Estudo comparativo de indexação de texto completo para recuperação de informações em sistemas gerenciadores de banco de dados

Comparative study of full-text indexing for information retrieval in database management systems

Edson Marchetti da Silva
Doutor em Ciência da Informação pela Universidade Federal de Minas Gerais – UFMG.
Professor no Centro Federal de Educação Tecnológica de Minas Gerais – CEFET/MG.
E-mail: edson@cefetmg.br

Lucas Meneses Mardegan
Especilialista em Banco de Dados pelo Centro Federal de Educação Tecnológica de Minas Gerais – CEFET/MG.
E-mail: lucasssm@hotmail.com

Resumo

A indexação completa de textos é, uma funcionalidade também implementada nos Sistemas Gerenciadores de Bancos de Dados (SGBD), que possibilita a pesquisa e recuperação de informações em documentos de textos de forma eficiente. Dessa forma, foram estudadas e comparadas às características da indexação completa de textos em três produtos de Sistemas Gerenciadores de Banco de Dados (SGBD): Microsoft SQL Server, Oracle Database e PostgreSQL. O objetivo é descrever e comparar as soluções, sob a perspectiva de características das funcionalidades e desempenho da carga da indexação completa de textos visando realizar, a posteriori, a busca por palavras-chave. Diante desse contexto, apresentam-se a fundamentação teórica sobre o processo de indexação automática, e os algoritmos utilizados por essas ferramentas para calcular a ordenação por relevância dos documentos retornados pela busca. Os resultados demonstram que à escolha de um SGBD irá depender da aplicabilidade do mesmo, sendo fundamentada pelo tipo de documento a ser indexado, funcionalidades implementadas no SGBD e orçamento para investimento em software.

Palavras-chave: Indexação automática. Recuperação de informação. Relevância. Bases de dados de textos completos.

Abstract

Full-text indexing is a feature also implemented in Database Management Systems (DBMS), which makes it possible to efficiently search and retrieve information in text documents. In this way, we have studied and compared the characteristics of full-text indexing in three DBMS products: Microsoft SQL Server, Oracle Database and PostgreSQL. The objective is to describe and compare the solutions, from the perspective of features characteristics and load performance of the complete indexing of texts aiming to carry out, afterwards, the search for keywords. In this context, we present the theoretical basis for the automatic indexing process, and the algorithms used by these tools to calculate the order by relevance of the documents returned by the search. The results demonstrate that the choice of a DBMS will depend on the applicability of the same, being based on the type of document to be indexed, functionalities implemented in the DBMS and budget for software investment.

Keywords: Automatic indexing. Information retrieval. Relevance. Full-text database.

DOI: 10.11606/issn.2178-2075.v10i1p281-301

1. Introdução

O presente trabalho foi inspirado pela crescente demanda de integrar aos sistemas de informações convencionais, aos mecanismos de busca por palavras-chave em conteúdos textuais. Beall (2008), em seu artigo intitulado: “The weaknesses of full-text searching”, destaca que se uma coleção de documentos é indexada apenas por indexação de texto completo, sem incluir a busca por metadados, isso pode gerar insatisfação dos bibliotecários, por pagar para ter acesso a uma base de conteúdos que é difícil de recuperar informação de forma ágil e precisa. Por trás do processo de indexação de texto completo, estão às técnicas de indexação, que viabilizam pesquisas com tempo de resposta instantânea e resultados classificados por relevância. Adicionalmente, um sistema de busca deve possibilitar, concomitantemente, o uso de filtros avançados por metadados. Portanto, este trabalho se justifica, por apresentar e comparar SGBD, os quais são uma plataforma em que a junção de ambos conceitos é de fácil implementação. Entretanto, este estudo se limita em avaliar o funcionamento da implementação da funcionalidade de indexação de texto completo no contexto dos SGBD, tendo em vista que filtrar metadados já é a funcionalidade fim dos mesmos.

Segundo Borges (2009), podemos assumir que indexar é a atividade de representar um documento através de uma descrição abreviada de seu conteúdo. Ou seja, documentos são selecionados e indexados, de tal forma que suas representações sejam registradas e armazenadas. O resultado desse processo é a possibilidade da recuperação de informação (RI) pelo usuário de forma ágil e precisa. Assim, neste trabalho, assume-se que a indexação completa de textos pode ser entendida como uma funcionalidade que possibilita ao usuário realizar consultas por termos, em conteúdo de texto em linguagem natural, os quais foram previamente normalizados e indexados possibilitando serem recuperados através do melhor casamento entre os termos da consulta com esses conteúdos textuais. Nesse contexto, buscou-se comparar três SGBD distintos, e as formas deles lidarem com à indexação de texto completo.

Portanto, o objetivo deste trabalho é analisar comparativamente a funcionalidade de indexação de texto completo (full-text search) e a recuperação de informação a partir de palavras-chave, considerando três SGBD, Microsoft SQL Server, Oracle Database e PostgreSQL, destacando-se as principais características implementadas por cada um deles e mensurando o desempenho do processo de indexação que possibilita a busca por palavras-chave.

2. Fundamentos conceituais

Nesta seção, são apresentados alguns conceitos gerais sobre o processo de RI. Logo após, apresentam-se os fundamentos do processo de indexação automatizado.

2.1. Conceitos gerais sobre o processo de RI

Devido à evolução do conhecimento humano, o qual resulta em um grande volume de informações de aspectos intelectuais e a premissa de conservação do conhecimento intrínseco nessas informações, emergiu-se a necessidade de organizar as coleções de informações para recuperá-las quando demandadas. Conforme afirma Alvarenga (2003), para permitir a RI aos usuários, os profissionais da representação do conhecimento, bibliotecários e demais profissionais da informação, lidam com a tarefa de tratar e organizar conhecimento visando facilitar o acesso a esses documentos.

> No processamento técnico tradicional, o documento vem sendo representado por um conjunto de informações relativas à sua descrição física e pontos de acesso (índices) pertinentes, representação esta preparada e armazenada em um contexto físico independente do documento primário. As informações nesse tipo de representação compreendem compactações que tentam descrever as características do documento, refletindo sua origem e conteúdo, facilitando sua recuperação. (ALVARENGA, 2003).

Dessa forma, a RI consiste em encontrar informações não-estruturadas, em uma grande coleção de documentos em forma de texto, a partir do fornecimento de parte da informação desejada, o que convencionalmente são chamados de descritores de busca. Com o volume de informações produzidas, a relevância e agilidade para se encontrar determinada informação, torna-se cada vez mais, um desafio a ser vencido, no qual a Ciência da Computação tem empenhado papel fundamental, visando automatizar esse processo, no contexto da proliferação da comunicação e difusão da informação em larga escala. Nesse sentido, Saracevic (1996) afirma que: “A base da relação entre CI e ciência da computação reside na aplicação dos computadores e da computação na recuperação da informação, assim como nos produtos, serviços e redes associadas.”

