métodos, algoritmos, técnicas ou abordagens que minimizassem o tempo de consulta a bancos de dados. Por essa razão, definiu-se um protocolo de RSL, apesentou-se a condução e os resultados desta revisão. As bases de dados pesquisadas foram *IEEE Xplore*, *ACM Digital Library*, *Science Direct* e *Scopus*, sendo considerados apenas artigos escritos em inglês. Foram encontrados um total de 323 estudos, nas quatro bases, sendo que somente 13 foram considerados relevantes, a partir dos critérios de inclusão e exclusão. Como resultados, foi identificado que a maior parte dos estudos usam alguma medida de similaridade, geralmente para um tipo de dado específico, ou técnica de indexação para melhorar as consultas em bancos de dados. Uma limitação dessa revisão foi a redução do escopo de busca, no qual não foram incluídos os trabalhos que tratassem de todos os tipos de bancos de dados, como os distribuídos. Como trabalho futuro, propõe-se a inclusão de tais estudos.

4 O MÉTODO DB-FUZZY

4.1 Introdução

O método DB-Fuzzy divide uma tabela de tamanho n em várias sub-tabelas. Seus tamanhos são aproximadamente s, que é um parâmetro do método, e $s \ll n$. De modo similar a uma tabela de dispersão, uma função de dispersão é usada para determinar qual das sub-tabelas contém o registro procurado. Para isso, as chaves da tabela de dispersão são tratados como um conjunto *fuzzy*. Graus de pertinência são usados para localizar a sub-tabela adequada que contém o registro procurado. Devido a natureza do método, também é necessário definir novos procedimentos para as operações básicas em bancos de dados. São elas: busca, deleção, atualização e inserção. Os detalhes da construção do método são mostrados no decorrer deste capítulo, assim como as mudanças necessárias para adaptar as operações básicas de um banco de dados ao método proposto.

4.2 Grau de Pertinência

A ideia do método é usar dois pontos de referência locais para competir como pontos de representação para uma dada entrada. Então, dado um conjunto p-dimensional $\mathbb{X} = \{x_1, \cdots, x_n\}$, ambos os pontos de referências locais $c_1 = (c_1^1, \cdots, c_1^p)$ e $c_2 = (c_2^1, \cdots, c_2^p)$ são definidos, respectivamente, de acordo com as Equações (4.1) e (4.2).

$$c_1^j = \min_{i=1}^{n}\{x_i^j\} \tag{4.1}$$

$$c_2^j = \max_{i=1}^{n}\{x_i^j\} \tag{4.2}$$

O grau de pertinência $\Gamma(x_k)$ para $x_k \in \mathbb{X}$ é computado de acordo com a Equação (4.3).

$$\Gamma(x_k) = \frac{d(x_k, c_1)}{d(x_k, c_1) + d(x_k, c_2)} \tag{4.3}$$

em que $d(x_k, c_1)$ e $d(x_k, c_2)$ são medidas de distância *city block* entre x_k e os elementos c_1 e c_2, respectivamente. Como o valor da chave é unidimensional, a distância é calculada como o valor absoluto da diferença entre a chave e os pontos c_1 e c_2. Assim, quanto mais perto a chave x_k estiver de c_1, menor o valor $d(x_k, c_1)$ e maior o valor $d(x_k, c_2)$;

analogamente, quanto mais perto x_k estiver de c_2, maior o valor $d(x_k, c_1)$ e menor o valor $d(x_k, c_2)$.

O grau de pertinência de um conjunto *fuzzy* sempre é um valor no intervalo $[0, 1]$. Então, se faz necessário provar que a Equação 4.3 possui um conjunto imagem adequado. Essa prova consiste de dois passos: demonstrar que $0 \leq \Gamma(x)$, para $c_1 \neq c_2$, e que $\Gamma(x_k) \leq 1$. A prova apresentada leva em consideração algumas propriedades de medidas de distâncias. Dados três elementos $x_q, x_r, x_s \in \mathbb{X}$, d é uma função $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_+ \cup \{0\}$ que satisfaz as propriedades presentes nas Equações (4.4- 4.7) (GAN; MA; WU, 2007):

1. Não-negatividade
$$d(x_q, x_r) \geq 0. \tag{4.4}$$

2. Reflexividade
$$d(x_q, x_r) = 0 \iff x_q = x_r. \tag{4.5}$$

3. Comutatividade
$$d(x_q, x_r) = d(x_q, x_r). \tag{4.6}$$

4. Desigualdade triangular
$$d(q, s) \leq d(q, r) + d(r, s). \tag{4.7}$$

1) Será demonstrado que $\Gamma(x_k) \geq 0$:
É suposto que $c_1 \neq c_2$. O numerador da Equação 4.3 é uma medida de distância, então esse valor é maior ou igual a zero. Nota-se que $d(x_k, c_1) = 0 \iff x = c_1$ e $d(x_k, c_2) = 0 \iff x = c_2$. Já que $c_1 \neq c_2$, x não pode assumir ao mesmo tempo ambos os valores de c_1 e c_2, pelo menos uma das distâncias no denominador será maior ou igual a zero. Consequentemente, a razão entre o numerador e o denominados sempre será maior ou igual a zero.

