7.PREZENTACIJA

Datoteka

Datoteku mozemo posmatrati kao LSP i kao FSP.
Kao LSP, ona je struktura nad skupom pojava jednog tipa entiteta ili struktura slogova nad datim tipom sloga.Cesto se posmatra kao linearna struktura slogova.
Kao FSP, predstavlja jednu LSP smestenu na eksterni memorijski uredjaj zajedno sa informacijama o nacinu smestanja LSP na uredjaj.Moze biti vidjena kao linearna struktura ili niz slogova, ili kao niz znakova ili bajtova.
Motivacija za upotrebu eksternih memorijskih uredjaja :
- potreba trajnog memorisanja podataka, potreba memorisanja velikih kolicina podataka, potreba tolerantno brzog pristupa i operativnog koriscenja velike kolicine trajnog memorisanja podataka, potreba postizanja niske cene memorisanja po jedinici kapaciteta.

Izbor memorijskog uredjaja:
1. OM je nepogodan izbor
Prednosti:
- kada bi cela datoteka mogla stati odjednom u OM, svi postupci vezani za obradu i organizovanje podataka sveli bi se na teoriju algoritama i struktura podataka u OM
- kratko vreme pristupa svakoj celiji (reda x 10ns) kao najmanjoj adresibilnoj jedinici
- RAM pristup - vreme pristupa ne zavisi od polozaja lokacije na memorijskom medijumu, opredeljeno je radom elektricnih komponenti

Nedostaci:
- nedovoljan kapacitet
- nemogucnost trajnog memorisanja podataka
- znacajno skuplje memorisanje po jedinici kapaciteta

2. Magnetni disk
Prednosti:

- veliki kapacitet
- mogucnost trajnog memorisanja podataka
- znacajno jeftinije memorisanje po jedinici kapaciteta
- direktan pristup - pristupa se grupi celija direktno a zatim svakoj celiji u grupi sekvencijalno, postoji mogucnost operativne upotrebe podataka

Nedostaci:

- vreme pristupa zavisi od polozaja lokacije na memorijskom medijumu i bitno je duze nego u slucaju OM
- sekundarni tip uredjaja - nemogucnost direktnog prihvatanja podataka od strane centralnog procesora

Jedinice magnetnih diskova

Opsta struktura memorijskog uredjaja sastoji se od upravljacke jedinice uredjaja i jedinice za memorisanje podataka.**Upravljacka jedinica uredjaja** sadrzi upravljacku logiku,adresni registar uredjaja,registar podataka uredjaja,registar statusa uredjaja,sklop za vremensko vodjenje uredjaja.**Jedinica za memorisanje podataka** sadrzi adresni mehanizam,memorijski medijum,pobudna kola i izlazne pojacavace.

Magnetni disk je karakteristicni predstavnik eksternih memorijskih uredjaja s rotacionim kretanjem medijuma.Ostali predstavnici su CD/DVD jedinice.Princip organizacije adresnog prostora na memorijskom medijumu i adresnog mehanizma je isti, razlicita je tehnologija memorisanja podataka.Memorijski medijum su jedna ili vise kruznih ploca na rotirajucoj osnovi sa slojem feromagnetskog materijala sa obe strane.Adresni mehanizam je komplet upisno-citajucih glava na nosacu(+ koracni motor za pokretanje).

Organizacija adresnog prostora je zasnovana na cilindricnom koordinatnom sistemu,koordinate - poluprecnik,ugao i visina.

Staza je kruznica koju opisuje upisno-citajuca glava na zadatom poluprecniku.Podaci se upisuju na stazu poduzno,u obliku niza bitova.Celija diska = 1 bit - najmanja jedinica upisa/citanja podataka.

Vrste diskova s obzirom na kapacitet staze :