O estudo de Gantz *et al*. (2007), relata que em 2006, a quantidade de informação digital criada e replicada corresponde a três milhões de vezes as informações contidas em todos os livros que já foram escritos. Esses autores explicam ainda que, o crescimento exponencial das informações se deve ao fato de os dados analógicos de voz e imagem serem produzidos

atualmente, em formato digital. Considerando o cenário atual, Gantz e Reinsel (2012) estimam que de 2005 a 2020 esse universo digital irá dobrar a cada dois anos, crescendo em um fator de 300, ou seja, de 130 exabytes para 40.000 exabytes, sendo mais de 5.200 gigabytes para cada homem, mulher e criança em 2020.

Soma-se ao aumento do quantitativo informacional, os desafios de lidar com a linguagem natural, em que se deve considerar a complexidade de pesquisar em documentos de texto, tais como: a polissemia (pluralidade de sentidos); a sinonímia (coincidência ou igualdade de significados entre palavras); a flexão de número (terminação de uma palavra para indicar singular ou plural); a flexão de grau (terminação de uma palavra para indicar tamanho nos substantivos e intensidade nos adjetivos e advérbios); a flexão de tempo (indica o presente, passado e futuro nos verbos); a flexão de modo (indica diferentes atitudes por parte do emissor, sendo indicativo, subjuntivo ou imperativo); a flexão de pessoa (serve para flexionar o verbo de acordo com a pessoa: emissor, receptor, de que se fala ou de quem se fala).

2.2. Fundamentos do processo de indexação automatizado

Segundo Robredo (1982), o processo de indexação automática realizado por sistemas computacionais é similar ao processo de leitura-memorização realizada pelo ser humano, sendo seu princípio baseado na comparação de cada palavra do texto com uma relação de palavras sem um significado determinado. Manning e Schütze (2003, p. 529-531) consideram que a pesquisa em RI lida com o desenvolvimento de algoritmos e modelos para recuperar informações a partir de um repositório de documentos. Para Borges (2009) a indexação automatizada pode ser considerada um modelo de extração com características estatísticas e probabilisticas.

Visando o aperfeiçoamento e agilidade no processo de RI, os Sistemas de Recuperação de Informação (SRI) utilizam a funcionalidade de indexação automatizada, podendo ou não fazer uso de SGBD. Desse modo, esses sistemas tratam a coleção de documentos fazendo a normalização e a indexação desses. Lancaster & Warner (1993) destacam que os SRI são uma interface entre uma coleção de informações, em meio impresso ou não, e uma população de usuários. Atribui-se as seguintes tarefas aos SRI: aquisição e armazenamento; organização e controle; distribuição e disseminação aos usuários. Os autores Russell & Norvig (2004, p. 813) deram a seguinte definição para o processo de RI: “tarefa de encontrar documentos relevantes

que atendam às necessidades de informação de um usuário". Ainda, segundo esses autores, os SRI se caracterizam por definir:

- o que é o documento a ser recuperado: um parágrafo, uma página, ou um texto completo;
- a forma de consulta: uma lista de palavras, uma sequência de palavras adjacentes em forma de uma frase ou parte de uma frase, se pode conter operadores booleanos e não-booleanos;
- o subconjunto de respostas: as relevantes e as não-relevantes;
- a forma de apresentação do resultado: uma lista ordenada de documentos, um mapa giratório com os resultados apresentados num espaço tridimensional.

Dentre os tipos de SRI, segundo a taxonomia apresentada por Baeza-Yates & Ribeiro-Neto (1999, p.21), destacam-se: (1) modelos clássicos se caracterizam por organizar os documentos de forma que sejam representados por um conjunto de palavras-chave, objetivando refletir o assunto do documento, sumarizando o conteúdo de forma significativa. Eles são subdivididos em Booleano, Vetorial e Probabilístico; (2) modelos estruturados permitem que além das palavras-chave utilizadas pelos modelos clássicos, as buscas sejam realizadas também a partir de partes intrínsecas no conteúdo do documento, como seções presente nos documentos, sejam elas títulos de tabelas e figuras, fontes de letra, entre outras informações. Esses modelos são subdivididos em: Lista não sobreposta e Proximidade dos nós; (3) modelos baseados em navegação, a busca é realizada da forma hipertextual, em um contexto em que, geralmente, não houve uma indexação prévia dos dados. Silva (2013, p.53-54) descreve que independente do modelo utilizado, para viabilizar as consultas a serem realizadas pelos usuários, todo SRI realiza um processo de preparação que antecede à indexação dos documentos, sendo que dentre as etapas desse processo, algumas são obrigatórias e outras são opcionais:

- Documentos: Realiza-se a definição da representação lógica dos documentos contida no *corpus*. Nessa etapa, define-se as informações que serão utilizadas para representar o documento. Exemplos: substantivos, verbos, conjunto de termos que aparecem no documento;

- Reconhecimento da estrutura (opcional): obtenção de informações que podem agregar valor sobre a estrutura do documento. Exemplos: Fontes em negrito e/ou itálico, letras maiúsculas ou minúsculas, separação dos parágrafos e sentenças, títulos;

- Acentos e espaços: realiza-se a padronização dos termos. Exemplo: Transformação da cadeia de caracteres em letras minúsculas, exclusão dos acentos, caracteres especiais e caracteres de formatação;

- Retirada das *stop words* (opcional): realiza-se a remoção das palavras que ocorrem com muita frequência em todos os documentos. Exemplos: remoção dos artigos, preposições, etc;

- Grupo de palavras (opcional): realiza-se o agrupamento de palavras e estruturas sintáticas e gramaticais. Exemplos: utilizam-se técnicas de detecção dos sintagmas nominais ou Lematização (*lemmatizer* em inglês) visando determinar o seu "lema";

- Redução ao prefixo *stemming* (opcional): realiza-se a Radicalização (*stemmer* em inglês) dos termos. Exemplos: redução ao prefixo "encontr" das palavras encontrar, encontrado, encontro;

- Indexação automática ou manual: realiza-se a organização dos termos em uma estrutura de lista invertida no qual cada termo encontrado no *corpus* do documento faça referência aos documentos nos quais o termo é encontrado. O conteúdo de todo o esse processamento é materializado no que chamamos de índice. O índice armazena as referências de organização dos conteúdos dos documentos, agilizando o retorno das respostas às buscas realizadas.

