Proiect structuri de date 2006 - 2007

Grupa:	**1000**	Nume:	**Ioniță**	Prenume:	**Cristian**
Număr temă:	**0**	Titlu temă:	**Stocuri**		

1. Ipoteze de lucru

Se presupune că numărul de tranzacții este mult mai mare decât numărul de produse (aproximativ 100 de tranzacții / produs) și tranzacțiile sunt distribuite relativ uniform.

Presupunem că operația cea mai frecventă este cea de introducere tranzacție. Operațiile de anulare tranzacție și cele care afectează lista de produse sunt rare. Operațiile de afișare au o frecvență de apariție medie.

2. Descrierea structurii de date folosite

Din descrierea problemei și ipotezele de lucru constatăm că operațiile critice sunt cele de regăsire a unui produs și a tranzacțiilor aferente acestuia (în special pentru ultima perioadă). Aceste operații sunt necesare pentru validare în cazul introducerii unei tranzacții și pentru afișarea rapoartelor.

Pentru operația cea mai frecventă (introducere tranzacție) trebuie să putem efectua rapid următoarele operații:

- verificarea existenței produsului
- determinarea stocului minim pentru perioada data tranzacției – prezent (care implică enumerarea tuturor tranzacțiilor referitoare la produsul respectiv din acest interval)
- alocarea numărului și inserarea tranzacției

Pentru a putea efectua rapid aceste operații vom proceda în felul următor: construim o structură pentru produs care să conțină datele produsului și o listă simplu înlănțuită cu tranzacțiile corespunzătoare ordonate descrescător după dată și tip. Produsele vor fi reținute într-un arbore binar de căutare folosind codul produsului ca și cheie. Structura *DateProiect* va conține un pointer către lista de produse și un întreg pentru ultimul număr de tranzacție.

Pentru accelerarea operației de aflare a stocului curent pentru produse vom reține pentru fiecare produs și stocul curent (recalculat după fiecare operație).Se observă că această structură permite o implementare eficientă și pentru celelalte comenzi specificate în descrierea proiectului.

3. Structură de date alternativă (ipoteze + descriere)

Presupunem că lista de produse este aproape fixă (adăugările și ștergerile de produse au loc extrem de rar).

În acesta caz putem stoca datele despre produse (inclusiv un pointer la lista de tranzacții) sortate după cod într-un vector alocat dinamic. Regăsirile se vor face executa cel puțin la fel de repede ca în cazul unui arbore binar de căutare folosind căutarea binară. Dezavantajul este că inserările și ștergerile de produse ar dura foarte mult și ar necesita multă memorie deoarece implică realocarea vectorului.

Pentru tranzacții se poate folosi o listă dublu înlănțuită în loc de listă simplă. Astfel am putea afișa eficient tranzacțiile în ordine cronologică, cu dezavantajul unui consum mai mare de memorie pentru legătura suplimentară și o complexitate mai mare a codului.

În cazul în care anularea de tranzacție ar fi mai frecventă operația se poate accelera considerabil folosind un arbore binar de căutare. Arborele ar conține ca informație în noduri codul tranzacției și un pointer la tranzacția din lista produsului. Ordonarea s-ar face pe baza codului. Această indexare, combinată cu utilizarea listelor dublu înlănțuite ar permite implementarea eficientă a operației de anulare tranzacție. Un avantaj suplimentar al acestei abordări ar fi faptul că putem efectua mai rapid și operația de salvare parcurgând arborele de indexare.

4. Descrieți implementarea pentru cea mai complexă comandă implementată
Cea mai complexă operație implementată este cea de înregistrare tranzacție deoarece presupune parcurgerea eficientă a mai multor etape: • identificarea produsului corespunzător • verificarea faptului că avem stoc suficient pentru înregistrarea tranzacției • găsirea poziției unde trebuie inserată tranzacția • alocarea numărului și inserarea tranzacției Identificarea produsului se prin căutare pe arborele binar de produse. Pașii 2 și 3 se execută simultan: se parcurge lista de tranzacții folosind doi pointeri (unul pentru elementul curent și unul pentru elementul anterior) până la găsirea poziției de inserare, calculându-se în paralel și stocul minim pentru perioadă. Datorită faptului că lista de tranzacții este sortată după perechea (data descrescător, tip tranzacție) știm sigur că am luat în calcul toate tranzacțiile necesare, fără nici o operație în plus. Alocarea numărului se face pe baza ultimului număr alocat reținut în structura *DateProiect*.
5. Descrierea modului de implementare în cazul în care volumul de date ar fi foarte mare
Vom presupune că volumul de date nu permite stocarea în memorie nici a listei de produse, nici a tranzacțiilor asociate unui produs. Pentru analiză vom reține ipotezele de la punctul 1, cu mențiunea că numărul de tranzacții / produs poate fi mult mai mare. Stocarea datelor pe disc se va face în două fișiere binare: un fișier pentru produse și unul pentru tranzacții. În cadrul fișierelor datele sunt stocate sub formă de înregistrări de dimensiune fixă într-o ordine oarecare. Adăugările se vor face la sfârșitul fișierului, iar ștergerea se va realiza prin interschimbarea elementului de șters cu ultimul element din fișier și reducerea dimensiunii fișierului. Pentru implementarea eficientă operațiilor se vor crea trei fișiere index utilizând arbori B: • un fișier index pentru produse cu cheia codul produsului • un fișier index pentru tranzacții folosind tripletul (cod produs, data în ordine inversă, tip tranzacție) ca și câmp cheie (util pentru a putea determina ușor stocul pentru un produs la o dată specificată, operație necesară pentru înregistrare tranzacții și rapoarte) • un fișier index pentru tranzacții folosind numărul tranzacției ca și cheie (este util pentru a putea găsi o tranzacție pentru anulare și pentru a putea determina rapid numărul ultimei tranzacții) Toate operațiile care modifică datele vor modifica atât fișierul de date cât și fișierele de index corespunzătoare. Se vor folosi aceeași algoritmi cu cei utilizați pentru structura de date prezentată la punctul 2 (căutările în arbore și parcurgerile de liste simple vor fi înlocuite cu parcurgeri sau căutări pe arborii B din fișierele index).
6. Observații și comentarii
Toate operațiile specificate în descriere au fost implementate. Datorită faptului că tranzacțiile sunt reținute într-o listă simplu înlănțuită ordonate în ordinea inversă a datelor nu putem să afișăm eficient tranzacțiile decât în ordine inversă (pentru comanda fisa_produs). Pentru salvarea și încărcarea datelor puteam folosi următoarea alternativă stoca toate comenzile executate într-un vector de șiruri de caractere în *DateProiect*. La salvare se scria un fișier text cu toate comenzile, iar la încărcare se ștergeau toate datele din *DateProiect* și se executau comenzile din fișierul text salvat anterior.