Relatório Técnico da Matéria de Estrutura de Dados

Leandro Botelho Alves de Miranda
leandrobotelhoalves@gmail.com

1 Descrição do Trabalho

A ferramenta desenvolvida como *trabalho na matéria de estrutura de dados* é uma aplicação que faz registros de dados de uma equipe de vendas de uma empresa, onde o usuário pode fazer consultas e manipulações de inserção, busca e remoção de vendas em um registro de vendas.

Cada venda é representada por uma classe *Venda* e composta por 4 atributos: *filial*, *ano_ mês*, *cod_ vendedor*, *total_ vendido*. Os atributos *filial* e *total_ vendido* são declarados como *strings*, porém são manipulados como valores inteiros. O atributo *cod_ vendedor* é do tipo *string*. O atributos *ano_ mês* é declarado como *string*, onde é composto por (1 valor inteiro representando o mês + "/"+ 1 valor inteiro representando o ano) [mês/ano].

O atributo filial têm valores que variam entre 1 e 99, enquanto o atributo *total_ vendido* varia entre 1 e 80.

O atributo *cod_ vendedor* é um atributo string representado por [COD + número identificador], onde número identificador varia entre 1 e 10.000. Ex: COD8907.

Diferente como declarado no título do trabalho, a estrutura do atributo *ano_ mês* é apresentada de forma inversa, onde a string é composta pelo mês inicialmente e depois o ano [mês/ano], onde o ano varia entre 1900 e 2050. Ex: 11/1955.

Cada objeto venda apresenta a seguinte estrutura em seu método de impressão:

```
filial,ano_mes,cod_vendedor,total_vendido
ex:86,1/2033,COD9193,27
```

Todas as operações de manipulação de dados acontecem (inserção e remoção) sobre estes atributos. Em relação as estruturas de dados, alguns atributos apresentam manipulação, que são descritos na próxima seção.

2 Detalhamento das estruturas de dados utilizadas neste trabalho

Em relação as estruturas de dados utilizadas nesta aplicação, cada estrutura é associada a um tipo de atributo, onde em seus campos ou nós são armazenados seus respectivos valores para cada objeto da base de dados.

As estruturas de dados presentes neste trabalho são:

- **árvore binária de busca**;
- **Lista dinâmica encadeada simples**.
- **Tabela de dispersão com encadeamento externo, com chaves duplas**;

Após o carregamento dos objetos(vendas) em memória, o programa cuida em gerar cada estrutura de dados tratando cada tipo de atributo. Maiores detalhes de como são manipulados estes atributos nas estruturas de dados são descritas nas próximas subseções. Ao final, com todas as estruturas de dados prontas, a aplicação fica disponível para o usuário efetuar suas consultas baseadas no enunciado no trabalho da matéria.

Outro fato importante, é que como podem existir atributos com valores repetidos entre vendas, como por exemplo, duas vendas feitas pela mesma filial. Diante disso, vai existir casos de valores iguais para um mesmo atributo, que são tratados nesta aplicação dependendo do tipo de atributo, pois no caso do atributo *cod_ vendedor* não são considerados valores repetidos.

2.1 Árvore binária de busca – ABB

A estrutura do nó desta árvore apresenta as seguintes informações, onde temos as informações dos nós a esquerda, direita e pai da árvore, e o conteúdo que pertence a classe Venda.

```
class No {
private Venda conteudo;
private No pai;
private No esquerda;
private No direita;
}
```

Porém o comportamento da árvore binária de busca é baseado na manipulação dos atributos pertencentes a classe Venda. Com isso, foi escolhido

dois atributos para a manipulação destes dados: ***filial*** e ***cod_ vendedor***. Além disso, foi gerados duas árvores binárias de busca, onde seu conteúdo é baseado nos valores destes dois atributos . A manipulação dos dois atributos respeita o critério de ordem adotados pelas políticas da árvore binária de busca, porém o atributo *filial* é do tipo inteiro e o atributo *cod_ vendedor* pertence ao tipo *string*.

Como enunciado anteriormente, o atributo *cod_ vendedor* apresenta valores únicos em relação as todas os objetos da base de dados, ou seja, a estrutura da ABB vai ser de formato convencional, evitando conteúdos duplicados.

O critério de ordem para valores inteiros, é baseado na ordem dos números de forma crescente, e em relação a Strings é baseado no critério adotado pelo método *compareTo()* do Java.