1. diskovi sa stazama konstantnog kapaciteta - promenljiva poduzna gustina zapisa.Najveca na stazi najmanjeg poluprecnika i obratno.Gubici u postizanju maksimalnog moguceg kapaciteta.
2. diskovi sa stazama promenljivog kapaciteta - konstantna poduzna gustina zapisa.Najveca moguca na stazi ili grupi susednih staza, komplikovanija organizacija adresnog prostora. Priblizavanje maksimalnom mogucem kapacitetu.
Cilindar je matematicki, skup svih staza istog poluprecnika.Prakticno, disk sadrzi 1-2 ploce, odnosno 3-4 R/W glave, veci broj staza tj. veci broj glava po jednom cilindru se postize softverskim putem.
Sektor je luk na stazi, konstantnog ugla.Staza se deli na konstantan broj sektora.Neretko, broj sektora na stazi je S = 2 na n. Izmedju svaka 2 sektora postoji medjusektorski razmak.Sektor je najmanja adresibilna jedinica diska.Svakom sektoru se pristupa direktno a svakom bitu unutar sektora sekvencijalno.
Sektorska organizacija adresnog prostora je kod diskova sa stazama konstantnog kapaciteta definisana je podelom staze na konstantan broj sektora konstantnog kapaciteta.Uspostava adresnog prostora diska sastoji se od fabricke pripreme i formatiranja diska od strane OS. Fabricke karakteristike su :
C - ukupan broj cilindara diska (diktira maksimalni kapacitet diska), T - ukupan broj staza po cilindru (tipicno T >= 16), S - ukupan broj sektora na stazi (tipicno S >= 64).Adresni prostor je diskretizovan numeracijom cilindara,staza i sektora.Numeracija cilindara je od 0 do C - 1, nulti je najveceg poluprecnika.Numeracija staza na jednom cilindru je od 0 do T - 1 odredjena redosledom glava na nosacu.Numeracija sektora na jednoj stazi je od 0 do S - 1, svaki sektor u zaglavlju ima upisan svoj redni broj, jedan sektor je odabran da bude pocetni i on oznacava pocetak staze.
Adresa sektora na disku - (u,c,t,s) gde je u - adresa uredjaja, c - redni broj cilindra, t - redni broj staze na cilindru, s - redni broj sektora na stazi.
Kapacitet diska:
Kapacitet sektora - konstanta

- Ks - efektivni kapacitet sektora, obuhvata prostor za korisne podatke tipicno je 512B.
- Ks na h - kapacitet zaglavlja sektora, obuhvata prostor za upisivanje

identifikacionog broja sektora i identifikacionog broja zamenskog sektora ukoliko je dati sektor van upotrebe.

- Ks na e - kapacitet prateceg dela sektora obuhvata prostor za kontrolni kod za detekciju i korekciju gresaka, garantuje verovatnocu nastanka do npr 1 / 10na15 neoporavljivih gresaka.

- Ks na u - ukupni kapacitet sektora - Ks na u = Ks + Ks na h + Ks na e Efektivni kapacitet staze - Kt = S*Ks. Efektivni kapacitet cilindra - Kc = T*Kt. Efektivni kapacitet diska - Kd = C * Kc.

Vreme pristupa sektoru:

To je vreme pristupa podacima na disku.Komponente, za zadatu adresu (u,c,t,s):

- vreme pozicioniranja kompleta glava na zadati cilindar c - opredeljeno je brzinom kretanja kompleta glava po poluprecniku, mehanicko kretanje.Reda velicine ms.
- vreme aktiviranja R/W glave za zadatu stazu t - opredeljeno je radom elektronickih komponenti.Bar za par redova velicina krace - zanemaruje se.
- vreme pozicioniranja R/W glave na pocetak zadatog sektora s - rotaciono kasnjenje - opredeljeno je brzinom rotacije paketa diskova, mehanicko kretanje.Reda velicine ms.

$t_p = t_c + t_h + t_r$

1. $0 <= t_c <= t_c$ na max - za predjeni put kompleta glava $i \in \{0,...,C-1\}$:
- iskazuje se brojem predjenih cilindara od cilindra na kojem su glave prethodno bile pozicionirane do trazenog cilindra
- t_c se moze posmatrati kao linearna funkcija od i
- za i = 0 je 0ms, za i = 1 je 0.8 - 2ms, za i = C-1 je 14 - 24 ms

Najbolje je da sukcesivno trazeni podaci budu smesteni na istom ili bar susednom cilindru.

2. $t_h \approx 0$
3. $0 < t_r <= 1/w$ - zavisi obrnuto proporcionalno od ugaone brzine rotacije, slucajna je velicina sa uniformnom raspodelom na (0, 1/w]

Srednje vreme pristupa sektoru je jednako zbiru srednjeg vremena pristupa cilindru i srednjeg rotacionog kasnjenja, tipicno je 9-12 ms.Srednje vreme pristupa cilindru je tipicno oko 8ms a srednje rotaciono kasnjenje je 1/2w, za 7200 ob/min je 4,17ms a za 10800 ob/min je 2,78ms.