2) Será demonstrado que $\Gamma(x_k) \leq 1$:
É suposto que $\Gamma(x_k)$ pode ser representado pela razão de dois números reais, p, q, com $p \geq 0$ e $q > 0$.

$$\Gamma(x_k) = \frac{p}{q} = \frac{d(x_k, c_1)}{d(x_k, c_1) + d(x_k, c_2)}. \tag{4.8}$$

Como $q = d(x_k, c_1) + d(x_k, c_2)$, $p = d(x_k, c_1)$ e $d(x_k, c_2) \geq 0$, conclui-se que $q \geq p$. Então, a razão $\dfrac{p}{q}$ é sempre menor que ou igual a 1. Consequentemente,

$$\frac{d(x, c_1)}{d(x, c_1) + d(x, c_2)} \leq 1. \tag{4.9}$$

Assim, de acordo com a definição de $\Gamma(x_k)$, as propriedades de pertinência *fuzzy* são satisfeitas.

4.3 Função de Mapeamento

Uma função de mapeamento é um processo que transforma os elementos do seu conjunto domínio em elementos de seu conjunto imagem. Para tabelas de dispersão, a função de dispersão leva um elemento para um compartimento da tabela. O objetivo da função é determinar qual sub-tabela provavelmente contém a informação procurada. Diferente das tabelas de dispersão clássicas, nessa nova metodologia, não há necessidade de se preocupar com colisões. Esse processo de mapeamento é sumarizado no Algoritmo 1.

Algorithm 1: FUNÇÃO DE MAPEAMENTO

Input: chave
Output: inteiro
1 **begin**
2 | Calcular o grau de pertinência da chave usando a Equação (4.3);
3 | Calcular a extensão de cada intervalo, dividindo 1.0 pelo número de intervalos;
4 | Dividir o grau de pertinência pela extensão do intervalo;
5 | Retornar o piso da divisão como um inteiro;
6 **end**

Ao transformar um conjunto de dados tabelar em um conjunto *fuzzy*, é possível computar o grau de pertinência de acordo com a Equação (4.3). O grau de pertinência é mapeado para um valor inteiro, que identifica a sub-tabela. Dessa forma, o intervalo $[0, 1]$ é dividido em vários intervalos menores com aproximadamente a mesma extensão. Para cada um desses sub-intervalos é atribuído um valor inteiro único, que identifica a sub-tabela associada. Para encontrar esses sub-intervalos, um processo de equalização é usado.

4.3.1 Processo de Equalização

Inicialmente, o intervalo $[0, 1]$ é dividido em pedaços menores, porém iguais. O número de sub-intervalos gerados é um parâmetro do método. As pertinências das chaves do banco de dados são computadas e alocadas em seu respectivo sub-intervalo. Nesse momento, frequentemente, os sub-intervalos têm quantidades diferentes de valores de pertinência. Um processo de equalização é aplicado para combinar sub-intervalos e produzir aproximadamente a mesma quantidade de elementos em cada subdivisão.

O processo de equalização garante que o tempo usado para buscar um registro é similar, independentemente da sub-tabela em que está armazenado. Esse processo é baseado no método para equalização discreta de histogramas de imagens (GONZALEZ;

WOODS, 2010). Nesse processo, os elementos do mesmo grupo não podem ser separados. Dois ou mais intervalos podem ser agregados em um único grupo, ou um único intervalo pode se tornar um grupo por si mesmo. O Algoritmo 2 detalha esse método.

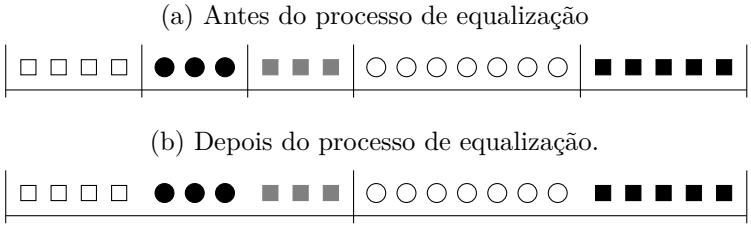
Algorithm 2: MÉTODO DE EQUALIZAÇÃO

```
   Input: conjunto de intervalos
   Output: conjunto de intervalos equalizado
 1 begin
 2     Compute o tamanho ideal das sub-tabelas;
 3     while existir um sub-intervalo fora de um grupo do
 4         if tamanho do grupo atual ≤ tamanho ideal then
 5             Adicione o sub-intervalo atual ao grupo atual;
 6         else
 7             if (tamanho ideal - tamanho do grupo atual) ≥ (tamanho do grupo atual
                + tamanho do compartimento atual - tamanho ideal) then
 8                 Adicione o sub-intervalo atual ao grupo atual;
 9                 Crie um novo grupo;
10             else
11                 Criar um novo grupo vazio;
12                 Adicione o sub-intervalo atual ao grupo atual;
13             end
14         end
15         Faça o novo grupo ser o grupo atual;
16         if Compartimento atual é o último then
17             Adicione o sub-intervalo atual ao grupo atual;
18         end
19     end
20 end
```

Dado um tamanho ideal, o método de equalização une intervalos em um único grupo até que a quantidade de elementos nesse grupo alcance tal tamanho. Para cada intervalo, o método decide se deve juntá-lo ao grupo atual ou ao próximo. Essa decisão é feita levando em consideração o tamanho do grupo atual. No caso de o tamanho do grupo atual ser menor que o tamanho ideal, os elementos do intervalo são adicionados ao grupo atual. Caso contrário, o método verifica se é melhor adicionar os elementos ao grupo atual ou ao próximo grupo. O último intervalo sempre é adicionado ao último grupo. Um exemplo desse processo é ilustrado na Figura 17, em que, para fins de exemplificação o tamanho ideal considerado é 11. Inicialmente, tem-se cinco conjuntos: quadrado branco, círculo preto, quadrado cinza, círculo branco e quadrado preto. Depois do processo, tem-se somente dois grupos. Apesar de os tamanhos finais dos grupos não serem idênticos, a diferença é mínima e não deverá interferir no tempo de busca.

