Serijska i sekvencijalna organizacija memorije

(20)

Osnovna struktura
– slogovi smešteni jedan za drugim,uzastopne memorijske lokacije
• u sukcesivne memorijske lokacije
– fizička struktura ne sadrži informacije o vezama
između slogova logičke strukture datoteke
– ne postoji veza između vrednosti ključa sloga i adrese
lokacije u koju je smešten
– redosled memorisanja slogova najčešće prema
hronološkom redosledu njihovog nastanka u realnom sistemu

-ako hocemo da dobijemo vezu izmedju slogova,serijska organizacija nam nije pogodna
– slogovi mogu, a i ne moraju, biti blokirani

ODRAĐENO DO 10.slajda na prethodnom predavanju.

Obrada serijske datoteke – vodeća

-direktna i redosledna obrada
– može se koristiti kao vodeća u režimu direktne obrade
– može se koristiti kao vodeća u redoslednoj obradi
datoteke čiji ključ sadrži(ako je uredjenja saglasno neopadajucim vrijednosti tog stranog kljuca)
• ukoliko se ide na sekvencijalni pristup slogovima u
hronološkom redosledu

Obrada serijske datoteke – obrađivana

-sav postupak obrade podrazumijeva **vodecu datoteku** koja nam sluzi za generisanje kljuceva po kojoj se vrsi trazenje u obrdjivanoj datoteci

-u obradjivanoj datoteci se nalaze slogovima koje treba obraditi u postupku

-obrada-niz koraka koji se sprovodi nad podacima koji podrazumijevaju neke agregacije,sracunavanja,generisanje novih vrijednosti,upis,brisanje,modifikaciju slogova u samoj datoteci.

– program koji vrši redoslednu obradu serijske datoteke,neka aplikacija koji smo mi ili neko drugi napisali za obradu datoteke

-taj program koji vrsi obradu te serijske datoteke naprije:
• učitava sukcesivne slogove vodeće datoteke
• svaki naredni slog vodeće datoteke sadrži logički narednu vrednost ključa obrađivane serijske datoteke,uvijek kroz vodecu putem citanja slogova pristupamo vrijednosti kljuca naredne serijske datoteke

• te vrednosti ključa se koriste kao argumenti za traženje u serijskoj datoteci metodom linearnog traženja(ide se od pocetne lokacije I fizicki se pozicionirano na narednu lokaciju u obradjivanoj datoteci kako bismo dosli do sloga)
– u režimu direktne obrade
• sukcesivni slogovi vodeće datoteke sadrže slučajno odabrane vrednosti ključa obrađivane serijske datoteke
• traženje je, ponovo, linearno
– traženje logički narednog i slučajno odabranog sloga serijske datoteke
• obavlja se identično, krećući od prvog sloga datoteke

Obrada serijske datoteke – obrađivana

Vodeca datoteka sadrzi neki broj slogova NV;
– putem vodeće datoteke od *Nv* = *Nvu* + *Nvn* slogova
• N*vu* - slogova inicira uspešna traženja
• *Nvn* slogova inicira neuspešna traženja

Imamo neki broj slogova putem kojem dolazimo do broja slogova u obradjivanoj datoteci,a i nekih slogova u vodecoj koji dovode do neuspjesnog trazenje.

– inicira ukupan prosečni broj traženja(Ruk) -suma uspjesnih I neuspjesnih trazenja

-ako se predstavlja na osnovu broja blokova,tada je broj pristupa koji je potreban za uspjesno trazenje je broj blokova u serijskoj datoteci /2,a za neuspjesno je jednak citavom broju blokova u serijskoj datoteci

– broj pristupa se ne razlikuje za slučaj direktne i redosledne obrade

Ažuriranje serijske datoteke

– **upis novog sloga**
• u prvu slobodnu lokaciju na kraju datoteke(pronaci kraj datoteke i tu se pozicionirati,tu upisati novi slog,I pomjeriti za jednu lokaciju udesno za kraj datoteke)
• mora mu prethoditi jedno neuspešno traženje(da se ne bi duplirali podaci,jer moze doci do narusavanja ogranicenja kljuca,potrebno prvo provjeriti da li postoji taj isti kljuc)
• jednostavan, ali zahteva veliki broj pristupa
– svaki *f*-ti put neophodno je proširiti datoteku novim blokom
• očekivani broj pristupa