Zonsko-sektorska organizacija adresnog prostora kod diskova sa stazama promenljivog kapaciteta definisana je podelom staza na sektore.Svaki sektor je konstantnog kapaciteta.Broj sektora na stazi je promenljiv,zavisi od poluprecnika staze,staze veceg poluprecnika sadrze vise sektora.Staze se grupisu po broju sadrzanih sektora.**Zona** je grupa susednih staza sa istim brojem sektora.Tehnika organizacije adresnog prostora je *zoned bit recording ZBR/ multiple zoned recording.*

Zakljucak:

- za vise od 5 redova velicine duze vreme pristupa nego kod OM.Kod diska je srednje vreme pristupa 9 - 12 ms, a kod OM ciklus (ukljucuje i srednje vreme pristupa) je 60ns.

Potrebne mere:

- poboljsati efikasnost prenosa podataka kroz U/I podsistem - efikasno koriscenje propusnog opsega spreznog podsistema.
- smanjiti potreban broj pristupa unapredjivanjem sistema diskova i izborom pogodne fizicke organizacije.

Sprezni podsistem

Sprezni (UI) podsistem predstavlja sistem veza i algoritama za fizicki prenos podataka izmedju kontrolera periferijskog uredjaja i OM. Sadrzi :

- linije podataka, adresne linije, upravljacke i informacione linije

Osnovna karakteristika je propusni opseg i predstavlja moguci broj prenetih bajtova u jedinici vremena.Dominantno zavisi od broja linija podataka i ciklusa spreznog podsistema ali i efektivno od propusnog opsega i ciklusa jedinice diska.Zahteva definisanje fiksne jedinice prenosa podataka na nivou operativnog sistema.

Osnovni koncepti:

1. Fizicki blok - organizaciona jedinica memorisanja podataka.Nedeljiva jedinica smestanja podataka na jedinici eksternog memorijskog uredjaja.Osnovna jedinica alokacije prostora na eksternom memorijskom uredjaju, fiksnog kapaciteta.U slucaju jedinice diska, predstavlja niz sukcesivnih sektora na istoj stazi diska.

2.Blok podataka - organizaciona jedinica prenosa podataka.Osnovna jedinica prenosa podataka, fiksnog kapaciteta.
-Kapacitet bloka:
Fiksne velicine,definisana unapred na nivou operativnog sistema.Krece se u rasponu [512B, 8KB],tipicne velicine su 2KB, 4KB, 8KB.
-Motivacija za koriscenje je postizanje sto boljeg iskoriscenja propusnog opsega i olaksano upravljanje prenosom podataka.
-U/I podsistem vrsi prenos samo celih blokova podataka.Jedinica diska obezbedjuje smestanje i preuzimanje samo celih fizickih blokova podataka.
-Fizicki blok na disku zauzima uvek ceo broj fizicki susednih sektora na istoj stazi diska.Prednost je sto se garantuje pristup celokupnom sadrzaju bloka uz potrosnju najvise jednog vremena pristupa.Nedostaci:
1. spoljna fragmentacija prostora - neiskorisceni sektori na kraju staze - mogu sluziti kao rezervni
2. unutrasnja fragmentacija prostora - ne mora ceo blok biti zauzet iskljucivo korisnim podacima
- Sistemski bafer je prostor u OM koji se alocira za potrebe smestanja sadrzaja jednog bloka podataka.Pripada sistemskom delu OM.Podaci razmenjeni sa eksternim memorijskim uredjajem smestaju se u bafere OM samo u jedinicama blokova.Posledica je da se zahteva veci kapacitet OM jer se u OM putem blokova prenose i podaci koji korisniku nisu neophodni u obradi i nece biti preneti u memoriju korisnickog programa.

Kontroler jedinice diska:

Zadaci:
1. dekodiranje i izvrsavanje R/W komande, dobijene od CPU
2. prijem adrese fizickog bloka na disku
3. upravljanje adresnim mehanizmom u cilju pozicioniranja na trazenu adresu - izdavanje naloga upravljackoj logici uredjaja
4. konverzija sadrzaja bloka - iz bajt-serijskog oblika u niz memorijskih reci pri citanju i iz niza memorijskih reci u bajt-serijski oblik pri upisu podataka.
5. ispitivanje statusa spremnosti jedinice diska za predaju ili preuzimanje podataka
6. privremeno memorisanje sadrzaja bloka - "cache" memorija na jedinici diska,

tipicnog kapaciteta 16MB.

