Universidade Federal Fluminense
Programa de Pós-graduação em Ciência da Computação
TIC10002 – Estruturas de Dados e Algoritmos (2017.1)
T1 – 12/06/2017

Professor: Luiz André Portes Paes Leme
Aluno: Paulo Cezar Lacerda Neto

Trabalho de Implementação

Enunciado do Trabalho

Os dados de venda da equipe de vendas de uma empresa estão armazenados em um arquivo txt que obedece ao seguinte formato: filial, ano_mês, cod_vendedor, total_vendido.

Faça um programa que carregue os dados do arquivo em estruturas de dados e que permita responder expressões como:

1) total de vendas das filiais com códigos entre 10 e 20
2) total de vendas das filiais com códigos entre 10 e 20 nos meses de Jan/17 até Jun/17
3) total de vendas de todas as filiais nos meses de Ago/17 até Out/17

As estruturas de dados devem permitir acesso eficiente para todos os tipos de expressões e a conjunção de condições sobre filiais e datas devem ser realizadas por meio de operação de interseção de conjuntos. Cada resumo de venda (linha) do arquivo deve dar origem a somente uma instância (objeto) no programa. Utilize as estruturas de dados mais eficientes para cada tipo de pergunta.

Abordagem de implementação

A abordagem utilizada para solucionar o problema acima foi o uso de Árvores AVL para contabilizar o total de vendas em um intervalo simples (expressões 1 e 3) e a técnica chamada Hash Join[1] para calcular o total de vendas considerando dois intervalos (expressão 2).

O funcionamento do programa ocorre da seguinte forma, inicialmente os dados do arquivo são lidos e para cada registro é criado um objeto do tipo Venda, após ser criado, o mesmo objeto é inserido em duas árvores AVL, a primeira utiliza como chave de ordenação o número da filial e a segunda utiliza como chave o ano_mês.

O conteúdo de cada nó das árvores é uma lista encadeada, e cada nó destas listas contém o objeto Venda, pois uma mesma filial pode apresentar mais de uma venda assim como é possível ocorrer mais de uma venda no mesmo ano_mês.

Com as árvores AVL construídas, é possível utilizá-las para responder as expressões 1 e 3 do enunciado do problema, basta navegar pela árvore começando pela raiz e percorrer os filhos que tem chave dentro do intervalo desejado, totalizando o montante das vendas.

Para responder a segunda expressão do enunciado, foi utilizada a técnica chamada Hash Join, que chega ao resultado desejado em duas etapas. Na primeira etapa selecionamos uma das árvores criadas anteriormente, por exemplo a árvore com chave Filial, e percorremos a árvore navegando pelo intervalo desejado, por exemplo, filiais com código entre 10 e 20. Cada venda dentro do intervalo percorrido é inserida em uma tabela hash a ser usada na segunda etapa do procedimento, esta tabela hash é implementada como uma lista estática em que cada elemento contém uma árvore AVL com as vendas que apresentam a mesma chave hash e, portanto, estão armazenadas no mesmo *bucket* da tabela.

Na segunda etapa do Hash Join selecionamos a outra árvore, neste exemplo, a árvore com chave ano_mês, e percorremos o intervalo correspondente a esta árvore, por exemplo vendas de Jan/17 até Jun/17 e ao percorrer os caminhos da árvore totalizamos as vendas para chegar ao resultado esperado, porém ao fazer isso só incluímos no somatório as vendas que também se encontram na tabela hash construída na primeira etapa do Hash Join, com isso garantimos que o resultado obtido será baseado apenas nas vendas que existem tanto no intervalo de filiais quanto no intervalo de datas.

[1] https://rosettacode.org/wiki/Hash_join

Análise da Solução

Vamos dividir a análise da solução implementada em duas partes, na primeira parte trataremos do caso em que ser chegar ao total de vendas com base em um único filtro, por exemplo filial ou intervalo de data, ou seja, são N vendas e cada venda está associada a uma de M filiais ou datas, temos as seguintes operações principais:

1) Inserção das N vendas na árvore AVL
2) Totalização das vendas dentro do intervalo desejado

Na operação de inserção a complexidade de tempo é de **O(N * Log N)** considerando que a altura máxima da árvore é Log N (caso cada venda tenha uma filial ou mês_ano diferente das outras, ou seja N=M), o número de vendas é N e o custo de inserir cada venda na respectiva lista associada a um dos nós da árvore é constante. Na operação de totalização, no pior caso, pode ser necessário percorrer todos os nós, o que corresponde a **O(N)**.

Para o caso em que se deseja calcular o total de vendas considerando dois intervalos, temos as seguintes operações principais:

1) Inserção das N vendas nas duas árvores AVL.
2) Criação da tabela hash a partir da primeira árvore.
3) Totalização das vendas dentro dos intervalos desejados a partir da segunda árvore.

A primeira operação tem complexidade **O(N * Log N)** como descrito anteriormente, já que a criação de duas árvores ao invés de uma não representa diferença significativa para a notação big O em termos de tempo. A criação da tabela hash apresenta complexidade **O(N * Log N)**, pois estou utilizando uma árvore AVL para armazenar os elementos que apresentam a mesma chave hash e no pior caso, embora improvável, todas as vendas serão inseridas no mesmo *bucket*. A totalização baseada no Hash Join apresenta complexidade **O(N * Log N)**, dado que todas as N vendas podem estar na interseção dos intervalos e também a tabela hash pode apresentar um alto grau de colisões, de modo que no pior caso todos os itens podem estar armazenados no mesmo *bucket.*

Organização do código

No diretório **program** podem ser encontrados o script **buildandrun.sh** e **buildandrun.bat** que são os scripts utilizados para executar o programa, além disso também podem ser encontrados o diretório **src** em que estão os códigos das classes utilizadas para implementar o programa e o diretório **input** em que está o arquivo de entrada **file1.txt** utilizado para o teste, ainda no diretório program pode ser visto o diretório **bin** em que estão os binários.

Como rodar o programa

Para rodar o programa basta executar um dos scripts **buildandrun.sh** ou **buildandrun.bat** dependendo da plataforma do seu sistema operacional.

Estes scripts são uma forma conveniente de executar o programa, mas caso você deseje executar diretamente usando o executável **java**, basta executá-lo como indicado abaixo e substituindo os parâmetros de linha de comando conforme necessário:

java -cp ./bin **Main** ARQ_ENTRADA FILIAL_INICIO FILIAL_FIM DATA_INICIO DATA_FIM

Parâmetros:
ARQ_ENTRADA: caminho do arquivo de entrada
FILIAL_INICIO: código de filial inicial para consulta
FILIAL_FIM: código de filial final para consulta
DATA_INICIO: data inicial da consulta
DATA_FIM: data final da consulta

Exemplo:
java -cp ./bin Main ./input/file1.txt 10 20 201701 201702

O conteúdo do arquivo de entrada tem o seguinte formato:
FILIAL,ANO_MES,COD_VEN,TOTAL_VEN

Exemplo:
10,201701,01,100.00
20,201701,02,100.00
10,201702,01,100.00