Cada grupo criado no processo de equalização representa uma das sub-tabelas que serão criadas no banco de dados. Um arranjo é usado para armazenar para qual sub-tabela

Figura 17 – Exemplo de sub-intervalos antes (a) e dos grupos formados depois (b) do processo de equalização, considerando 11 o tamanho ideal.



Fonte: Autoria Própria (2020).

os elementos de um intervalo serão realocados. Cada índice do arranjo representa um inteiro associado a um intervalo e o valor armazenado no índice representa o grupo em que esse intervalo termina. Por fim, as sub-tabelas são efetivamente criadas no banco de dados e todas os registros da tabela original são realocados para elas. Uma vez que as sub-tabelas foram criadas, é necessário definir novos procedimentos para as operações básicas em um banco de dados.

4.4 Operações do banco de dados

O método DB-Fuzzy modifica a estrutura do banco de dados ao dividir uma tabela em várias outras. Mesmo que para o usuário final essa modificação não seja aparente, é necessária a adaptação das operações de busca, deleção, atualização e inserção. Esses procedimentos devem levar em consideração a alocação dos registros em sub-tabelas, além de assegurar que o tempo de busca seja idêntico para qualquer tupla. As próximas Subseções demonstram as modificações necessárias nos procedimentos.

4.4.1 Operação de Busca

A operação de busca é a mais usada no ciclo de vida do banco de dados. Usuários usam essa função para acessar os registros no banco de dados. A chave precisa ser um parâmetro para essa função. Primeiro, é necessário determinar em qual sub-tabela o registro está. Para isso, a chave é mapeada para um valor inteiro usando o Algoritmo 1 em que o grau de pertinência é encontrado usando a Equação 4.3 descrito previamente. Então, é encontrado em qual grupo o inteiro está incluído, procedendo a verificação do arranjo criado no Processo de Equalização (Seção 4.3.1). Por fim, uma consulta SQL comum é usada para recuperar o registro na sub-tabela específica. Se a tupla estiver no banco de dados, ela é retornada; caso contrário, um registro nulo é retornado. Esse procedimento de busca pode ser sumarizado no Algoritmo 3.

Em termos de análise de complexidade, considerando a quantidade de registros em cada sub-tabela s e a quantidade de registros na tabela original n, é escolhido um

Algorithm 3: Busca

Input: Chave
Output: Registro

```
1 Mapeie o chave para o intervalo correspondente usando o Algoritmo 1;
2 Encontre o grupo em que esse intervalo está incluído depois do processo de
    equalização usando o arranjo criado por ele;
3 Use uma consulta SQL para recuperar a tupla da sub-tabela correta;
4 if registro é encontrado then
5 |   Retorne o registro;
6 else
7 |   Retorne um registro nulo;
8 end
```

$s \ll n$. O algoritmo 1 tem complexidade de tempo constante; assim, a complexidade para computar o índice da sub-tabela correta é, também, constante, já que a quantidade de sub-tabelas e o tamanho da tabela original não interferem nesse cálculo. Logo, somente o tamanho de cada sub-tabela, s, deve ser considerado na complexidade da busca. O método leva, no máximo, $O(s)$ passos para encontrar um registro específico. Como o tamanho das sub-tabelas é fixado em um valor constante durante a execução do método, a complexidade pode ser considerada $O(1)$, sendo um tempo constante de busca.

4.4.2 Operação de Deleção

Quando um registro não é mais necessário, independentemente do motivo, ele precisa ser removido do banco de dados. A operação de deleção é importante porque previne a superpopulação nos bancos de dados. Dada uma chave, o primeiro passo é determinar em qual sub-tabela o registro está armazenado. Para isso, o grau de pertinência do registro é calculado usando a Equação 4.3 e, então, o resultado é mapeado para um valor inteiro. Em seguida, é encontrado o grupo que o inteiro está e uma sentença SQL comum é usada para deletar o registro da sub-tabela. Se esse procedimento for eficaz, é retornado o valor verdadeiro. A operação de deleção pode ser sumarizado pelo Algoritmo 4.

4.4.3 Operação de Atualização

A função de atualização é usada para alterar um ou mais atributos de um registro. Diferentemente do processo de deleção, o registro não é apagado do banco de dados, somente os valores contidos nos atributos são modificados. Além de uma chave, novos valores para os atributos do registro precisam ser recebidos como entrada. Para atualizar um registro, o primeiro passo é determinar em qual sub-tabela esse registro está. O grau de pertinência é calculado usando a Equação 4.3 e o resultado é mapeado para um valor inteiro. Uma sentença SQL comum é usada para atualizar os valores dos atributos do

Algorithm 4: Deleção

Input: Chave
Output: Booleano
1 Mapeie a chave para o intervalo correspondente usando o Algoritmo 1;
2 Encontre em qual grupo esse inteiro está depois do processo de equalização usando o arranjo criado por ele;
3 Apague o registro da sub-tabela usando uma sentença SQL;
4 **if** *bem-sucedido* **then**
5 | Retorne verdeiro;
6 **else**
7 | Retorne falso;
8 **end**

registro depois de identificar em qual grupo esse inteiro está, procedendo uma verificação do arranjo criado no Processo de Equalização. Caso os atributos tenha sido modificados com êxito, um valor booleano verdadeiro é retornado; caso contrário, é retornado um valor booleano falso. Esse procedimento pode ser sumarizado pelo Algoritmo 5.