3. Metodologia

A metodologia empregada neste trabalho utilizou uma abordagem dividida em cinco etapas realizadas em momentos distintos e detalhadas a seguir.

Na primeira etapa realizou-se a preparação e a instalação do ambiente virtual e dos SGBD a serem estudados. Todos os SGBD utilizaram o mesmo computador físico / máquina virtual para a realização do presente estudo.

Na segunda etapa foram criadas as estruturas dos bancos de dados, como tabelas e atributos. Os três SGBD foram conceituados utilizando-se o TOAD Data Modeler, ao final foram exportados os *scripts* DDL para criação dessas estruturas nos bancos de dados. Nessa etapa, também foram coletados os textos a serem utilizados para realizar os testes.

Na terceira etapa foram instalados e configurados os pacotes para que os produtos Microsoft SQL Server e Oracle Database funcionassem com pesquisas realizadas em textos carregados nas tabelas do banco de dados. Para o produto PostgreSQL fez-se indispensável realizar uma conversão dos documentos através do software Apache Tika[1]. Também foram criados *scripts* adicionais com *procedure* e *sequence* de acordo com cada SGBD.

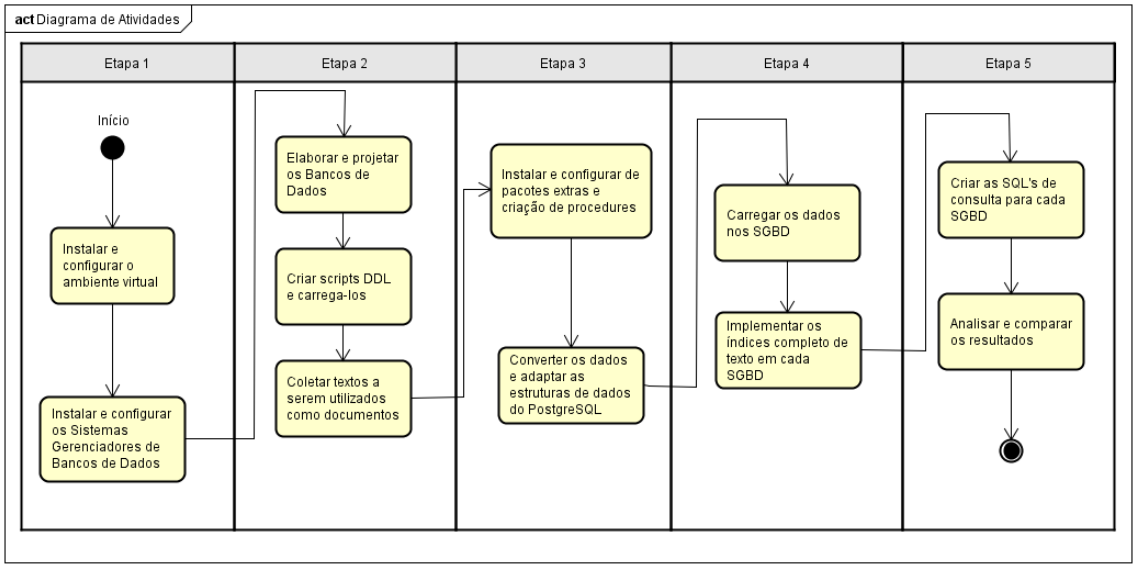
Na quarta etapa foram carregados os dados em cada banco de dados. Um tratamento específico foi dado para cada SGBD que possui uma forma distinta de configurar e implementar seus índices.

Finalmente, na quinta etapa foram criados os arquivos SQL responsáveis pelas pesquisas nos documentos, os quais permitiram testar e consultar os dados indexados pelos bancos de dados, resultando em métricas de desempenho na indexação que foram comparadas em cada SGBD objeto de estudo deste trabalho.

A Figura 1 apresenta um diagrama de atividades, demonstrando o fluxo de trabalho e os procedimentos realizados, de forma geral, em cada etapa do presente estudo.

[1] https://tika.apache.org/ Acesso nov. 2018.

Figura 1 – Passos executados na aplicação da metodologia



Fonte: Elaborado pelos autores.

Dentre os produtos estudados: Microsoft SQL[2], Oracle[3] e PostgreSQL[4], verificou-se que cada SGBD, desenvolveu e implementou seu próprio método para normalizar e indexar os documentos armazenados. As funcionalidades, bem como, as limitações foram relacionadas e comparadas a fim de demonstrar as diferenças e similitudes entre os produtos. A Tabela 1, apresenta as características gerais dos produtos de software, levando-se em consideração os componentes responsáveis pela indexação de texto completo.

Tabela 1 – Quadro comparativo das características

	Nome do Componente	Tipos de Dados	Idiomas Suportados	Características Comuns	Características Específicas	Limitações
Microsoft SQL Server	Full-Text Search	char, varchar, nchar, nvarchar, text, ntext, image, xml, ou varbinary(max) e FILESTREAM.	≅50	- Separadores de palavras e *stemmers;* - Listas de palavras irrelevantes (*stop words*) com possibilidade de edição; - Dicionário de sinônimos com possibilidade de expansão; - Nenhum dos produtos possui a funcionalidade de lematização;	- Cada índice completo de texto indexa uma ou mais colunas, no qual cada coluna pode usar um idioma específico; - Filtragem de documentos (MS Word, PDF, etc.) através do componente iFilter;	- Necessidade de instalação de plug-ins a parte para fazer pesquisas em documentos que a Microsoft não é proprietária. Ex.: arquivos PDF.
Oracle Database	Oracle Text	varchar2 (até 4000 caracteres), CLOB (*Character Large Object*), BLOB (*Binary Large Object*).	≅31		- Filtragem de documentos (MS Word, PDF, etc.); - Indexação direta de protocolos HTTP; - Seccionador de HTML e XML; - Chamada de programas Java e C++ via procedimentos;	- Limitações com fontes incorporadas (*embeedeed fonts)* em arquivos PDF.
PostgreSQL	Full Text Search	tsvector ou tsquery	≅15		- Possibilidade de implementação de dois tipos de índices para realização de pesquisa completa de textos. - GiST (Generalized Search Tree) - colunas podem ser do tipo tsvector ou tsquery; - GIN (Generalized Inverted Tree) - colunas devem ser do tipo tsvector;	- Limita-se a indexar somente textos pré-processados, não indexando arquivos externos como PDF, MS Word e etc;

Fonte: Elaborada pelos autores.

[2] https://www.microsoft.com/pt-br/sql-server/sql-server-2017-editions Acesso nov. 2018.
[3] https://www.oracle.com/br/index.html Acesso nov. 2018.
[4] https://www.postgresql.org Acesso nov. 2018.