U/I podsistem za fizicki prenos podataka

Zadaci:

1. inicijalizacija prenosa podataka - uvek je realizuje CPU.Zadaju se vrste R/W operacije, adrese bloka podataka na disku, adrese bafera u OM, kapacitet bloka podataka za prenos
2. fizicka razmena podataka na relaciji kontroler - OM - iterativni postupak , razmena memorijska rec po rec
3. ispitivanje statusa spremnosti uredjaja

Vrste:

- klasicni 'programirani' prenos moze biti uslovni sa ispitivanjem statusa spremnosti uredjaja i bezuslovni bez ispitivanja statusa spremnosti uredjaja
- prenos iniciran prekidima
- Direct Memory Access (DMA) prenos - angazovanje DMA kontrolera za fizicku razmenu podataka i ispitivanje statusa spremnosti uredjaja
- prenos putem specijalizovanih procesora

Efikasnost razmene podataka

Parametri koji imaju dominantan uticaj :

- srednje vreme pristupa bloku na disku - t_p na sr = 9 - 12ms
- vreme ucitavanja/pisanja sadrzaja bloka na disk :
    - Kb kapacitet bloka
    - Kt = S*Ks efektivni kapacitet staze
    - Tb = Kb / (w*Kt) vreme ucitavanja/pisanja sadrzaja bloka
- propusni opseg diska - Vd = w*Kt [MB/s]

Cak i za vece vrednosti Kb je Tb << t_p na sr .

Kod sukcesivnog ucitavanja blokova jednog cilindra, potrebno vreme razmene podataka je t_u na A , a kod ucitavanja blokova slucajno rasporedjenih po razlicitim cilindrima diska potrebno vreme razmene podataka je t_u na B. t_u na B >> t_u na A. Ako se prenosi sadrzaj jednog bloka datoteke sa diska,potrebno vreme razmene podataka je t_u na CD a iz memorije je t_u CM. t_u na CD >> t_u na CM.OM je brzi

uredjaj.
Ako se prenosi sadrzaj sukcesivnih blokova sa diska,potrebno vreme razmene podataka je t_u na DD a iz memorije je t_u na DM. t_u na DD $\sim$ t_u na DM. Bitno poboljsana efikasnost razmene podataka sa diska.
Sistemi disk jedinica:
1. klasterske arhitekture sistema disk jedinica - vise nezavisnih jedinica diskova,povezanih jednim spreznim sistemom.Jedinstveni adresni sistem i nacini pristupa od strane razlicitih procesorskih jedinica u arhitekturi.
2. nizovi disk jedinica - Redudant Array of Independent Disks (RAID) sistemi - vise jedinica diskova koje se ponasaju kao jedna,redudantno memorisanje podataka,diskovi su po potrebi izmenljivi
Obezbedjuju razmestanje istih ili sukcesivno trazenih blokova na vise nezavisnih disk jedinica.

Performanse obrade podataka

Tehnike obezbedjenja dobrih performansi:
1.koriscenje uticaja tehnologije i tehnoloskih parametara (T)
2.projektovanje odgovarajuce arhitekture sistema diskova (A)
3.izbor odgovarajuceg OS i podesavanje parametara OS (O)
4.projektovanje odgovarajuce FSP datoteka (P)

Skracenje srednjeg vremena pristupa:
-izbor disk jedinica boljih proizvodjackih karakteristika , npr. vece brzine rotacije,kraceg vremena pristupa... (T)
-upotreba sistema diskova (RAID, klasteri) sa simultanim pristupom blokovima (A)
-rasporedjivanje slucajno,a sukcesivno trazenih blokova na razlicite disk jedinice (A+P)
Efikasno koriscenje propusnog opsega diska:
-izbor veceg kapaciteta bloka (O)
-izbor odgovarajuce FSP datoteke , saglasno potrebama programa (P)
-efikasnija upotreba raspolozivog kapaciteta bloka (P)

Minimizacija potrebnog broja pristupa:
-"kesiranje" dela sadrzaja diska u memoriji kontrolera (T)
-povecanje kapaciteta OM (T)
-"kesiranje" dela sadrzaja diska u OM (O)
-rezervacija veceg broja bafera za datoteku u OM (O)
-izbor odgovarajuce FSP datoteke, saglasno potrebama programa (P)
Skracenje vremena prenosa i obrade podataka:
-izbor spreznog podsistema boljih karakteristika, npr. veceg propusnog opsega (T)
-izbor OM boljih karakteristika , npr kraceg ciklusa (T)
-izbor CPU boljih karakteristika, npr. vise frekvencije, sa vise kes memorije ili vise procesorskih jezgara (T)

