Universidade Federal de São Carlos
Departamento de Computação
Organização e Recuperação da Informação

Aplicação de Árvore B+

Prof. Jander Moreira

Alunos:
Luiz Fernando Daneluzzi de Souza
Marcel Garcia Will

São Carlos
01 de outubro de 2003.

Objetivos

Este seminário tem o intuito de, além de dar uma visão de implementação do trabalho sobre árvore B+, levantar algumas discussões sobre pontos que poderiam ter sido implementados de maneira diferentes das que foram.

Introdução

Árvores são estruturas para armazenamento de informação de alta capacidade e com a vantagem de se conseguir recuperar a informação de maneira rápida.

Existem vários tipos de árvores, sendo as mais conhecidas: Árvore Binária, Árvores Binária de Busca Balanceada, AVL, Árvore B, Árvore B+ e outras variações.

Vamos estudar a árvore B+.

Uma árvore B+ de ordem N pode ter um máximo de N-1 dados dentro de um nó e possuir N ponteiros para filhos que esteja um nível abaixo. Os dados são inseridos somente nos nós folhas, que formam uma lista encadeada de dados, o que permite uma busca seqüencial dos dados armazenados. Os nós internos formam uma árvore B.

Classe dataItem

A classe dataItem representa as informações que serão armazenadas pela árvore. Ela possui um campo chave, com o tipo a ser especificado, devido ao uso de template, e um campo do tipo long, para armazenar a informação desejada.

Ambos os campos são públicos, de maneira a facilitar o acesso aos dados e a única função é o construtor da classe, que recebe como parâmetro um valor para a chave e outro para o dado.

Também por falta de tempo, não foi possível implementar uma função para saída dos dados da classe, o que permitiria que se utilizasse a chave e a informação como membros privados.

Class dataItem	
variáveis	DT key; long dData;
funções	dataItem(DT key, long dData);

Diagrama da classe dataItem

Classe node

A classe node representa os nós da árvore, tanto os nós internos quanto os nós folha, sendo que a diferenciação entre eles é feita por um campo inteiro, que quando possui a valor 1 indica que o nó é folha e quando possui valor 2, indica que o nó é interno.

Também foi possível essa implementação devido a similaridade das funções que cada tipo de nó necessita, além do que, no caso da árvore B+, a raiz começa como nó folha e após a primeira divisão ela se torna um nó interno, o que estava dificultando e muito a implementação da árvore.

Class node	
variáveis	int ORD; int nodeType; int numItems; node<NT>* arrayChild[MAXSIZE]; NT arrayKey[MAXSIZE-1]; dataItem<NT>* arrayItems[MAXSIZE-1]; node<NT>* nextNode; node<NT>* pParent;

funções	node(int nType);
	void connectNext(node<NT>* pNext);
	void connectChild(int childNum, node<NT>* pChild);
	node<NT>* disconnectChild(int childNum);
	node<NT>* getChild(int childNum);
	node<NT>* getParent();
	bool isLeaf();
	bool isFull();
	int getNumItems();
	void setNumItems(int nItems);
	NT getKey(int pos);
	dataItem<NT>* getItem(int pos);
	dataItem<NT>* findItem(NT key);
	void insertLeafItem(NT key, dataItem<NT>* pData);
	bool removeLeafItem(NT key);
	void insertInnerItem(NT key, node<NT>* pNode);
	void insertInnerItem(NT key, node<NT>* pCurrent, node<NT>* pNewNode);
	bool removeInnerItem(NT key);

Diagrama da classe dataItem

Classe bpTree

É a classe que define a árvore B+, sendo que é nela que estão as funções de busca, inserção, remoção e determinação do ponteiro para o próximo filho.

Tanto a pesquisa quanto a inserção são recursivas, sendo que a recurssão é feita através das funções findK e insertR.

O tratamento para o caso de divisão em um nível inferior é feito dentro das funções insert e insertR, após a volta da chamada para a inserção.

Class bpTree	
variáveis	node<TT>* pRoot; int ORD;

funções	bpTree();
	dataItem<TT>* find(TT key);
	dataItem<TT>* findK(TT key, node<TT>* pCurrent);
	void insert(dataItem<TT>* dData);
	void insertR(TT key, dataItem<TT>* pData, node<TT>* pCurrent, node<TT>* pNode, TT &promoted, bool &Up);
	bool remove(TT key);
	node<TT>* getNextChild(TT key, node<TT>* pCurrent);

Diagrama da classe dataItem

Comentários Adicionais

Alguns comentários devem ser feitos antes do término da apresentação sobre a implementação da árvore B+.

Tivemos muitas dificuldades para gerar o código referente a remoção dos itens da árvore B+, que aliado à proximidade da data de entrega do trabalho, acabou por não ser implementada.

O método utilizado para organizar as chaves dentro dos nós foi o BubbleSort. Este não é o método é o mais eficaz para a ordenação de um arranjo, mas em compensação é de fácil implementação.

Foi criado um ponteiro para um nó pai, pensando em utilizá-lo na remoção, o que, conseqüentemente, acabou por não ser utilizado.

No geral, tentamos uma implementação da árvore B+ voltada para que funcionasse, não primando por eficiência, já que os testes seriam realizados com uma pequena quantidade de chaves, o que não influenciaria demasiadamente na performance. Caso houvesse um maior espaço de tempo para a implementação, poderíamos ter alterado alguns pontos do trabalho.