4. Cálculo de Relevância

O cálculo de relevância das respostas, pode ser realizado por diferentes técnicas, que visam ordenar as respostas, apresentando primeiro os documentos mais similares aos descritores da busca. A Tabela 2, apresenta o nome da função utilizada para calcular a relevância para cada SGBD comparado neste estudo.

Tabela 2 – Funções de similaridade disponibilizadas pelos SGBD.

	Cálculo de Relevância Utilizado
Microsoft SQL Server	**Funções de Similaridade:** CONTAINSTABLE, ISABOUT e FREETEXTTABLE[5]
Oracle Database	**Funções de Similaridade:** SCORE (DEFINESCORE), SCORE (DEFINEMERGE) E ABOUT[6]
PostgreSQL	**Funções de Similaridade:** TS_RANK; TS_RANK_CD; PG_SIMILARITY; Função UDF[7]

Fonte: Elaborada pelos autores.

O Microsoft SQL Server, implementa três funções de similaridade para realizar o cálculo de relevância, CONSTAINSTABLE, ISABOUT e FREETEXTTABLE, as quais produzem uma pontuação para cada resposta encontrada pelo SGBD. A função CONSTAINSTABLE retorna uma tabela, com zero ou mais linhas, que contêm uma correspondência aproximada com palavras, frases e proximidades de palavras (aquelas com uma certa distância entre os termos de busca inicial). Um valor de classificação de relevância *Rank* para cada linha é retornada, podendo variar entre 0 a 1000, o qual indica quão bem a linha corresponde com a busca em questão. A Expressão 1, apresenta a fórmula utilizada para calcular a pontuação ao utilizar a função CONSTAINSTABLE:

```
StatisticalWeight = Log2( ( 2 + IndexedRowCount ) / KeyRowCount )
Rank = min( MaxQueryRank, HitCount * 16 * StatisticalWeight / MaxOccurrence )
```

(1)

A função ISABOUT, produz uma consulta baseada no algoritmo de Jaccard, que é calculada pela Expressão 2.

```
ContainsRank = same formula used for CONTAINSTABLE ranking of a single term (above).
Weight = the weight specified in the query for each term. Default weight is 1.
WeightedSum = Σ[key=1 to n] ContainsRankKey * WeightKey
Rank =  ( MaxQueryRank * WeightedSum ) / ( ( Σ[key=1 to n] ContainsRankKey^2 )
      + ( Σ[key=1 to n] WeightKey^2 ) - ( WeightedSum ) )
```

(2)

O SQL Server ainda possibilita à utilização de uma outra função de similaridade, conhecida como FREETEXTTABLE. Essa função é baseada na fórmula de classificação OKAPI BM25, técnica essa proposta por Robertson & Spark (1976). Adicionalmente, essa função utiliza um método para expandir os termos de busca, via geração de inflexão, de forma à agregar novos termos flexionados a partir das palavras originais da consulta. Esses sinônimos gerados são tratados como termos separados e igualmente ponderados. Cada palavra encontrada na busca é contabilizada no resultado utilizado para realizar a ordenação por relevância.

No Oracle o cálculo de relevância é implementado através de três funções de similaridade. A primeira delas é baseada na fórmula de Salton e Buckley (1988). Ela é nomeada como DEFINESCORE, e utiliza o Term Frequency/Inverted Document Frequency (TF/IDF), que retorna valores de pontuação entre 0 e 100. Ou seja, considera que, os termos pesquisados que ocorrem frequentemente em vários documentos são considerados com baixa pontuação, por sua vez, os termos pesquisados que ocorrem com frequência em um documento, mas raramente na coleção de documentos, são considerados com alta pontuação. (ORACLE TEXT REFERENCE, 2015).

A Tabela 3, apresenta alguns dos argumentos da função DEFINESCORE.

Tabela 3 – Critérios do predicado DEFINESCORE

NOME	DESCRIÇÃO
DISCRETE	Se o termo existir no documento, pontuação = 100. Caso contrário, pontuação = 0.
OCCURRENCE	Pontuação baseada no número de ocorrências.
RELEVANCE	Score baseado na relevância do documento.
COMPLETION	Pontuação baseada na cobertura. Os documentos terão um *score* mais alto caso a relação entre o número de termos correspondentes e o número de todos os termos na seção (incluindo as *stop words*) for maior. Somente aplicável com o operador WITHIN.
IGNORE	Para ignorar a pontuação deste termo.

Fonte: ORACLE, 2011.

A segunda função é a DEFINEMERGE, a qual define a pontuação entre os nós com os operadores AND ou OR. A pontuação pode ser mesclada de três formas: escolhendo o valor mínimo, escolhendo o valor máximo ou calculando a pontuação médias de todos os nós filhos.

Por fim, a terceira alternativa utiliza a função ABOUT. Nesse caso, ela tem a limitação de ser dependente do idioma. Ou seja, a execução da função demanda por interpretar conceitos a partir de uma base de dados de conhecimento, a qual atualmente está disponível apenas para os idiomas inglês e francês (ORACLE TEXT REFERENCE, 2011).

O PostgreSQL implementa função de busca booleana, a qual não permite a ordenação por relevância, e as funções por similaridade que produzem respostas ordenadas de forma

decrescente por relevância, são elas: TSRANK, TSRANK_CD e PG_SIMILARITY. Adicionalmente, o Postgres, permite o uso de Funções Definidas pelo Usuário (UDF). As funções TSRANK e TSRANK_CD, consideram a frequência em que o termo aparece no documento para determinar o quão importante ele é. Ademais, também pode-se considerar em que parte do texto cada termo aparece para realizar o cálculo. Especificamente a função TSRANK_CD, ainda computa a densidade de correlacionamento (*cover density)*, descrita por Clarke, Comack & Tudhope (2000), que define quão próximos estão os termos no documento.

As funções de relevância também podem receber um número passado como argumento, tal como: 0, 1, 2, 4, 8, 16, 32. O valor passado define qual técnica será utilizada no cálculo de relevância. O significado de cada opção é apresentado na Tabela 4.

Tabela 4 – Opções que definem a técnica de classificação

OPÇÃO	DESCRIÇÃO
0	Padrão. Ignora o tamanho do documento;
1	Divide a classificação por 1 + o logaritmo do tamanho do documento;
2	Divide a classificação pelo tamanho do documento;
4	Divide a classificação pela distância do significado harmônico entre as proximidades (somente implementado pela função TSRANK_CD);
8	Divide a classificação pelo número de palavras únicas no documento;
16	Divide a classificação por 1 + o logaritmo do número de palavras únicas do documento;
32	Divide a classificação por ela mesma +1.