Algorithm 5: Atualização

Input: Chave, Novos Valores para os Atributos
Output: booleano
1 Mapeie o chave para o intervalo correspondente usando o Algoritmo 1;
2 Encontre o grupo em que esse intervalo está incluído depois do processo de equalização usando o arranjo criado por ele;
3 Usar uma sentença SQL para ajustar os atributos do registro para os novos valores;
4 **if** *bem-sucedido* **then**
5 | Retorne verdadeiro;
6 **else**
7 | Retorne falso;
8 **end**

4.4.4 Operação de Inserção

O procedimento de inserção é usado para adicionar novos registros no banco de dados. Apesar de essa operação ser mais usada no começo do ciclo de vida do banco de dados, a operação de inserção pode ser usada em qualquer momento desse ciclo. Diferentemente dos procedimentos anteriores, quando um novo registro é adicionado no banco de dados, não há uma chave associada a ele. Uma vez que as chaves são inteiros e crescem incrementalmente por um, a chave para o novo registro pode ser determinada somando um à maior chave no banco de dados. Então, a sub-tabela na qual a chave será armazenada é determinada ao calcular o grau de permanência usando a Equação 4.3 e mapeando o resultado para um valor inteiro. Depois de encontrar a qual grupo o registro deve se juntar, ele é efetivamente armazenado em uma sub-tabela.

Deve-se ser considerado, também, a situação em que uma sub-tabela se torne muito maior que as outras. Para remediar essa situação, um limiar (σ) é definido. Se uma sub-tabela aumentar de tamanho o suficiente para passar desse limiar, a sub-tabela é dividida em duas novas sub-tabelas. Para isso, os registros são organizados de acordo com suas chaves e o registro na mediana é determinado. Esse registro é usado como ponto de divisão da sub-tabela, as chaves menores que ou iguais a ele permanecem na sub-tabela e as chaves maiores que ele são realocadas para a nova sub-tabela. Também, é necessário atualizar o arranjo com os índices criado no processo de equalização e um valor booleano é retornado indicando êxito ou fracasso. O procedimento de inserção pode ser sumarizado no Algoritmo 6.

Algorithm 6: Inserção

Input: Registro
Output: Booleano

```
1  Encontrar a maior chave no banco de dados;
2  Incrementar essa chave em um;
3  Mapear essa chave para o intervalo correspondente usnado o Algoritmo 1;
4  Encontrar em qual grupo esse inteiro deve estar depois do processo de equalização,
     usando o arranjo criado nesse processo;
5  Adicionar o registro na sub-tabela correta usando uma sentença SQL comum;
6  if tamanho ≥ σ then
7  |   Ordenar a tabela usando o atributo chave como referência;
8  |   Encontrar o valor mediano das chaves;
9  |   Dividir a tabela em duas;
10 |   Atualizar o arranjo de índices;
11 end
12 if bem-sucedido then
13 |   Retorne verdadeiro;
14 else
15 |   Retorne falso;
16 end
```

Este capítulo teve como objetivo definir o método DB-Fuzzy e novos procedimentos para as operações básicas em bancos de dados. A construção do método foi divida em duas etapas: a definição de uma função de mapeamento baseada em conjuntos fuzzy e de um método de equalização. Os procedimentos de busca, deleção, atualização e inserção também foram modificados a fim de respeitar a natureza do método.

5 EXPERIMENTOS E RESULTADOS

Este capítulo consiste de duas partes. A primeira descreve o projeto dos experimentos feitos para testar o método proposto. É apresentada a base de dados usada, assim como a metodologia do experimento e os parâmetros que foram observados para comparação. Na segunda parte, os resultados dos experimentos são mostrados e discutidos, em termos de média, desvio pedrão e intervalo de confiança.

5.1 Experimentos

Por uma questão de simplicidade, cinco bases de dados foram criadas no sistema de banco de dados PostgreSQL que se diferenciam somente em tamanho. Cada base de dados consiste de uma única tabela composta por dois atributos: um do tipo serial e um do tipo *character varying*. As tabelas foram populadas com valores aleatórios para simular diversas operações de inserção e remoção. É importante notar que atributos do tipo serial não possuem valores duplicados.

Os experimentos são iniciados aplicando o método DB-Fuzzy, como descrito no Capítulo 4. Para fins de teste, o tamanho ideal para as sub-tabelas foi definido em 1000. Esse processo é chamado de inicialização. Então, busca-se por mil registros aleatórios presentes na base da dados. O tempo gasto no processo de inicialização e nas buscas de cada registro foram medidos.

Para comparação, o método descrito em Patil *et al.* (2012) foi escolhido dentre os métodos encontrados na Revisão Sistemática apresentada no Capítulo 3, por ser o que mais se assemelha ao método DB-Fuzzy. Assim, o método de Patil *et al.* (2012) foi adaptado para usar dígitos ao invés de letras do alfabeto. O número de compartimentos foi definido em 20 e os registros organizados de acordo com o atributo do tipo serial. Para esse método, somente o tempo de buscar mil registros aleatórios foi medido.

Por fim, uma comparação estatística entre os tempos de busca de ambos os métodos é feita usando o método *bootstrap*, isso é, não é feita nenhuma suposição paramétrica sobre a população que gerou a amostra aleatória. Os passos para a metodologia básica do *bootstrap* são mostrados no Algoritmo 7, adaptado de Martinez e Martinez (2015), em que $\hat{\theta}$ é uma estimativa de uma parâmetro θ, $x^{*b} = x_1^{*b}, \ldots, x_n^{*b}$ é uma nova amostra obtida com reposição da amostra inicial x, e $\hat{\theta}^{*b}$ é a estimativa para a b-ésima base de dados. B é configurado para 2000. A confiança usada foi 95%.

