**Comparação de busca com grande quantidade de dados utilizando Árvores Rubro Negras**

Claudio da Silva T. Júnior Pietro N. Neiva

Instituto de Educação Superior de Brasília - IESB

Campus Sul SGAS Quadra 613/614 - Lotes 97 e 98 L2 Sul - Brasília - DF, 70.200-730, Brazil

[claudiojunior9662@gmail.com](mailto:claudiojunior9662@gmail.com), [nazarpietro@gmail.com](mailto:nazarpietro@gmail.com)

March 2020

***Abstract.*** The search for data is indispensable in any application. We obtained a spreadsheet with data on medicines used in national territory from the government database and, with this data, we will look for solutions with the Red Black Tree to optimize the consultation, insertion and removal of records.

***Key-words***: RBT (Red Black Tree).

***Resumo***. A busca de dados é indispensável em qualquer aplicativo. Nós obtemos uma planilha com dados sobre medicamentos utilizados em território nacional da base de dados do governo e, com esses dados, procuraremos soluções com a Red Black Tree para otimizar a consulta, inserção e remoção de registros.

***Palavras chave***: RBT (Árvore Rubro Negra, ***R****ed* ***B****lack* ***T****ree*).

**1 - Introdução**

Os dados disponibilizados no portal do governo brasileiro (dados.gov.br) para acesso ao público são disponibilizados em planilhas, uma forma ineﬁciente para lidar com tamanha quantidade de informações. De posse de um dataset de preços de medicamentos, iremos tentar otimizar esse tratamento de informações com a ajuda da RBT e uma Structured Query Language, intermediados por uma aplicação em JAVA com um CRUD básico.

**2 - Objetivos**

**2.1 - Objetivo geral**

Implementação de um sistema de consulta e registro de medicamentos utilizando uma RBT para aperfeiçoar a velocidade de interação com o usuário na Busca, inserção e manipulação dos dados sobre medicamentos.

**2.2 - Objetivo específico**

Consultar a base de dados do governo para entender a periodicidade que estes são atualizados por meio de planilhas, converter esses dados para uma query language, implementar uma RBT utilizando a linguagem JAVA, implementar um CRUD básico para interação com a RBT com o usuário. Com isso queremos facilitar a manipulação de dados na consulta de medicamentos, que atualmente tem que fazer a consulta pelo Excel, de maneira pouco intuitiva e sem manipulação de grupo de dados. Queremos otimizar também a inserção de dados e atualização da tabela a medida que saírem novos dados pelo ministério da Saúde, demonstrando ser mais rápido e intuitivo pela manipulação com RBT.

**3 - Referencial Teórico**

**3.1 - Dev C++ IDE**

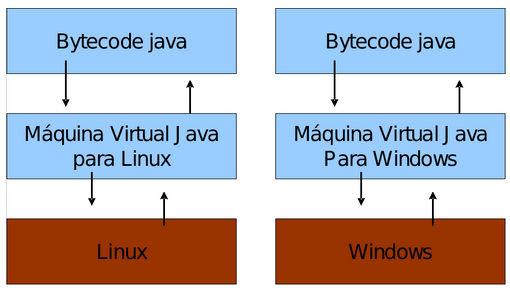
Dev-C++ é um *IDE* (ambiente de desenvolvimento integrado) completo para a linguagem de programação C/C++. Ele usa a porta Mingw do GCC (*GNU compiler collection*) como compilador. O Dev C++ também pode ser usado em combinação com o *Cygwin* ou qualquer outro compilador baseado em GCC [1].

**3.2 - C++**

C++ linguagem de programação compilada multiparadigma (seu suporte inclui linguagem imperativa, orientada a objetos e genérica) e de uso geral. Desde os anos 1990 é uma das linguagens comerciais mais populares, sendo bastante usada também na academia por seu grande desempenho e base de utilizadores [2].

**3.3 - Java**

Java foi criado pela antiga Sun Microsystems e mantida através de um comitê (http://www.jcp.org) . Utiliza o conceito de máquina virtual, onde existe entre o sistema operacional e a aplicação, uma camada extra responsável por “traduzir” o que a aplicação deseja fazer para as respectivas chamadas do sistema operacional onde está rodando no momento. [3].



[1] - Dev-C++ 5 (currently beta). BloodshedSoftware. Disponível em: <https://www.bloodshed.net/devcpp.html>. Acesso em: 18/04/2020.

[2] - C++. Wikiland. Disponível em: <https://www.wikiwand.com/pt/C%2B%2B>. Acesso em: 18/04/2020.

[3] - Capítulo 2 - O que é java. Caelum. Disponível em: <https://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java/>. Acesso em: 18/04/2020.

**3.4 - Eclipse IDE**

- Eclipse IDE é um ambiente de desenvolvimento, desenvolvido em Java pela IBM, seguindo o modelo open source de desenvolvimento de software [4].

**3.5 - MariaDB**

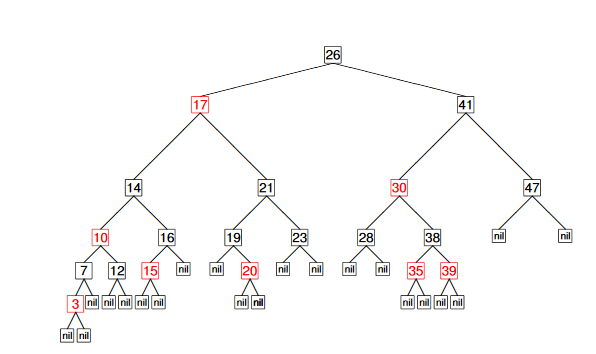
MariaDB, SGDB que surgiu como fork do MySQL, criado pelo próprio fundador após sua aquisição pela Oracle [5].