Fonte: POSTGRESQL, 2007.

Ambas funções consideram as características estruturais do documento pesquisado, no qual, o peso varia de acordo com a parte em que os termos foram encontrados no documento, tal como: título, resumo, corpo etc. O argumento de peso (*weight*), oferece a habilidade de fazer com que a busca, considere mais ou menos relevante um termo dependendo em qual parte ele se encontra no documento. A ordem crescente dos pesos é D, C, B, A. A função *setweight* é que define o peso de entrada dos lexemas, de uma subparte do documento. Exemplo: Título pode ter um maior peso, em segundo lugar, o resumo e em terceiro o restante do documento. O valor default atribuído para o peso, por categoria é: {D=0,1; C=0,2; B=0,4 e A=1,0}

A função PG_SIMILARITY, deve ser instalada e configurada a parte, e as funções UDF devem ser criadas conforme a necessidade do usuário.

5. Apresentação e Análise dos Resultados

Nesta seção, são apresentados os passos detalhados no percurso adotado, visando alcançar os objetivos propostos, bem como, os resultados oriundos das buscas por descritores, realizada pelos SGBD estudados.

Inicialmente, foi criado um ambiente virtual, para instalação dos SGBD. Utilizou-se o software VMware Workstation versão 11.1, o qual foi instalado em um computador com processador Intel i7-4790K 4 GHz (8 núcleos), 8 GB de memória RAM, disco SSD de 40GB e com o sistema operacional Windows Server 2008 R2 64 bits instalado. Os três SGBD, objeto deste estudo, foram instalados nas seguintes versões: Oracle Database 11gR2, Microsoft SQL Server 2014 e PostgreSQL 9.3.

Após instalados os três SGBD, eles foram configurados minimamente para permitir o acesso as bases principais do sistema, garantindo que as configurações posteriores para criação da estrutura de dados e carregamento dos bancos com as informações dos documentos selecionados fossem bem-sucedidos.

Após finalizar as configurações iniciais de cada SGBD, os *scripts* de criação das estruturas de dados foram carregados, e iniciou-se a coleta dos documentos que serviram de insumo para popular as tabelas e atributos dos bancos de dados. Os documentos selecionados, totalizaram 15 arquivos, sendo em sua maioria livros escolhidos aleatoriamente.

Para o Microsoft SQL Server, foi necessário baixar, instalar e configurar um *plug-in* da Adobe, o qual permitiu converter para texto os arquivos em formato pdf. Nativamente, o SQL Server fornece suporte apenas para arquivos em formatos proprietários da Microsoft.

Para o Oracle Database, foi necessário configurar um diretório, funcionalidade *directory* do Oracle, para que os arquivos pdf fossem armazenados. A localização, referida anteriormente, tem de ser repassada para a *procedure* de importação, a qual se encarrega de carregar esses arquivos em formato pdf para o banco de dados.

Nenhuma configuração para importação de arquivos em pdf foi realizada no caso do PostgreSQL, afinal, o mesmo só dá suporte à indexação completa de textos em arquivos já previamente convertidos para texto. A definição da estrutura da tabela no PostgreSQL teve que ser alterada, pois cada coluna criada do tipo Tsvector, o qual armazenam texto pré-processado de documentos, tem a capacidade limitada para arquivar até 1Mb de dados. Assim foram criadas

mais duas colunas do tipo Tsvector para armazenar os documentos que ultrapassavam esse limite. Como o comando de carga do Postgres (copy) não converte diretamente conteúdo de texto para o tipo Tsvector, uma *procedure* e uma *trigger* tiveram de ser implementadas para esse fim.

O nome, tamanho em *bytes* dos documentos em Pdf utilizados pelo Microsoft SQL Server e Oracle Database e em TXT utilizados para o Postgres estão descritos na Tabela 5.

Tabela 5 – Documentos caregados nos SGBD

Nome do Documento	PDF tamanho em bytes	TXT Tamanho em bytes
Allan Kardec - A Gênese	1.294.593	539.190
Allan Kardec - O Céu e o Inferno	1.246.382	549.150
Allan Kardec - O Evangelho Segundo o Espiritismo	1.386.978	563.271
Allan Kardec - O Livro dos Espíritos	1.399.717	633.463
Allan Kardec - O Livro dos Médiuns	1.402.918	620.472
Chico Xavier - Evangelho por Emmanuel Completo	7.750.233	2.799.988
Chico Xavier - Humberto de Campos - Boa Nova	599.867	220.845
Chico Xavier - Humberto de Campos - Brasil Coração do Mundo Pátria do Evangelho	540.320	184.739
Chico Xavier - Paulo e Estevão	1.671.485	1.022.355
Janaina Farias - Carlos Drummond de Andrade - Salve Salve	254.614	5.105
Janaina Farias - Jean Lucca - Amor Antigo	146.403	936
Janaina Farias - Jean Lucca – Suportar	146.360	936
Janaina Farias - Machado de Assis - Reflexões Patrióticas	242.357	936
Janaina Farias - Mario de Andrade - O Enigma	146.016	936
Janaina Farias - Tom Jobim - Versos ao Meu Brasil	148.399	936

Fonte: Elaborada pelos autores.

Após realizar a carga a partir de *scripts*, foi implementado em cada SGBD os índices responsáveis pela indexação de texto completo. Novamente, com exceção do PostgreSQL, para o qual não foi necessário a criação de um índice. Apesar da possibilidade em fazer otimizações e configurações nos SGBD, direcionadas à indexação de texto completo, nenhuma configuração extra foi realizada visando melhorar o desempenho dessa funcionalidade.

6. Desempenho da Indexação dos Dados

O desempenho da indexação de texto completa (tempo de carregamento dos SGBD) foi mensurando a partir de cinco carregamentos dos dados, calculando ao final a média entre os resultados obtidos. Destaca-se que para evitar interferências, antes de cada carregamento, os índices eram excluídos e criados novamente. Enquanto os testes eram executados em um SGBD, os demais ficaram desabilitados para não influenciar no processamento do outro.

Na Tabela 6 são apresentados, o tempo gasto em cada tomada e a média geral de todas as tomadas de tempo, considerando o início e a finalização do processo de carga dos dados, bem como, a quantidade de palavras indexadas em cada SGBD.

Tabela 6 – Tempo de Carga x Palavras Indexadas

	CARREGAMENTO / TEMPO (HH:MM:SS)						
SGBD	1º	2º	3º	4º	5º	Média	Palavras Indexadas
Microsoft SQL Server	00:23:20	00:25:42	00:23:13	00:23:02	00:23:31	**00:23:46**	67552
Oracle Database	00:00:11	00:00:10	00:00:11	00:00:11	00:00:11	**00:00:11**	61459
PostgreSQL	00:00:01	00:00:01	00:00:01	00:00:01	00:00:01	**00:00:01**	97676

Fonte: Elaborada pelos autores.