O tamanho de um intervalo de confiança expressa sua precisão, um intervalo menor implica uma estimativa mais precisa (MONTGOMERY; RUNGER, 2018). Ao comparar

Algorithm 7: Bootstrap Básico

1 Dada uma amostra aleatória, $x = (x_1, \ldots, x_n)$, calcule $\hat{\theta}$;
2 Selecione uma amostra com reposição a partir da amostra original para encontrar $x^{*b} = x_1^{*b}, \ldots, x_n^{*b}$;
3 Calcule a mesma estatística usando a amostra *bootstrap* do passo 2 para encontrar $\hat{\theta}^{*b}$;
4 Repita os passos 2 e 3, B vezes;
5 Use essa estimativa da distribuição para obter o intervalo de confiança;

dois intervalos, eles pode se intersectar ou não. Se eles intersectam, esses intervalos são considerados estaticamente iguais. Se o limite superior de um intervalo é menor que o limite inferior de um segundo intervalo, o primeiro intervalo é considerado melhor que o segundo.

O computador usado para os experimentos possui a seguinte especificação:

- CPU: Intel(R) Core(TM) i5-8265U;
- RAM: 8,00GB;
- Sistema Operacional: Windows 10 Home;
- Versão do Java: 1.8.0_231; Java HotSpot(TM) 64-Bit Server VM 25.231-b11.

5.2 Resultados e Discussão

O tempo que o método DB-Fuzzy leva para reorganizar o banco de dados em sub-tabelas, chamado de tempo de inicialização, é mostrado na Tabela 11. Como esperado, o tempo de inicialização é diretamente proporcional ao tamanho do banco de dados. O método proposto por Patil *et al.* (2012) também precisa de tempo para organizar o banco de dados, porém esse processo não é automatizado e depende da habilidade de quem o faz.

Tabela 11 – Tempo de inicialização do método DB-Fuzzy

Banco de Dados	Tempo de Inicialização
10k	0.085 h
50k	0.426 h
100k	0.803 h
500k	5.289 h
1000k	9.587 h

Fonte: Autoria Própria (2020).

Foi medido o tempo de buscar, representado como x, um registro mil vezes para ambos os métodos. As medidas de média ($\bar{x}$), desvio padrão (σ) e intervalo de confiança

(IC) são mostrados na Tabela 12. Além disso, gráficos dos intervalos de confiança são mostrados na Figura 18 para melhor visualizá-los.

Tabela 12 – Medidas de Média, Desvio Padrão e Intervalo de Confiança de ambos os métodos

DB	Método DB-Fuzzy			Método de Patil *et al.* (2012)		
	$\bar{x}$	σ	IC	$\bar{x}$	σ	IC
10k	**0.0248 s**	**0.0031**	**[0.0245 : 0.0253]**	0.0360 s	0.0101	[0.0359 : 0.0385]
50k	**0.0241 s**	**0.0024**	[0.0238 : 0.0245]	0.0247 s	0.0036	[0.0244 : 0.0253]
100k	**0.0244 s**	**0.0015**	**[0.0242 : 0.0246]**	0.0539 s	0.0072	[0.0530 : 0.0548]
500k	**0.0271 s**	**0.0013**	**[0.0269 : 0.0272]**	0.0526 s	0.0063	[0.0518 : 0.0534]
1000k	**0.0239 s**	**0.0021**	**[0.0237 : 0.0243]**	0.0490 s	0.0113	[0.0476 : 0.0504]

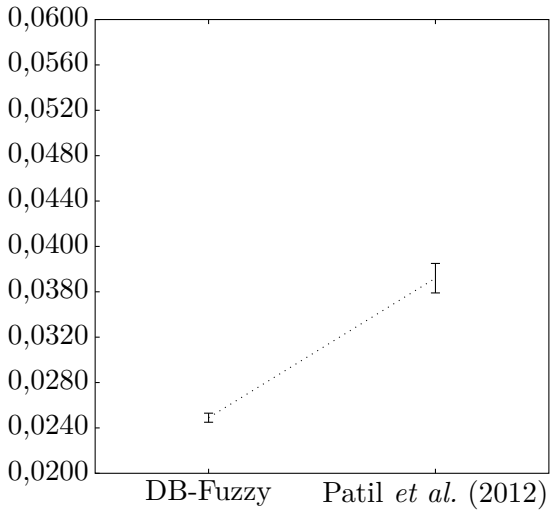
Fonte: Autoria Própria (2020).

Os intervalos de confiança do método DB-Fuzzy e do método proposto por Patil *et al.* (2012) não se intersectam nas bases de dados de 10k, 100k, 500k e 1000k, o que significa que eles não são estatisticamente iguais. Na Tabela 12, o melhor intervalo para cada base de dados está destacado em negrito. Nessas bases, o método DB-Fuzzy tem um tempo de busca melhor já que o limite superior de seu intervalo de confiança é menor que o limite inferior do intervalo do método de Patil *et al.* (2012), com 95% de significância. Já que os intervalos intersectam, não se pode chegar a nenhuma conclusão sobre o tempo de busca na base de 50k. Ademais, o tamanho dos intervalos do método DB-Fuzzy é menor em todas as bases de dados, o que significa que esses intervalos são mais precisos. Assim, é provado estatisticamente que o tempo de busca do método DB-Fuzzy é menor que ou igual ao método proposto por Patil *et al.* (2012).