Ri moze biti:

n-broj slogova

Rn+1-ako se mora dodati jedan pristup,u postojuci blok, za kraj datoteke (1)

Rn+2-ako se mora dodati novi blok onda imamo,pristup kraju datoteteke i dodavanje novog bloka(2)

Ako imamo vec jedan blok sacuvan u datoteci,kako bih upisali oznaku za kraj datoteke,poslije upisa slogova,prosirujemo datoteku novim blokom(+1 pristup)

• očekivani broj pristupa

Za upis novog sloga prvo se mora proci kroz cijelu datoteku da bismo imali neuspjesno trazenje da nismo pronasli taj slog

-Ri=Rn+1(za oznaku za kraj datoteke)+1/f(za svaki f-I pristup za dodavanje novog bloka)

PRVA SLOBODNA LOKACIJA JE UVIJEK KRAJ DATOTEKE

Ažuriranje serijske datoteke

– **brisanje postojećeg sloga**

• mora mu prethoditi jedno uspešno traženje(pronaci taj slog)

• najčešće samo **logičko brisanje** – izmenom statusa aktuelnosti sloga(statusni slog)

• fizičko brisanje zahteva veliki broj pristupa(prije pristupu tom slogu koji zelimo da obrisemo moramo da pristupimo svim slogovima prije njega,kad se dodje do tog sloga,poslije njegovog brisanja,moramo da pomjerimo sve slogove za jedno mjesto unaprijed kako bi se to popunilo,te to znaci da se mora pristupiti svim slogovima)

– **modifikacija sadržaja postojećeg sloga**

• mora mu prethoditi jedno uspešno traženje

– očekivani broj pristupa za

• logičko brisanje ili

• modifikaciju sadržaja sloga(prvo pristup slogu pa onda modifikacija)

***Rd*= *Ru* +1 (Ru-uspjesan pristup + 1 za azuriranje)**

kad se logicki obrisu da se i fizicki obrisu,da dovedemo datoteku u azurno stanje,ako je dovoljno mala moze i fizicko brisanje

Oblasti primene i ocena karakteristika

– pogodne kao male datoteke

• kada mogu stati cele u OM

• zbog veoma velikog broja pristupa potrebnog za pronalaženje logički narednog ili slučajno odabranog sloga(ne postoji veza izmedju logicki susednih) zbog toga se serijska organizacija izbegava
• druge vrste organizacije donose samo mala poboljšanja u efikasnosti obrade malih datoteka
– serijska organizacija podataka moze da se prilagodi u kombinaciji sa indeksnim strukturama(direktan pristup slogovima,direktna obrada-imamo vodecu datoteku sa indeksnom strukturom I dalje se pristupa direktno na osnovu kljuca vodece datoteke koja ima indeksnu organizaciju)
• veoma pogodna za direktnu obradu
• osnovna fizička struktura relacionih baza podataka
– serijska datoteka kao rezultat obuhvata podataka(odlicna za prihvat podataka)
• polazna osnova za izgradnju datoteka sa drugim vrstama organizacije p

Sekvencijalna organizacija

Osnovna struktura

– slogovi su smešteni sukcesivno jedan za drugim
– logički susedni slogovi smeštaju se u fizički susedne lokacije
• postoji informacija o vezama između slogova logičke strukture podataka datoteke, ugrađena u fizičku strukturu
• realizovana kao linearna logička struktura podataka,svaki logicki slog ima jedno logickog prethodnika,a najvise jednog sljedbenika
– smeštanjem sloga sa većom vrednošću ključa u lokaciju sa većom adresom(rastuce uredjenje)
• rastuće uređenje po vrednostima ključa $\Rightarrow$ slog sa najmanjom vrednošću ključa smešta se u prvu lokaciju(moze i upadajuce uredjenje)
– naziva se i fizički sekvencijalnom organizacijom