Organizacija datoteke i OS

Operativni sistem omogucava organizovanje razlicitih FSP datoteka na jedinicama diskova.Vodi racuna o organizovanju podataka i upotrebi svih datoteka na jedinicama diskova.Moze da pruza razlicite poglede na FSP datoteke:
-kao linearne strukture slogova - najcesce za potrebe korisnickih programa
-kao niza znakova ili bajtova - za potrebe sistemskih programa, a moze i za potrebe korisnickih programa
-kao (linearne ili neke drugacije) strukture blokova - za potrebe memorisanja i razmene podataka kroz U/I podsistem.
Organizacija podataka na jedinici diska:
-OS odrzava na jedinici diska strukture podataka o:
1. proizvodjackim karakteristikama same disk jedinice
2. ispravnim i neispravnim sektorima kao i o zamenskim sektorima za neispravne sektore
3. slobodnom i zauzetom prostoru (fizickim blokovima) na disku
4. katalogu (hijerarhijskoj strukturi foldera) sa pokazivacima na opise datoteka
5. sistemskoj i alokacionoj tabeli svake datoteke - sistemska tabela sadrzi osnovne podatke o datoteci.Alokaciona tabela sadrzi pokazivace na podrucja diska koja su alocirana za potrebe datoteke.

8.PREZENTACIJA

Usluge OS u organizaciji datoteka

Datotecki sistem OS-a - zadaci :
- upravljanje datotekama (FSP na eksternim memorijskim uredjajima)
- upravljanje i realizacija razmene podataka izmedju aplikativnih programa i datoteka
- obezbedjenje mehanizama zastite od neovlascenog pristupa datotekama i ostecenja podataka - u uslovima visekorisnickog i multiprogramskog rezima rada
- obezbedjenje podrske razlicitih pogleda na LSP datoteke - preslikavanja LSP <-> FSP , obavlja jedan deo ili sve zadatke prethodna 3 tipa

Usluge datoteckih sistema :
1. Usluge niskog nivoa - pokrivaju pobrojane zadatke ali obezbedjuju pogled na LSP datoteke samo kao na niz bajtova(znakova) i njeno preslikavanje u FSP niza blokova.
To su servisi koji iskljucivo pripadaju operativnom sistemu.
2. Usluge visokog nivoa - obavezno ukljucuju sve usluge niskog nivoa, obezbedjuju razlicite poglede na LSP datoteke kao razlicitih struktura nad skupom slogova, blokova pa i bajtova i njenih preslikavanja u FSP niza blokova, obezbedjuju izgradnju specijalnih pomocnih struktura za poboljsanje efikasnosti obrade podataka, obezbedjuju trazenja zasnovana na vrednostima podataka.
To su servisi koji mogu biti ugradjeni u operativni sistem(karakteristicno za OS kod mainframe racunara), pakete funkcija programskog jezika (tipicno za sve savremene programske jezike), sistem za upravljanje bazama podataka (ovi servisi su obavezna komponenta svakog SUBP).Takodje, mogu biti prepusteni nivou aplikativnog programa (najcesce u slucaju razvoja specijalizovanih aplikacija i koriscenja specijalizovanih procesorskih uredjaja za prikupljanje,razmenu i memorisanje podataka).
Usluge niskog nivoa su:
1.upravljanje prostorom eksternog memorijskog uredjaja
2.upravljanje katalogom

3.upravljanje fizickom razmenom podataka
4.obezbedjenje veze programa i datoteke
5.sistemski pozivi
Usluge visokog nivoa:
1.metode pristupa

Upravljanje memorijskim prostorom

Rutine za upravljanje prostorom eksternog memorijskom uredjaja - jedinice diska:
- uspostava adresnog prostora i fajl sistema - obuhvata formatiranje diska, niskog i visokog nivoa kao i kreiranje strukture podataka sa evidencijom slobodnog i zauzetog prostora diska
- odrzavanje strukture podataka sa evidencijom slobodnog i zauzetog prostora na disku - obuhvata alociranje slobodnog prostora na zahtev drugih rutina OS, dealociranje slobodnog prostora na zahtev drugih rutina OS, reorganizaciju slobodnog prostora na zahtev administratora sistema
- arhiviranje, restauracija i oporavak sadrzaja diska - backup,restore,recovery rutine

Upravljanje katalogom

Struktura kataloga:
- hijerarhijska struktura direktorijuma - struktura tipa stabla.Ima korenski direktorijum koji se formira automatski u postupku formatiranja diska.Svaki cvor u strukturi moze biti direktorijum ili datoteka.Uvodimo pojam tekuceg direktorijuma.
Formiran je na jednoj ili grupi jedinica eksternih memorijskih uredjaja s direktnim pristupom.Dozvoljava relativno i apsolutno referenciranje cvorova u strukturi.
Rutine za upravljanje katalogom:
- kreiranje, brisanje, preimenovanje i prevezivanje direktorijuma u strukturi
- izlistavanje i pretrazivanje sadrzaja direktorijuma i datoteka
- kreiranje i brisanje datoteka u direktorijumu - upotreba odgovarajucih

sistemskih poziva
- preimenovanje, kopiranje i premestanje datoteka u strukturi
- dodela i ukidanje prava pristupa nad direktorijumima i datotekama
- izmena nekih atributa datoteka