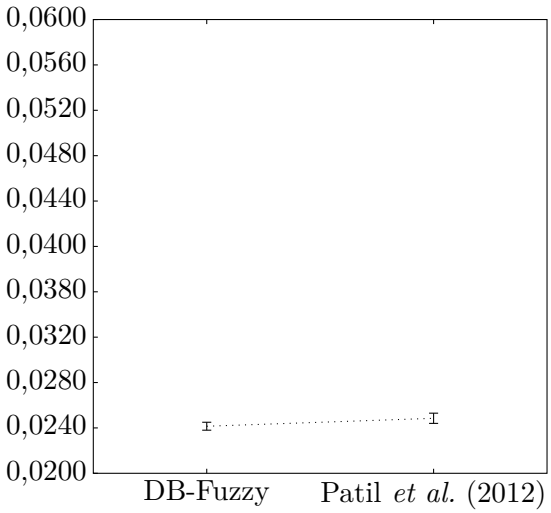
Outro dado presente na Tabela 12 é em relação a média e desvio padrão do tempo de busca. Dado que o método proposto foi considerado melhor ou igual ao método usado para comparação, analisando as medidas da média e do desvio padrão, o tempo de busca não sofre variação significativa quando o tamanho da base muda, o que prova, experimentalmente, que o método DB-Fuzzy tem complexidade constante. Além disso, o método DB-Fuzzy pode ser estendido e adaptado para lidar com diferentes tipos de chave, incluindo chaves múltiplas. O único requerimento para seu uso é que uma medida de distância adequada seja implementada.

Figura 18 – Intervalos de Confiança.

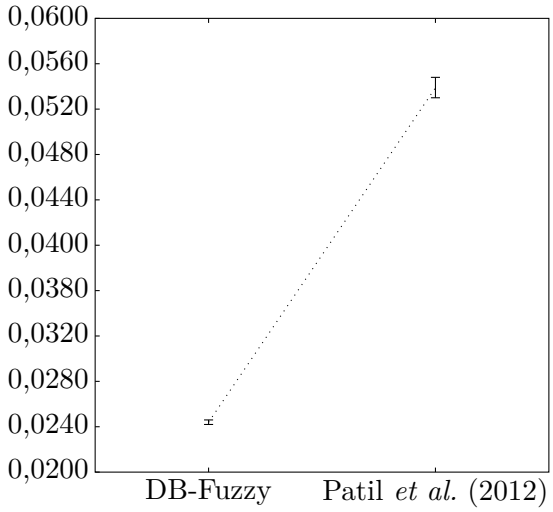
(a) Base de dados 10k.



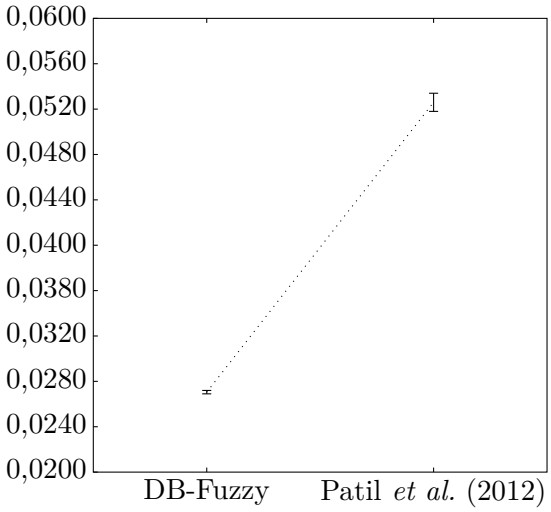
(b) Base de dados 50k.



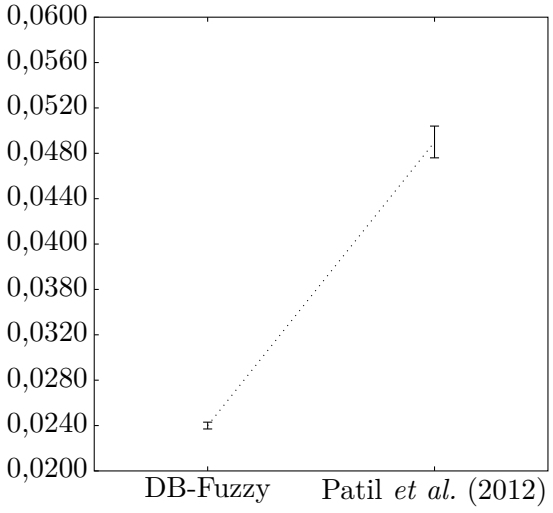
(c) Base de dados 100k.



(d) Base de dados 500k.



(e) Base de dados 1000k.



Fonte: Autoria Própria (2020).

6 CONCLUSÃO

Este trabalho apresenta uma nova abordagem para buscar registros em bancos de dados. Desse modo, foi proposto um método, descrito nesse documento, com o objetivo de resolver esse problema. O método chamado DB-Fuzzy é baseado em tabelas de dispersão e utiliza conceitos relacionados a conjuntos nebulosos para definir uma função de mapeamento. Além disso, as funções básicas de um banco de dados precisaram ser adaptadas para a natureza do método. Para validação, foi feita uma comparação estatística com outro método presente na literatura.

Os principais resultados obtidos com esta pesquisa, podem ser organizados seguindo os objetivos específicos apresentados no **Capítulo 1**.

Investigar o estado da arte relacionado à busca em bancos de dados.

Com o propósito de obter o estado da arte dos métodos, algoritmos, técnicas ou abordagens que minimizassem o tempo de consulta a bancos de dados foi feita uma revisão sistemática de literatura. Nessa revisão, foi identificado que a maior parte dos estudos usam alguma medida de similaridade, geralmente para um tipo de dado específico, ou técnica de indexação para melhorar as consultas em bancos de dados.