Ao ser comparado com o Oracle, o Microsoft SQL Server apresentou um maior consumo de tempo. Isso pode ser atribuido, à dependência de um componente externo para ler os documentos em pdf, e convertê-los para texto. Afinal, o Oracle Database possui essa funcionalidade intrínseca. Dessa forma, o desempenho do Oracle Database na indexação dos dados foi muito superior ao do Microsoft SQL Server.

Como o PostgreSQL não possui a funcionalidade de indexar arquivos em pdf, sendo necessário realizar uma conversão prévia dos documentos para que a indexação de texto completa possa ser realizada, mesmo assim, ele teve um desempenho extremamente rápido, quando comparado ao Oracle.

Um ponto interessante a ser observado, é que, a quantidade total de palavras indexadas também foram diferentes em cada SGBD, conforme mostra a Tabela 6. Isso ocorre, devido as diferenças do processo de indexação realizado por cada produto. O Microsoft SQL Server e o Oracle Database possuem funcionalidades para criarem um catálogo, o qual é utilizado para armazenar as palavras indexadas a partir dos documentos. Dessa forma, a indexação de cada palavra é realizada apenas uma vez, mesmo se ela possuir múltiplas ocorrências nos documentos. Por outro lado, o PostgreSQL indexa cada documento de forma independente do outro, em um atributo do tipo Tsvector, inserindo uma linha na tabela para cada documento.

Isso explica a quantidade superior de palavras indexadas. Ou seja, um mesmo léxico, pode ser indexado mais de uma vez, desde que esteja em documentos distintos.

7. Comparação das Funcionalidades de Recuperação de Informação

Todas as buscas foram realizadas considerando o mesmo descritor, a palavra “caridade”. O objetivo foi de obter o comportamento da classificação das respostas a partir do cálculo de relevância, implementado por cada SGBD. A consulta SQL utilizada obedeceu a sintaxe adequada a cada produto.

A Figura 2, apresenta o resultado das consultas realizadas no Microsoft SQL Server, utilizando as funções: (a) CONSTAINSTABLE; (b) ISABOUT em conjunto com a função CONTAINSTABLE utilizando um peso de 0.75 como argumento; (c) FREETEXTTABLE. As técnicas de relevância aplicadas para ordenar as respostas foram respectivamente: frequência de ocorrência da palavra com verificação da posição do termo na sentença; coeficiente de Jaccard; e Okapi BM 25, respectivamente.

Figura 2 – Consultas SQL utilizando as funções do Microsoft SQL Server

```
SELECT NOMEARQUIVO_MENSAGEM, KEY_TBL.RANK
FROM MENSAGEM AS FT_TBL
INNER JOIN CONTAINSTABLE(MENSAGEM, DOCUMENTOARQUIVO_MENSAGEM, 'caridade') AS KEY_TBL
ON FT_TBL.ID_MENSAGEM = KEY_TBL.[KEY]
ORDER BY KEY_TBL.RANK DESC;
```

(a) CONSTAINSTABLE

```
SELECT NOMEARQUIVO_MENSAGEM
FROM MENSAGEM
WHERE CONTAINS(DOCUMENTOARQUIVO_MENSAGEM, 'ISABOUT("caridade" WEIGHT (0.75))');
```

(b) ISABOUT em conjunto com CONTAINSTABLE

```
SELECT NOMEARQUIVO_MENSAGEM,KEY_TBL.RANK
FROM MENSAGEM AS FT_TBL
INNER JOIN FREETEXTTABLE(MENSAGEM, DOCUMENTOARQUIVO_MENSAGEM, 'caridade') AS KEY_TBL
ON FT_TBL.ID_MENSAGEM = KEY_TBL.[KEY]
ORDER BY KEY_TBL.RANK DESC;
```

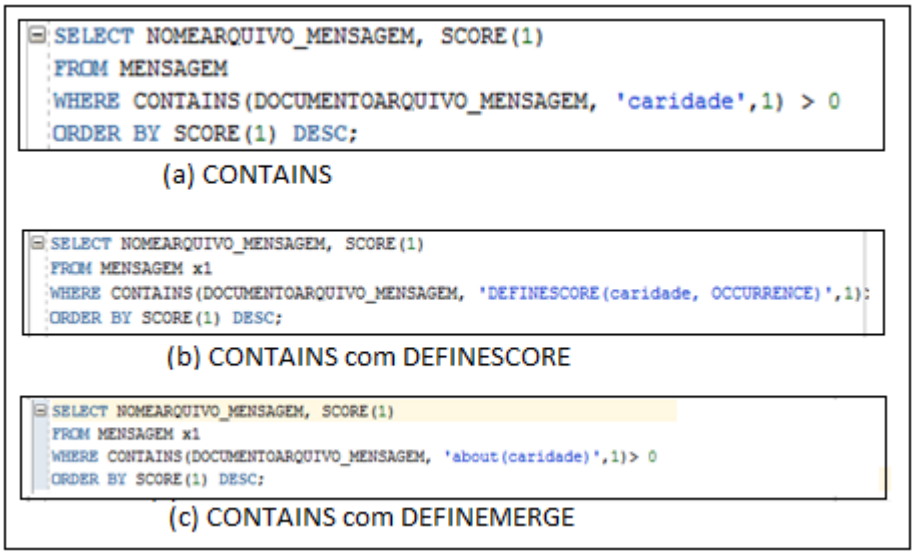
(c) FREETEXTTABLE

Fonte: Elaborada pelos autores

Os resultados obtidos pelas consultas realizadas, utilizando as diferentes funções, são diferentes. Tanto a ordem dos documentos, quanto a pontuação calculada na resposta. Isso demonstra que a ordenação das tuplas foi disposta a partir de uma pontuação, a qual se baseia na fórmula para obter um peso estatístico.

A Figura 3, apresenta as consultas realizadas no Oracle Database, utilizando as funções: (a) CONTAINS; (b) CONTAINS com DEFINESCORE; (c) CONTAINS com ABOUT. As técnicas de relevância aplicadas para ordenar as respostas foram respectivamente: A formula de Salton e Buckley (1988); a função DEFINESCORE utilizando o parâmetro OCCURRENCE (no qual a pontuação foi baseada no número de ocorrências da palavra utilizada como argumento para a pesquisa); a função ABOUT, que apesar do Oracle não implementar nativamente um banco de dados de conceitos para o Português-Brasil, apresentou resposta similar à função DEFINESCORE. Em todas as opções, utilizou-se a função SCORE, a qual produz o critério de relevância a ser ordenado pela cláusula *group by* do comando SQL.