Porém, em relação a AAB com manipulação do atributo *filial*, vai existir um caso especial na AAB. Esse caso especial envolve a manipulação de nós que contém mais de uma venda como conteúdo do nó. Neste caso, vendas que apresentam filiais iguais, vai existir mais uma estrutura de dados onde é armazenado mais de uma venda, que no caso, é uma lista encadeada simples. O nó para esse tipo de árvore apresenta a seguinte estrutura.

```
class No {


Venda conteudo;
ListaDinamicaDeVenda vendas;
No pai = null;
No esquerda = null;
No direita = null;
}
```

Nesta estrutura, a classe *ListaDinamicaDeVenda* é uma classe que representa a estrutura de dados Lista dinâmica encadeada simples. A função desta lista é armazenar todas as vendas que apresentem o mesmo valor para o atributo filial.

A escolha do uso da ABB deve-se pois ela se torna ótima na alocação de memória, são fáceis de manipular, custo de operações são simples, além de sua complexidade ser $O(\log n)$, onde n é o número de objetos.

2.2 Tabela de dispersão (hash) com encadeamento externo, com chaves duplas

A tabela de dispersão utilizada neste trabalho é uma tabela hash que para o tratamento de colisões utiliza encadeamento exterior.

Ela é utilizada para a manipulação do atributo *ano_mes* presente na classe Venda. A grande diferença dessa tabela *hash* com as tabelas *hash* tradicionais é que esta utiliza uma chave dupla para o acesso ao valor da tabela. O valor das chaves são representados pelo acesso ao mês e o ano da venda. A sua função de dispersão 1 retorna para o vetor A de dimensão 2, dois valores representando o acesso para o mês (x) e o ano (y).

$$A[0] := f(x,y) = (x\%12) A[1] := f(x,y) = (y\%1900) \quad (1)$$

O resultado desta função, indica onde a respectiva venda é armazenada na tabela de mês/ano.

Para o caso de colisões, o tratamento destas é feito o uso de listas encadeadas, uma para cada possível endereço base mês/ano. A representação desta tabela é melhor vista na Figura 1.

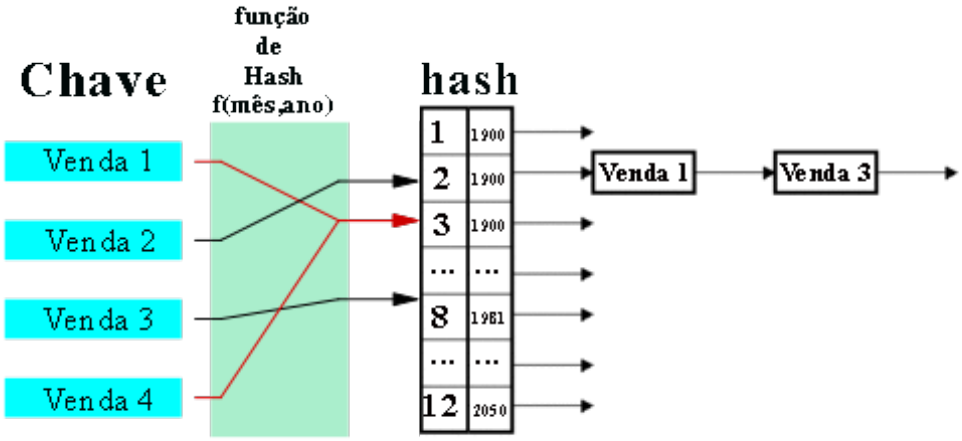

Figura 1: Representação gráfica da Tabela Hash, com chaves duplas representando mês/ano.

A escolha desse método é que a consulta é de $O(1)$ para as chaves e para outras informações referente a uma respectiva venda i, a complexidade no pior caso é percorrer a lista toda em uma respectiva chave (x,y).

2.3 Lista dinâmica encadeada simples

O uso da lista dinâmica encadeada simples é útil em atributos que apresentem o mesmo valor em objetos(vendas) distintas. O seu objetivo é armazenar vendas que contém o mesmo de valor atributo. Exemplo: Uso da lista

dinâmica nos nós da ABB para o atributo filial e tabela de dispersão para o atributo mês/ano.

3 Informações técnicas de uso da aplicação

A aplicação deste programa resolve de forma dinâmica as respectivas questões dadas no enunciado no trabalho: as alternativas a,b e c. O calculo do total de compras é feito a partir dos valores do número de filiais e do calendário.

A interface de apresentação do programa é visto na Figura 2.

Figura 2: Imagem da interface principal.