**3.6 - *RBT* (Árvore Rubro Negra - *R****ed* ***B****lack* ***T****ree***)**

**3.6.1 - Árvore Rubro-Negra**

Árvore Rubro-Negra é uma árvore binária de busca em que cada nó é constituído dos seguintes campos:

* cor (1 bit): pode ser vermelho ou preto;
* *key* (e.g. inteiro): indica o valor de uma chave;
* *left*, *right*: ponteiros que apontam para a subárvore esquerda e direita, respectivamente;
* pai: ponteiro que aponta para o nó pai. O campo pai do nó raiz aponta para *null*. O ponteiro pai facilita a operação da árvore rubro-negra, conforme visto a seguir.



**3.6.2 - Propriedades**

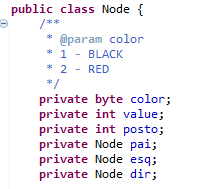
Além das propriedades da árvore binária de busca, a *RBT* possui algumas propriedades adicionais:

* Todo nó da árvore ou é vermelho ou é preto;
* A raiz é preta;
* Toda folha (null) é preta;

[4] - Eclipse (Software). Wikipédia. Disponível em: <https://pt.wikipedia.org/wiki/Eclipse_(software)>. Acesso em: 18/04/2020.

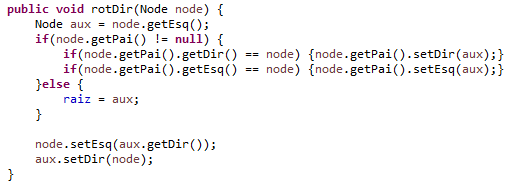
[5] - MariaDB. Wikipédia. Disponível em: <https://pt.wikipedia.org/wiki/MariaDB>. Acesso em: 18/04/2020.

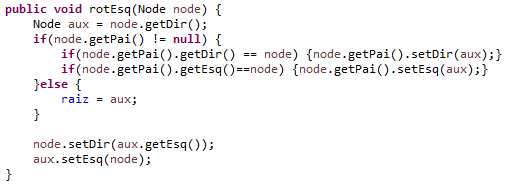
* Se um nó é vermelho, então ambos os filhos são pretos;
* Para todo nó, todos os caminhos do nó até as folhas descendentes contêm o mesmo número de nós pretos.



A altura *h* de uma árvore rubro-negra de *n* chaves ou nós internos é no máximo 2 log(n + 1).

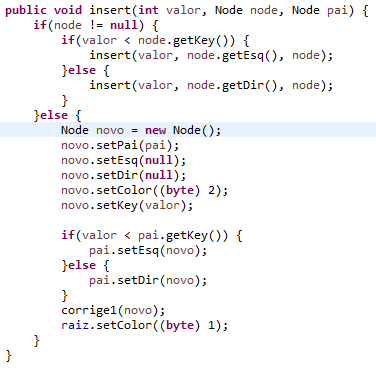
**3.6.3 - Rotações**

****

****

Os métodos de rotação (*rotDir* e *rotEsq,* rotação para direita e rotação para esquerda, resp) mudam alguns ponteiros da árvore, preservando a propriedade de árvore binária de busca, levando tempo constante de O(1).

**3.6.4 - Inserções**



O algoritmo de inserção não viola as propriedades de árvore rubro-negra.

* Todo nó da árvore ou é vermelho ou é preto;
* A raiz é preta;
* Toda folha (*null*) é preta;
* Se um nó é vermelho, então ambos os filhos são pretos;
* Para todo nó, todos os caminhos do nó até as folhas descendentes contêm o mesmo número de nós pretos.

O algoritmo de inserção substitui um nó sentinela (preto) por um novo nó interno vermelho *z* contendo o valor novo inserido. Este nó aponta, por sua vez, a dois nós sentinela (preto), à esquerda e à direita. [6]

**4 - Desenvolvimento**

**4.1 - Adaptação do *dataset***

De posse da base de dados, disponível em [7], um arquivo no formato *.csv*, com 25.681 registros e 40 colunas, com tamanho ocupado em disco de 9.40 MB, foi convertido em *.sql* utilizando a ferramenta disponível em [8], resultando em espaço em disco de 10.5 MB. Os dados foram replicados, aplicando o final sequencial “`\_X” para cada coluna “SUBSTANCIA”, resultando em um total de registros de 1.101.441 e um espaço em disco de aproximadamente 400 MB

[6] - Árvore Rubro-Negra. USP. Disponível em: <https://www.ime.usp.br/~song/mac5710/slides/08rb.pdf>. Acesso em: 18/04/2020. Adaptado.

[7] - Preços de medicamentos - consumidor. Dados.gov.br. Disponível em: <http://www.dados.gov.br/dataset/anvisa-precos-de-medicamentos-consumidor>. Acesso em: 18/04/2020.

[8] - Convert CSV to SQL. Disponível em: <https://www.convertcsv.com/csv-to-sql.html>. Acesso em: 18/04/2020.

**4.2 - Implementação da RBT**

Utilizando a linguagem C++ com a teoria de árvore binária Rubro-negra, Implementando no Dev C++ e utilizando o referencial teórico, conseguimos implementar o balanceamento da árvore à medida que são feitas novas inserções, entretanto o balanceamento a medida que são excluídos os elementos da árvore tem encontrado diﬁculdade, especialmente se o balanceamento for feito com a raiz envolvida. Após a implementação em C++, foi decidido a implementação em Java, tendo em vista também a facilidade para a criação de um CRUD básico com interface gráﬁca.

**5 - Conclusão**

**6 - Proposta futura**