Figura 3 – Consultas SQL utilizando as funções do Oracle Database



Fonte: Elaborada pelos autores.

A Figura 4, demonstra as consultas SQL no PostgreSQL utilizando a função TSRANK com todas as opções de classificação disponíveis pelo SGBD. Portanto, foram realizadas sete execuções da sentença SQL, sendo uma para cada argumento válido.

Figura 4 – Consulta SQL utilizando a função TSRANK - PostgreSQL

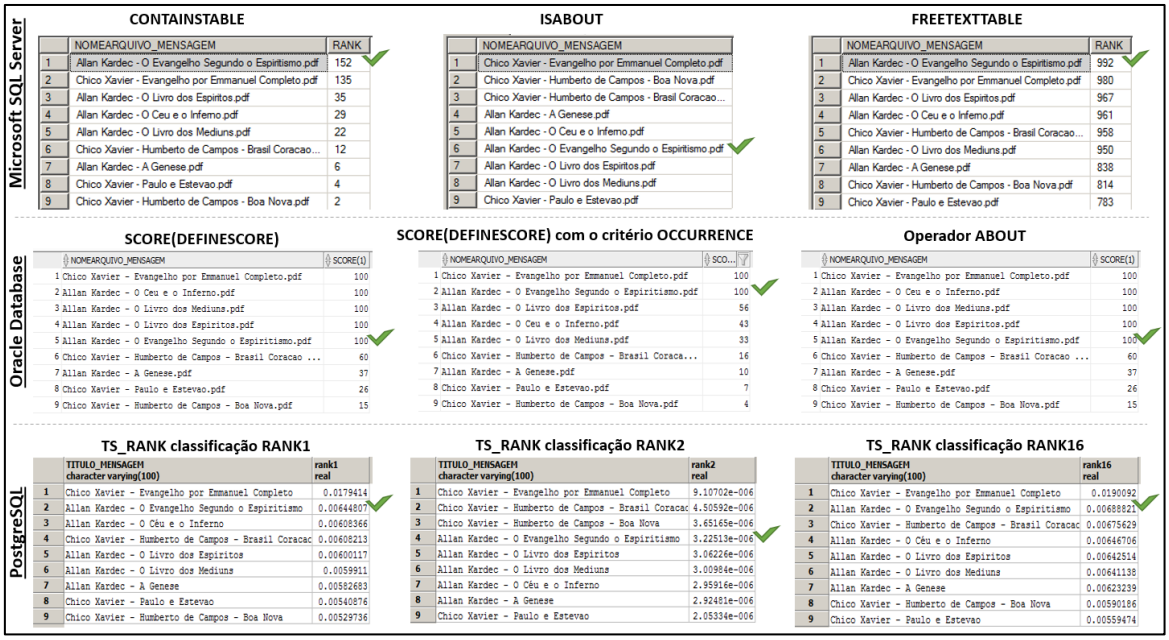
```
SELECT "TITULO_MENSAGEM",
ts_rank("TEXTO_MENSAGEM1", query,0) + ts_rank("TEXTO_MENSAGEM2", query,0) + ts_rank("TEXTO_MENSAGEM3", query,0) AS rank0
FROM "MENSAGEM",
to_tsquery('caridade') query
WHERE query @@ "TEXTO_MENSAGEM1"
ORDER BY rank0 desc;
```

Fonte: Elaborada pelos autores

Destaca-se que o resultado das consultas realizadas, tanto na ordem dos documentos, quanto na pontuação dada pelo sistema, foram distintos para cada parâmetro utilizado. A Figura 5, apresenta o resultado da busca, em cada caso. Apareçe ainda, nessa mesma figura, através "*check*" nas linhas, mostrando um document específico em diferentes poisições na resposta

dependo da técnica de relevância utilizada. O document analisado é entitulado: “O evangelho segundo o espititismo”.

Figura 5 – Resultados das consultas SQL utilizando as funcionalidades de cálculo de relevância por SGBD



Fonte: Elaborada pelos autores

Dessa forma, observa-se que apesar das consultas retornarem sempre os mesmos resultados, ou seja, os mesmos documentos, existem diferenças na ordenação desses documentos e na pontuação retornada. Isso ocorre pois, cada função de relevância, adota uma técnica de cálculo distinta. Esse resultado era esperado, tendo em vista que as técnicas aplicadas não são determinísticas. Cada uma delas tenta à sua maneira encontrar o *best matching* entre os parâmetros de busca e os documentos pesquisados.

Por fim, cabe ressaltar que, em relação ao desempenho na busca, não foi possível comparar os resultados entre os SGDB. Pois, a base de dados implementada para atender ao escopo deste estudo, é pequena para produzir números comparáveis. Afinal, todas as consultas retornaram tempos de resposta inferiores à um segundo.

8. Conclusões

A indexação de texto completa implementada nos SGBD é uma técnica de RI, que a partir palavras-chave visa localizar os documentos que melhor se enquadram com os descritores da consulta.

Como apresentado, a técnica baseia-se na criação de índices textuais, considerando as regras de um idioma em específico, no qual todas as palavras se tornam pontos passíveis na recuperação de documentos, aumentando significativamente a quantidade de documentos relevantes nos processos de buscas realizadas pelo usuário.

Considerando que o objetivo proposto deste trabalho, foi comparar o desempenho de indexação dos documentos e as funcionalidades disponibilizadas para realizar as busca por palavras-chave em uma coleção de documentos persistidos em cada um dos três SGBD, conclui-se que tais funcionalidades são implementadas em cada SGBD de maneira particular, tendo cada qual o próprio método para indexar, classificar e retornar as respostas ordenadas de acordo com os critérios de relevância estabelecidos por cada uma das técnicas.

A partir dos resultados obtidos no desempenho da indexação dos dados, verificou-se que o SGBD PostgreSQL, é o que teve o resultado superior com relação ao tempo, mesmo ao gerar um léxico específico para cada documento, o PostgreSQL indexa uma palavra mais de uma vez. Isso pode resultar em um maior consumo de espaço de armazenamento para uma mesma coleção de documentos. Adicionalmente, no caso do PostgreSQl, pode-se definir a criação de um índice para os atributos do tipo Ts_vector, o qual irá gerar um léxico distinto de termos, os quais apontam para os documentos referenciados. Isso irá melhorar o desempenho de busca para grandes volumes de dados. Entretanto, não foi avaliado empiricamente qual seria essa melhoria no desempenho com a criação do índice geral do léxico, tendo em vista que a criação desse tipo de índice trará maiores benefícios para indexar grandes quantidades de documentos, afinal para cada documento o Postgres já implementada um índice individual. Portanto, essa abordagem pode ser testada em trabalhos futuros.