Definir o método para diminuição do tempo de busca em bancos de dados

O método desenvolvido parte do pressuposto que bases de dados menores precisam de menos tempo para recuperar um registro específico. Então, uma base de dados grande foi dividida em bases menores e uma função de mapeamento foi criada para decidir em qual dessas bases menores os registro serão alocados. A função de mapeamento utiliza conceitos de lógica fuzzy.

Validar o método proposto

Para validação do método, foi feita uma comparação estatística com outro método presente na literatura. A variável usada para comparação foi o tempo de busca e foram usadas as estatísticas média e intervalo de confiança calculados a partir de uma população criada de acordo com o método *bootstrap* definido em Martinez e Martinez (2015). Foram feitos cinco experimentos e, com exceção de um, os intervalos de confiança do método proposto são melhores.

Assim, a principal contribuição apresentada neste trabalho é o método para diminuição do tempo de busca em bancos de dados relacionais, que leva um tempo constante para a busca de um registro, apresentado no Capítulo 4. A Revisão Sistemática de Literatura

em métodos para redução de tempo em consultas de bancos de dados, apresentada no Capítulo 3, também é uma contribuição interessante desta pesquisa em sua constatação do estado da arte. Apesar disso, o método tem como limitação que a busca só pode ser realizada para um registro individualmente e ser necessário conhecer a chave previamente.

Em trabalhos futuros, seria interessante testar outras medidas de distância, como a euclidiana ou a dos cossenos, para calcular o grau de pertinência e verificar a influência dessa mudança no tempo gasto na inicialização. A implementação de novas medidas de distância é, também, a chave para adaptar o método para bancos de dados não-relacionais. No decorrer dos experimentos, o parâmetro tamanho da sub-tabela foi definido em 1000, porém ainda é necessário um estudo que analise o comportamento do método com a variação desse parâmetro. Outro trabalho futuro interessante, seria o método em bases de dados ainda maiores.

Referências

AMARAL, F. *Introdução À Ciência de Dados: mineração de dados e big data*. [S.l.]: ALTA BOOKS, 2016. ISBN 9788576089346.

ANAND, V.; RAO, C. M. Mongodb and oracle nosql: A technical critique for design decisions. In: *2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)*. [S.l.: s.n.], 2016. p. 1–4.

AOYAMA, K. *et al.* Double-layer neighborhood graph based similarity search for fast query-by-example spoken term detection. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2015. p. 5216–5220. ISSN 1520-6149.

AZEVEDO, E.; CONCI, A.; LETA, F. *Computação gráfica - volume 2: Teoria e prática*. [S.l.]: Elsevier Editora Ltda., 2018. (Computação gráfica). ISBN 9788535223293.

BAYER, R.; MCCREIGHT, E. M. Organization and maintenance of large ordered indexes. *Acta Informatica*, v. 1, n. 3, p. 173–189, Sep 1972. ISSN 1432-0525.

BIOLCHINI, J. C. de A. *et al.* Scientific research ontology to support systematic review in software engineering. *Adv. Eng. Inform.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 21, n. 2, p. 133–151, abr. 2007. ISSN 1474-0346. Disponível em: <http://dx.doi.org/10.1016/j.aei.2006.11.006>.

BONNY, T.; SOUDAN, B. Computation time reduction to speed-up the database searching process. In: *2015 IEEE International Symposium on Multiple-Valued Logic*. [S.l.: s.n.], 2015. p. 121–126. ISSN 0195-623X.

BURGER, W.; BURGE, M. *Digital Image Processing: An Algorithmic Introduction Using Java*. [S.l.]: Springer London, 2009. (Texts in Computer Science). ISBN 9781846289682.

CARVALHO, A. V. de; OLIVEIRA, M. A.; ROCHA, A. Retrieval of very large temporal datasets for interactive tasks. In: IEEE. *Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on*. [S.l.], 2013. p. 1–7.

CARVALHO, A. V. de; OLIVEIRA, M. A.; ROCHA, A. Improvements to efficient retrieval of very large temporal datasets with the travellight method. In: *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2014. p. 1–7. ISSN 2166-0727.

CHEN, L.; HWANG, K.; WU, J. Mapreduce skyline query processing with a new angular partitioning approach. In: *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum*. [S.l.: s.n.], 2012. p. 2262–2270.

COMER, D. Ubiquitous b-tree. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 11, n. 2, p. 121–137, jun. 1979. ISSN 0360-0300.

CORMEN, T. *et al. Introduction to Algorithms*. [S.l.]: MIT Press, 2009. (Computer science). ISBN 9780262533058.

COX, E. *The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, and Maintaining Fuzzy Systems*. [S.l.]: AP Professional, 1994. (The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, and Maintaining Fuzzy Systems, v. 1). ISBN 9780121942700.

DATE, C. *Introdução a sistemas de bancos de dados*. [S.l.]: ELSEVIER EDITORA, 2004. ISBN 9788535212730.

DROZDEK, A. *Estrutura de dados e algoritmos em C++*. [S.l.]: Pioneira Thomson Learning, 2005. ISBN 8522102953.

ELMASRI, R.; NAVATHE, S. *Sistemas de banco de dados*. [S.l.]: ADDISON WESLEY BRA, 2011. ISBN 8579360854.

FUNAKI, K. *et al.* Evaluation of parallel indexing scheme for big data. In: *2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence*. [S.l.: s.n.], 2015. p. 148–153.