Upravljanje fizickom razmenom podataka

Rutine za upravljanje fizickom razmenom podataka:
1. rutine fizickog U/I (U/I supervizor, drajveri uredjaja)
2. upravljanje razmenom blokova kroz U/I podsistem, izmedju kontrolera eksternog memorijskog uredjaja i OM.Obuhvata:
- inicijalizacija fizickog prenosa podataka jednog fizickog bloka zadavanjem parametara prenosa
- razmena poruka sa programima za fizicki prenos podataka koji se realizuje bilo kao CPU ili DMA prenos
- prosledjivanje statusa uspesnosti obavljenog prenosa podataka

Obezbedjenje veze programa i datoteke

Upravljanje strukturama podataka o:
- eksternim memorijskim uredjajima (A)
- datotekama na eksternom memorijskom uredjaju (B)
- upotrebi datoteka u aplikativnim programima (C)
- datotekama u operativnoj upotrebi (D)
Tabela OS je struktura podataka za opis nekog resursa kojim OS operativno upravlja.Gore navedene strukture su predstavljene delom putem tabela OS.
Tabele OS koje su vazne za obezbedjenje veze programa i datoteke su :
- tabela uredjaja
- sistemska tabela datoteke
- alokaciona tabela datoteke
- tabela logickih imena datoteke
- tabela procesa

- tabela otvorenih datoteka
- tabela opisa datoteke

1.Podaci (A) o eksternom memorijskom uredjaju :
OS odrzava na jedinici diska strukture podataka o:
- proizvodjackim karakteristikama same disk jedinice (naziv,adresa i tip uredjaja,ukupan broj cilindra,staza po cilindru i sektora po stazi)
- numeraciji sektora
- ispravnim i neispravnim sektorima
- zamenskim sektorima za neispravne sektore
- definisanom kapacitetu bloka
- korenskom direktorijumu i sistemu kataloga
- slobodnom prostoru

Tabela uredjaja (TU) obuhvata zapis sa podacima neophodnim za koriscenje uredjaja.Formira se prilikom inicijalizacije OS ili montiranja uredjaja.Sadrzi podatke o karakteristikama uredjaja - disk jedinice.Sadrzi adrese rutina za upravljanje fizickim U/I za dati uredjaj.Koriste je rutine za upravljanje fizickim U/I.Primer je OS Unix - tabela drajvera uredjaja gde se svaki zapis odnosi na jedan uredjaj i sadrzi adresu drajvera uredjaja - reprezentuje deo TU a podaci o karakteristikama uredjaja se nalaze u drajveru.

2.Podaci (B) o datotekama na eksternom memorijskom uredjaju :
OS odrzava na jedinici diska strukture podataka o datotekama razmestenim po sistemu kataloga.

Sistemska tabela datoteke (STD) predstavlja trajan zapis o datoteci koji odrzava OS.Formira se prilikom kreiranja datoteke a unistava se prilikom brisanja datoteke.Ucitava se u OM kada se datoteka operativno koristi a modifikuje se prilikom izmena sadrzaja datoteke.Spregnuta je sa sistemom kataloga.
Sadrzi naziv datoteke, ekstenziju i verziju datoteke, vrstu datoteke, podatke o vlasniku i ovlascenjima za koriscenje datoteke, velicinu datoteke, datum i vreme kreiranja ili poslednje modifikacije sadrzaja, podatke o ostalim atributima kao i alokacionu tabelu datoteke.
Konkretan sadrzaj i struktura zavise od izabranog OS.

Alokaciona tabela datoteke (ATD) je mapa alociranog prostora diska za datoteku.Predstavlja neprazan niz parova tipa (pokazivac, broj blokova) gde je

pokazivac (c,t,s) adresa pocetka zone alociranog prostora a broj blokova ukazuje na velicinu zone alociranog prostora, iskazanu brojem fizickih blokova.Svako alociranje nove zone prostora izaziva formiranje novog para tipa (pokazivac,broj blokova) u nizu a svako dealociranje nepotrebne zone prostora izaziva brisanje para tipa (pokazivac,broj blokova) iz niza.
Predstavlja jedno resenje vodjenja evidencije o alociranom prostoru datoteke.