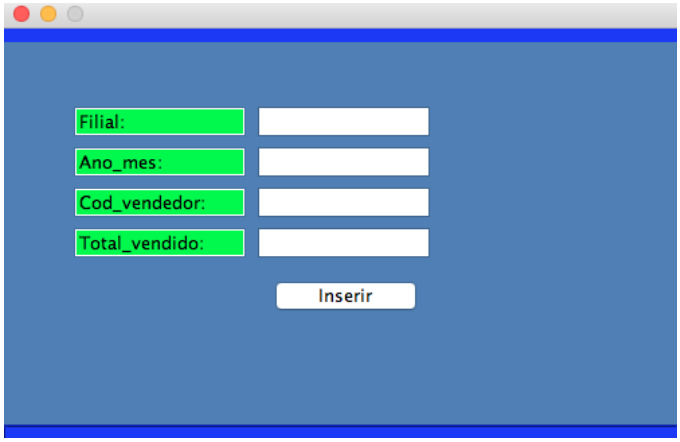

Figura 3: Imagem da interface de inserção.

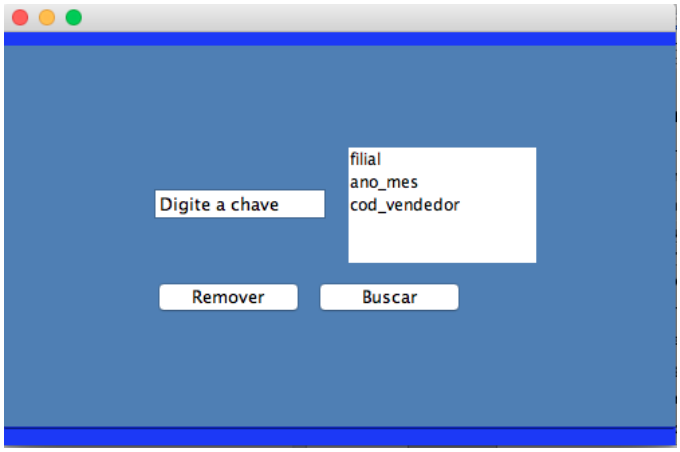

Figura 4: Imagem da interface de remoção.

3.1 Funcionalidades e guia de uso

Para a execução do programa, o usuário deve seguir alguns de passos:

1. Digitar um número n objetos que pretende inicializar e apertar o botão **Gerar Dataset**.

2. Após o *dataset* gerado, o usuário pode escolher duas alternativas: a primeira é selecionar ativar quais campos que pretende manipular, mas ANTES, o usuário DEVE PREENCHER OS CAMPOS QUE PRETENDE FAZER OS CÁLCULOS e DEPOIS escolher quais campos deve habilitar, apertando o botão **Ativar Campo**. Ao apertar, o botão ficará **ativado** ou **desativado**, permitindo que esses atributos façam parte do cálculo do total de vendas, no caso tanto o atributo código de vendedor ou atributo mês/ano. Caso contrário, acontece erros no cálculo da soma total de compras.

3. Ao selecionar estas opções, o passo final é apertar o botão **Calcular total de vendas**, onde no final vai sair o resultado com o cálculo do total de vendas.

4. Ou usuário pode inserir mais dados na base ao apertar o botão **Inserir Dados**. Ao selecionar essa opção, o usuário preenche todos os atributos de uma venda e gera uma venda nova. A nova janela é vista na Figura 3. PS: NÃO FECHE A NOVA JANELA, senão as operações de inserção não serão visíveis nas estruturas de dados.

5. O usuário também pode remover ou buscar dados em seu *dataset* apertando o botão **Rem/Bus Dados**. Ao selecionar esta opção, o usuário

escolhe, ou remover ou buscar uma venda na base de dados, escolhendo o respectivo atributo e digitando a chave. Ex: se selecionar ano_mes, o usuário pode digitar "12/2010" e escolher se o usuário quer fazer uma consulta na base de dados ou remover uma venda com esta respectiva chave. A nova janela é vista na Figura 4. PS: NÃO FECHE A NOVA JANELA, senão as operações de remoção não serão visíveis nas estruturas de dados.

Quando for manipular com consultas ao total de vendas, sempre preencha os campos primeiros e depois selecione o botão **Ativar campo**/ **ativado**/ **desativado** .

Em relação a execução do programa, o primeiro passo vai gerar n vendas de forma aleatória, variando o número de filiais, o número de datas, o número do código de funcionários e o número total de vendas do funcionário, juntamente com todas as estruturas de dados referentes a cada atributo, como.

Os resultados das consultas estão sendo armazenadas em listas encadeadas para cada tipo de consulta e após é feita a comparação entre os objetos de cada lista encadeada e computando o total de vendas final.

O botão **Ver Dataset** gera uma janela mostrando ao usuário todas as vendas ativas no programa. É recomendável para *datasets* pequenos para uma melhor visualização. Caso queira usar para *datasets* maiores, os resultados estão impressos no console do programa. Este botão também é util, quando o usuário quer saber como está o *dataset* após as operações de inserção e remoção, além da visualização da árvore dos atributos.