GAN, G.; MA, C.; WU, J. *Data Clustering: Theory, Algorithms, and Applications"*. [S.l.]: Society for Industrial and Applied Mathematics, 2007. (ASA-SIAM Series on Statistics and Applied Probability). ISBN 9780898716238.

GANTZ, J.; REINSEL, D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, v. 2007, n. 2012, p. 1–16, 2012.

GIANGRECO, I.; KABARY, I. A.; SCHULDT, H. Adam - a database and information retrieval system for big multimedia collections. In: *2014 IEEE International Congress on Big Data*. [S.l.: s.n.], 2014. p. 406–413. ISSN 2379-7703.

GONZALEZ, R. C.; WOODS, R. E. *Processamento Digital de Imagens*. [S.l.]: Pearson Prentice Hall, 2010. ISBN 9788576054016.

HAYASHI, T.; ISHII, N.; YAMAGUCHI, M. Fast music information retrieval with indirect matching. In: *2014 22nd European Signal Processing Conference (EUSIPCO)*. [S.l.: s.n.], 2014. p. 1567–1571. ISSN 2076-1465.

HSUEH, Y.; HASCOET, T. Caching support for skyline query processing with partially ordered domains. *IEEE Transactions on Knowledge and Data Engineering*, v. 26, n. 11, p. 2649–2661, Nov 2014. ISSN 1041-4347.

KITCHENHAM, B. *Procedures for Performing Systematic Reviews*. Department of Computer Science, Keele University, UK, 2004.

KLIR, G.; YUAN, B. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. [S.l.]: Prentice Hall PTR, 1995. ISBN 9780131011717.

KOSKO, B. Fuzziness vs. probability. *International Journal of General Systems*, Taylor & Francis, v. 17, n. 2-3, p. 211–240, 1990.

KOSKO, B. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991. ISBN 0-13-611435-0.

LAZZERINI, B.; JAIN, L.; DUMITRESCU, D. *Fuzzy Sets & their Application to Clustering & Training*. [S.l.]: CRC Press, 2000. (International Series on Computational Intelligence). ISBN 9781482273977.

LI, J.-C. Boos ting multiple hash tables to search. In: *2012 International Conference on Machine Learning and Cybernetics*. [S.l.: s.n.], 2012. v. 1, p. 57–62. ISSN 2160-1348.

MACHADO, F. *Banco de Dados - Projeto e Implementação*. [S.l.]: Saraiva Educação S.A., 2018. ISBN 9788536509846.

MARTINEZ, W.; MARTINEZ, A. *Computational Statistics Handbook with MATLAB, Third Edition*. [S.l.]: Taylor & Francis, 2015. (Chapman & Hall/CRC Computer Science & Data Analysis). ISBN 9781466592735.

MONTGOMERY, D.; RUNGER, G. *Applied Statistics and Probability for Engineers*. [S.l.]: Wiley, 2018. ISBN 9781119400332.

MORRIS, R. Scatter storage techniques. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 11, n. 1, p. 38–44, jan. 1968. ISSN 0001-0782.

NGUYEN, B. *et al.* Integrating spatial information into inverted index for large-scale image retrieval. In: *2014 IEEE International Symposium on Multimedia*. [S.l.: s.n.], 2014. p. 102–105.

OLIVEIRA, M. M. A. *et al.* Um estudo comparativo entre banco de dados orientados a objetos, bancos de dados relacionais e framework para mapeamento objeto/relacional, no contexto de uma aplixação web. *HOLOS*, 2015. ISSN 1518-1634.

OLIVEIRA, P. F. d.; GUERRA, S.; MCDONNELL, R. *Ciência de dados com R: Introdução*. [S.l.]: Editora IBPAD, 2018. ISBN 978-85-54230-00-5.

PATIL, P. S. *et al.* Customised approach for efficient data storing and retrieving from university database using repetitive frequency indexing. In: *2012 1st International Conference on Recent Advances in Information Technology (RAIT)*. [S.l.: s.n.], 2012. p. 511–514.

PREISS, B. *Estruturas de dados e algoritmos: padrões de projetos orientados a objetos com Java*. [S.l.]: ELSEVIER EDITORA, 2000. ISBN 85711006930.

SOMMERVILLE, I. *Engenharia de software*. [S.l.]: PEARSON BRASIL, 2011. ISBN 9788579361081.

SZWARCFITER, J.; MARKENZON, L. *Estruturas de dados e seus algoritmos*. [S.l.]: Livros Tecnicos e Cientificos, 2010. ISBN 9788521617501.

TAKANO, K. *et al.* An adaptive search system using heterogeneous document vector spaces. In: *2009 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*. [S.l.: s.n.], 2009. p. 193–198. ISSN 1555-5798.

VYAWAHARE H. R.; KARDE, P. P.; THAKARE, V. M. A hybrid database approach using graph and relational database. In: *2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)*. [S.l.: s.n.], 2018. p. 1–4.

WEBER, R.; SCHEK, H.-J.; BLOTT, S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: *VLDB*. [S.l.: s.n.], 1998. v. 98, p. 194–205.

WU, T. *et al.* Optimizing the query performance of block index through data analysis and i/o modeling. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. New York, NY, USA: ACM, 2017. (SC '17), p. 12:1–12:10. ISBN 978-1-4503-5114-0. Disponível em: <http://doi.acm.org/10.1145/3126908.3126934>.

YANG, L. Uniformization of discrete data. In: DENG, X.; DU, D.-Z. (Ed.). *Algorithms and Computation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 453–462. ISBN 978-3-540-32426-3.

YU, C.; MENG, W. Progress in database search strategies. *IEEE Software*, v. 11, n. 3, p. 11–19, 1994.