3.Podaci (C) o upotrebi datoteka u aplikativnim programima :
Omogucavaju vezu izmedju aplikativnih programa i OS.Formira ih kompajler a koristi i dopunjava OS na osnovu specifikacija upotrebe datoteka u programu.Nalaze se u delu OM rezervisanom za aplikativni program.Sadrzaj ovih podataka moze zavisiti od nivoa usluga OS.Tu mogu biti ukljuceni podaci vezani za format sloga ili bloka datoteke, zeljene nacine upotrebe datoteke i zeljene nacine pristupa podacima datoteke.

Tabela logickih imena datoteke (TLI) predstavlja niz parova tipa (ime datoteke, pokazivac) gde ime datoteke ukazuje na povezani naziv datoteke (iz STD) a pokazivac ukazuje na zapis sa podacima o otvorenoj datoteci.Indeks niza je redni broj specificirane datoteke,uobicajeno : 0 - standardna ulazna datoteka(pridruzena tastaturi), 1 - standardna izlazna datoteka(pridruzena monitoru) i 2 - standardna datoteka gresaka(pridruzena monitoru).

4.Podaci (D) o datotekama u operativnoj upotrebi :
Formira ih i koristi OS kada je potrebno operativno koriscenje datoteke iz aplikativnog programa.Obuhvata otvaranje datoteke - priprema datoteke za operativno koriscenje podataka od strane procesa OS - aplikativnog programa. Obezbedjuju uvezivanje podataka sadrzanih u TU, STD i TLI.Nalaze se u sistemskom delu OM kojim upravlja OS.

Tabela procesa (TP) sadrzi ID oznaku procesa i neophodne podatke o procesu kreiranom na osnovu aplikativnog programa.Ukljucuje podatke o trenutnom stanju procesa na nivou OS.Ukljucuje podatke iz TLI. U okviru nje imamo **tabelu otvorenih datoteka procesa (TOP)** koja obuhvata TLI u sistemskom delu OM sa pokazivacima prema zapisima o otvorenim datotekama.

Tabela otvorenih datoteka (TOD) predstavlja niz zapisa o otvorenim datotekama u celom sistemu.Svako otvaranje predstavlja jedan zapis u TOD.Indeks niza je redni broj otvorene datoteke u sistemu.Svaki zapis sadrzi podatke neophodne da

bi se obavljala razmena podataka izmedju datoteke i aplikativnog programa.
Sadrzaj svakog zapisa u TOD obuhvata moguce nacine koriscenja datoteke, pokazivac na tekucu poziciju u datoteci (tekuci pokazivac), niz pokazivaca prema rezervisanim sistemskim baferima, pokazivac prema TU/drajveru uredjaja u tabeli drajvera, pokazivac na tabelu opisa datoteke (TOS), veze izmedju blokova i sistemskih bafera u obliku (*rbr_bloka, status, bafer*) gde je *rbr_bloka* - redni broj ucitanog bloka u sistemski bafer, *status* - indikator izmene sadrzaja bloka nakon poslednjeg ucitavanja, *bafer* - oznaka sistemskog bafera u koji je blok ucitan.
Tabela opisa datoteke (TOS) predstavlja prekopirani sadrzaj STD sa ATD u OM.Azurira se tokom upotrebe datoteke.Prenosi se azurirani sadrzaj u STD pri zavrsetku rada s datotekom.

Sistemski pozivi

Pozivi rutina OS za upravljanje datotekama - karakteristike:
1.Pruzaju usluge niskog nivoa, tj obezbedjuju:
- pogled na datoteku kao na niz bajtova - obuhvata razmenu podataka izmedju aplikativnog programa i datoteke, koji su organizovani kao nizovi bajtova
- upravljanje blokovima datoteke - obuhvata grupisanje nizova blokova u bajtove kao i razmenu kompletnih blokova izmedju eksternih memorijskih uredjaja i sistemskih bafera u OM
- upravljanje sistemskim baferima - obuhvata naloge za rezervisanje i otpustanje bafera kao i evidenciju smestanja blokova u bafere
- nezavisnost aplikativnog programa od fizickih karakteristika eksternog memorijskog uredjaja - transformacija rednog broja bajta u redni broj bloka i rednog broja bloka u apsolutnu adresu bloka na disku
2.Vode racuna o karakteristikama datoteke:
- pocetak datoteke - oznacen pozicijom bajta sa rednim brojem 1 (alternativno 0)
- kraj datoteke - oznacen pozicijom poslednjeg bajta , uvecanom za 1 (alternativno samo pozicijom poslednjeg bajta)
- indikator tekuce pozicije (tekuci pokazivac, indikator aktuelnosti) - iskazan kao redni broj bajta na kojem zapocinje operacija (alternativno na kojem je zavrsila