Sekvencijalna datoteka sadrzi blokove

– veza između memorisanih vrednosti ključa $k(S)$ i adresa lokacija
• nije ugrađena u strukturu datoteke

• ne predstavlja bilo kakvu matematičku funkciju
– slogovi se smeštaju u blokovima od po *f* (≥1) slogova
• poželjno da faktor blokiranja *f* bude što veći,da moze da se namapira velicina logickog bloka na fizicki blok
– savremeni OS (*Unix*) i programski jezici (*C*, *C++*, *Java*) podržavaju samo serijski način pristupa
• korisnicima je ostavljeno da naprave svoje sopstvene sekvencijalne metode pristupa

Mala sekvencijalna datoteka(Dsec)

-polja koja sadrze dio kljuca i polja kao nisu dio kljuca

slogova *N* = 13
– faktor blokiranja *f* = 3
– slogovi
• isti sadržaj kao i Dser
• dvojke (*k*(*Si*),*p*(*Si*))
• *k*(*Si*) - vrednost ključa
• *p*(*Si*) - konkretizacija ostalih obeležja sloga *Si* (*i = 1,...,13*)
• oznaka kraja datoteke: ∗
• indeksi *i* (*i=1,...,13*) ukazuju na logički redosled smeštanja slogova

A1
03 ***p(S 1)*** **07** ***p(S 2)*** **13** ***p(S 3)***
A2
15 ***p(S 4)*** **19** ***p(S 5)*** **23** ***p(S 6)***
A3
25 ***p(S 7)*** **27** ***p(S 8)*** **29** ***p(S 9)***
A4
34 ***p(S 10)*** **43** ***p(S 11)*** **49** ***p(S 12)***
A5
64 ***p(S 13)*** ***

Sada se dodavanje novog sloga vrsi,tj upisuje npr 16 upisuje izmedju 15 i 19,tj 16 se upisuje na mjesto 19 te se sve ostalo iza njega siftuje za jedno mjesta do kraja.

Slog se ne razlikuje u odnosu na serijsku,sadrzi polja kljuc I polja koja su neprimarna(ostala obiljezja)

Formiranje sekvencijalne datoteke

– najčešće sortiranjem serijske datoteke
– slogovi,saglasno rastućim ili opadajućim vrednostima ključa

Traženje sloga u sekvencijalnoj datoteci

• logički narednog ili
• slučajno odabranog
– **traženje slučajno odabranog sloga**
• moguća primena metoda
– linearnog traženja(krenemo od pocetka dok ne stignemo da nekog sloga koji nam sadrzi argument trazenje po kljucu)
– binarnog traženja(u n iteracija prepolovi sadrzaj i nadje slog koji sadrzi taj kljuc)
• nema praktičnog smisla ako je datoteka velika i smeštena na eksterni memorijski uređaj
• ima praktičnog smisla ako je cela datoteka smeštena u OM
– nju, u tom slučaju, može predstavljati
» neka linearna struktura nad skupom slogova ili
» blok neke druge datoteke, npr. indeks-sekvencijalne(ucitavanje jednog dela indeks-sekvencijalne i trazenje slucajno odabranog sloga u ucitanom dijelu)

– **traženje logički narednog sloga**
• linearnom metodom traženja
• počevši od tekućeg, fizički susedni blokovi se učitavaju u OM
• u centralnoj jedinici se vrši upoređivanje argumenata traženja i vrednosti ključa sukcesivnih slogova dok se(kad se ZAUSTAVLJA)
– traženi slog ne pronađe
– argument traženja ne postane manji od vrednosti ključa sloga

– ne dođe do kraja datoteke (zaustavi se npr pri trazenju 16 ako prije 19 nije onda ga ni nema)