Ainda sobre o desempenho de indexação dos dados, o Oracle Database foi muito superior quando comparado ao Microsoft SQL Server, pois o Microsoft SQL Server necessita de executar um componente de software externo para ler os documentos em pdf e convertê-los para texto, antes de poder indexar os documentos. Por outro lado, a quantidade de palavras indexadas pelo Microsoft SQL Server foi maior que a quantidade de palavras indexadas pelo

Oracle Database. No caso epecífico do PostgreSQL, a quantidade de palavras indexadas, foi superior a ambos. Essas diferenças ocorrem devido ao emprego de diferentes técnicas de normalização de dados empregada caso a caso, tais como uso de *stemmer*, lista de *stop words* etc.

Ao analisar os resultados encontrados na funcionalidade de RI, como o cálculo de relevância implementado em cada SGBD possui implementação distinta, verificou-se que cada SGBD apresenta uma ordenação dos resultados da consulta que pode ser distinta dentre os diferentes SGBD avaliados.

Por fim, conclui-se que é inviável indicar qual é o melhor SGBD para indexar e recuperar documentos textuais, dentre os avaliados. Desse modo, a escolha do SGBD a ser utilizado para desempenhar a função de indexação texto completo dependerá de alguns fatores, tais como: tipo de documento a ser indexado, funcionalidades utilizadas e os recursos financeiros disponíveis para adquirir licenças de software (SGBD). Observa-se também que, o suporte à indexação de texto completa implementada nos SGBD testados, já está madura o suficiente para ser aplicada em sistemas de informação tradicionais que poderão se beneficiar pela implementação de funcionalidades de consultas *ad hoc* em textos contidos nos SGBD associadas à consulta por metadados estruturados com um tempo de resposta aceitável.

Após a apresentação dos resultados obtidos, apresentam-se aqui algumas sugestões de trabalhos futuros que poderão ser desenvolvidos a partir deste estudo realizado. A partir dos resultados obtidos, constatou-se que a mensuração do desempenho do processo de indexação para a RI pode ser influenciada por uma série de fatores, tendo consequentemente os resultados impactados por esses fatores. Ademais, além de expandir o número de documentos do *corpus* a serem indexados visando avaliar diferentes tamanhos de bases de dados a fim de avaliar a escalabilidade linear ou não nos tempos de resposta, outros desdobramentos poderiam ser avaliados, como analisar dentre as diferentes técnicas implementadas nos diferentes SGBD, quais serão passíveis de retornarem resultados mais precisos em domínios específicos.

Referências

ALVARENGA, L. Representação do conhecimento na perspectiva da Ciência da Informação em tempo e espaço digital. **Encontros Bibli: revista eletrônica de biblioteconomia e ciência da informação**, Florianópolis, v. 8, n. 15, 2013.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern information retrieval**. New York: ACM Press, 1999. 511p.

BEALL, J. **The journal of academic librarianship**, [*s.l.*], v. 34, n. 5, p. 438–444, 2008.

BORGES, G. S. B. **Indexação automática de documentos textuais**: proposta de critérios essenciais. 2009. Dissertação (Mestrado) – Escola de Ciência da Informação, UFMG, Belo Horizonte, 2009.

CLARKE, C. L. A; CORMACK, G. V.; TUDHOPE, E. A. Relevance ranking for one to three term queries. **Information Processing and Management: an International Journal,** [*s.l.*], v. 36, n. 2, p. 291-311, 2000.

GANTZ, J. F. et al. **A forecast of worldwide information growth through 2010**. Framingham: IDC, 2007. Disponível em: <http://www.tobb.org.tr/BilgiHizmetleri/Documents/Raporlar/Expanding_Digital_Universe_IDC_WhitePaper_022507.pdf>. Acesso em: 7 jul. 2017.

GANTZ, J. F.; REINSEL, D. **The digital universe in 2020**: big data, bigger digital shadow s, and biggest grow th in the far east. Framingham: IDC, 2012. Disponível em: <https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf>. Acesso em: 7 jul. 2017.

LANCASTER, F. W.; WARNER, A. J. Information retrieval today. **Information Resource**, *[s.l.]*, 1993. 341p.

MANNING, C. D.; SCTÜTZE, H. **Foundations of statistical natural language processing**. MIT, 2003. 680 p.

MICROSOFT. TechNet. **Pesquisa de texto completo (SQL Server)**. 31 mar. 2012. Disponível em: <https://technet.microsoft.com/pt-br/library/ms142571(v=sql.105).aspx>. Acesso em: 14 jul. 2016.

ORACLE TEXT REFERENCE. Oct. 2015. Disponível em: <https://docs.oracle.com/cd/E11882_01/text.112/e24436.pdf>. Acesso em: 10 de ago. 2017.

ORACLE TEXT REFERENCE. **Oracle Text CONTAINS Query Operators**. Nov. 2011. Disponível em: <https://docs.oracle.com/cd/B28359_01/text.111/b28304/cqoper.htm#CCREF0301>. Acesso em: 22 de ago. 2017.

POSTGRESQL 9.3.18 DOCUMENTATION: controlling text search. Ago. 2017. Disponível em: <https://www.postgresql.org/docs/9.3/static/textsearch-controls.html>. Acesso em: 10 de set. 2017.

ROBERTSON, S. E.; SPARCK J. K. Relevance weighting of search terms, **Journal of the American Society for Information Science**, v. 27, p. 129–146, 1976.

ROBREDO, J. A indexação automática de textos: o presente já entrou no futuro**. Estudos Avançados em Ciência da Informação,** Brasília, v. 1, p. 235-274, 1982.

RUSSELL, S. J.; NORVIG, P. **Inteligência artificial.** Rio de Janeiro: Elsevier, 2004. 1021p.

SALTON, G.; BUCKLEY, C. Term-weighting approaches in automatic text retrieval. **Information Processing and Management**, v. 24, n. 5, p. 513-523, 1988.

SARACEVIC, T. Ciência da informação: origem, evolução e relações. **Perspectivas em Ciência da Informação,** Belo Horizonte, v. 1, n. 1, p. 41-62, jan./jun. 1996.

SILVA, E. M. **Recuperação da informação através de busca comparada em domínio específico, baseado em expressões multipalavras.** 2013. Tese (Doutorado) – Escola de Ciência da Informação, UFMG, Belo Horizonte, 2013.

Artigo submetido em: 28 nov. 2018
Artigo aceito em: 22 abr. 2019