• traženje novog, logički narednog sloga, započinje od sloga
na kojem se prethodno traženje zaustavilo(od tekuceg sloga datoteke)

– **traženje logički narednog sloga**
– broj pristupa pri uspešnom i pri neuspešnom traženju

$$0 \leq R \leq B - i$$

• *i* - redni broj tekućeg bloka u odnosu na početak,B-broj blokova

-sada se pristupa blokovima(osnova za sekvencijalnu)
– broj poređenja argumenata traženja i vrednosti
ključeva slogova, pri uspešnom i neuspešnom
traženju

$$1 \leq U \leq N - i + 1$$

• *i* - redni broj tekućeg sloga

-N-ukupan broj sloga

+1 zbog oznake kraja datoteke

Obrada sekvencijalne datoteke
– **vodeća datoteka u direktnoj i redoslednoj obradi**
• česta upotreba
• sukcesivno učitavanje fizički susednih slogova, počevši od
prvog pa do poslednjeg(ukoliko je potrebno proci kroz sve slogove vodece da bi se doslo do sloga u obradjivanoj)
• ukupan broj pristupa, kada se sekvencijalna datoteka koristi
kao vodeća u obradi,je jednak broju blokova = (N+1)/f ; f-faktor blokiranja

Obrada sekvencijalne datoteke - obrađivana
• redosledna
• direktna
– **direktna obrada**
• ima smisla ako je sekvencijalna datoteka mala, tako da se
može smestiti u operativnu memoriju
• performanse obrade malo se razlikuju od performansi obrade serijske

datoteke

Opet imamo broj slogova koji inicira uspjesno i neuspjesno trazenje i prosecan broj pristupa.

Prosecan broj pristupa za neuspjesno trazenje je B/2;

– **redosledna obrada**
• iterativan proces
• vodeća datoteka generiše logički naredne vrednosti ključa za traženje u obrađivanoj, sekvencijalnoj datoteci
– svaki korak obrade = traženje logički narednog sloga
• vrši se metodom linearnog traženja
– svaki blok datoteke učitava se u OM samo jedanput,I kadi h ucita onda dalju obradu podataka radimo nad podacima koji su ucitani
– vodeća datoteka sadrži N v ($Nv \geq 1$) slogova
• uključuje vrednost ključa veću ili jednaku najvećoj vrednosti ključa u obrađivanoj datoteci

<u>Ažuriranje sekvencijalne datoteke</u>
– **upis novog sloga**
• pronalaženje mesta upisa novog sloga – implementira se u postupku neuspešnog traženja
– lokacija sloga sa prvom većom vrednošću ključa od datog
• pomeranje za jednu lokaciju udesno svih slogova sa vrednostima ključa većim od vrednosti ključa novog sloga
– **brisanje postojećeg sloga**
• prethodno pronalaženje sloga – uspešno traženje
• pomeranje za jednu lokaciju ulevo svih slogova sa većom vrednošću ključa, ako se brisanje vrši fizički
– **modifikacija sadržaja sloga**
• prethodno pronalaženje sloga – uspešno traženje
– upis i brisanje: ozbiljan problem ukupnog broja pristupa

Azuriranje se vrsi:

– **u režimu direktne obrade**
• u proseku, pomeranje polovine od ukupnog broja slogova za jednu lokaciju udesno (pri upisu) ili ulevo (pri brisanju) sloga
• primenjuje se kada je kompletna datoteka smeštena u OM

Za svaki pristup vrsimo pomeranje slogova u zavisnosti da li radimo dodavanje ili brisanje
– **u režimu redoslednje obrade**
• poseban iterativni postupak
– kreiranje potpuno nove datoteke, na osnovu postojeće
• primeren kada se datoteka ne može kompletno smestiti u operativnu memoriju
• datoteke i uloge u obradi
– *Ds* - obrađivana, ulazna (stara) sekvencijalna datoteka
– *Dn* - obrađena, izlazna (nova) sekvencijalna datoteka
– *Dp* - vodeća datoteka promena, serijska, ulazna
– *Dg* - datoteka grešaka, izlazna

Primjer na predavanju,vrijeme : 69.minut

– format sloga datoteke *Ds* i *Dn* identičan (*k*(*Si*), *p*(*Si*))

Ds -vrijednosti kljuca,Dn-neprimarno polje,mora da sadrzi azuriranje
– format sloga datoteke promena *Dp*: (*k*(*Si*),*pp*(*Si*),*sp*(*Si*))

Dp-smjestanje promjena koji izvrsi korisnik,prosiruje se jednim polje koje sadrzi treba da se odradi(d-delete,m-modify,i-insert)

Ona se sortira I tako prelazi iz serijske u sekvencijalnu(Dp->SORT->Dp).Tada program za uzuriranje uzima vrijednosti iz stare sekvencijalne datoteke I datoteke promena paralelno I generise novu sekvencijalnu datoteku.

• *sp*(*Si*) - polje statusa izvršene operacije, moguće vrednosti:
– *n* – novi slog, *m* – podaci za modifikaciju, *b* – slog za brisanje
– format sloga datoteke grešaka *Dg*: (*k*(*Si*), *p*(*Si*), *sg*(*Si*))
• *sg*(*Si*) - polje opisa greške, moguće vrednosti ukazuju na:

– pokušaj upisa već postojećeg sloga u datoteku
– pokušaj brisanja ili modifikacije nepostojećeg sloga datoteke

-ovim smanjen broj pristupa za svaku ovu operaciju(uradi se spajanje datoteke promene i stare sekvencijalne)

• sekvencijalni pristup sa učitavanjem slogova *Ss*(*Ds*) i *Sp*(*Dp*)
• upoređivanje vrednosti ključeva tekućih slogova
• generisanje novih slogova *Sn*(*Dn*) na osnovu sadržaja tekućih slogova *Ss* i *Sp*
• upis slogova *Sn* u datoteku *Dn*
– dužina intervala između dva ažuriranja
• određuje se tako da se tokom njega nakupi toliki broj promena koji bi opravdao pristupanje svim slogovima stare i generisanje nove datoteke
• duži interval $\Rightarrow$ veća efikasnost obrade, ali i duže vreme neusaglašenosti sadržaja datoteke sa realnim stanjem

– datoteka promena *Dp* sadrži *Nv* = *Nvn* + *Nvb* + *Nvm* slogova
– *Nvn* za upis, *Nvb* za brisanje i *Nvm* za modifikaciju

- *Bv* blokova: Bv=(Nv+1)/f;
– postojeća datoteka *Ds* sadrži *Bs* blokova: Bs=(N+1)/f;
– nova datoteka *Dn* sadrži *Bn* blokova: Bn=(N+Nvn+Nvb+1)/f;

-srednji broj pristupa pri ažuriranju datoteke za jedno traženje logički narednog sloga
$R=(Bv+Bs+Bn)/Nv;$

Oblasti primene i ocena karakteristika
– prednosti
• najpogodnija fizička organizacija za redoslednu obradu
• ekonomično korišćenje memorijskog prostora
• mogućnost korišćenja i magnetne trake i magnetnog diska, kao medijuma
– nedostaci
• nepogodnost za direktnu obradu

• potreba sortiranja pri formiranju
• relativno dugotrajan postupak ažuriranja

– najpogodnija fizička organizacija za redoslednu obradu
• režim redosledne obrade često se koristi u praksi, u paketnoj (batch) obradi podataka(u jednom ucitavanju dobija f slogova iz jednog bloka I predstavlja dalje podatke za obradu)
• posledica činjenice da su logički susedni slogovi smešteni u fizički susedne lokacije
• učitavanjem jednog bloka u OM, pribavlja se *f* slogova koji najverovatnije učestvuju u narednim koracima obrade
• poželjno je da *f* bude što veći
• kada *N*
$v \rightarrow N$, tada $R \rightarrow 1 / f$, te se s povećanjem *f* poboljšava efikasnost obrade