prethodna operacija)
3.Podrzavaju sekvencijalni pristup bajtovima datoteke:
- pri operacijama ucitavanja/zapisivanja zahtevaju zadavanje ukupnog broja bajtova za operaciju.Automatski odrzavaju vrednost tekuceg pokazivaca.
4.Podrzavaju direktni pristup bajtovima datoteke:
- pri operacijama pozicioniranja zahtevaju zadavanje vrednosti tekuceg pokazivaca (rednog broja bajta u odnosu na koji zapocinje sledeca operacija)
5.Preuzimaju parametre poziva iz pozivajuceg okruzenja - aplikativnog programa ili okruzenja zaduzenog za pruzanje usluga visokog nivoa
6.Prosledjuju u pozivajuce okruzenje informacije o statusu izvrsenja sistemskog poziva:
- osnova za obradu izuzetaka
- izuzetak (exception) je dogadjaj koji izaziva prekid normalnog toka obrade podataka.Primeri:
    - greska pri ucitavanju ili zapisivanju podataka
    - pokusaj otvaranja nepostojece ili vec otvorene datoteke
    - pokusaj citanja nepostojecih bajtova (preko kraja datoteke)

Tipovi sistemskih poziva:
1. create - kreiranje datoteke
2. open - otvaranje datoteke
3. read - ucitavanje dela sadrzaja datoteke
4. write - zapisivanje podataka u datoteku
5. seek - pozicioniranje na zeljenu lokaciju
6. close - zatvaranje datoteke
7. sync - praznjenje izmenjenih bafera
8. delete - brisanje i unistavanje datoteke
9. truncate - brisanje sadrzaja datoteke
10. stat - preuzimanje informacija o datoteci

Create
- Sistemski poziv za kreiranje datoteke na osnovu zadatih parametara (naziv sa putanjom, ovlascenja, tip, velicina...). Unix : int creat (char *naziv,int ovlascenja). Kreira potpuno novu datoteku i otvara je za pisanje.Proverava sve uslove

neophodne za kreiranje datoteke (postojanje volumena, putanje, raspolozivog prostora), alocira inicijalni prostor za novu datoteku na disku, kreira STD sa ATD, formira zapis u direktorijumu i uvezuje ga sa STD, vraca podatak o uspesnosti operacije i fajl deskriptor.

Open

- Sistemski poziv za otvaranje datoteke na osnovu zadatih parametara (naziv sa putanjom, nacin otvaranja, eventualno i ovlascenja). Unix : int open (char *naziv, int nacin_otvar[int ovlascenja]).Obuhvata pripremu datoteke za operativnu upotrebu.U operativnoj upotrebi datoteke zahteva se pristup podacima u STD i TLI.Pogodno je smestiti te potrebne podatke u OM da bi se izbeglo repetitivno pristupanje disku i visestruko obavljanje istih pripremnih operacija za potrebe svake pojedinacne R/W operacije.Nacini otvaranja datoteke:
- otvaranje nepostojece datoteke - zahteva se inicijalno kreiranje datoteke (O_CREAT)
- otvaranje postojece datoteke
- otvaranje u rezimu dozvole citanja i pisanja podataka na zeljenoj poziciji (O_RDWR)
- otvaranje u rezimu ekskluzivnog citanja sadrzaja (O_RDONLY)
- otvaranje u rezimu ekskluzivnog pisanja sadrzaja sa dodavanjem novog sadrzaja na kraj datoteke (O_APPEND), sa prepisivanjem postojeceg sadrzaja datoteke (O_WRONLY), sa prethodnim brisanjem postojeceg sadrzaja datoteke (O_TRUNC).

Zadaci:

- ukoliko je specificiran zahtev kreiranja nove datoteke, sprovodjenje poziva tipa Create
- u slucaju zahteva za otvaranje postojece datoteke, provera ostvarenosti potrebnih uslova (postojanje trazenog volumena, specificirane putanje sa direktorijumom i trazenim fizickim nazivom datoteke kao i ovlascenja za izvodjenje zahtevane operacije tipa Open)
- prenos sadrzaja STD u OM (kreiranje TOS i uvezivanje sa STD)
- formiranje zapisa u TOD i uvezivanje sa TOS i TLI u TP (povezivanje fizickog imena i logicke oznake datoteke u TLI kao i povezivanje adresa odgovarajucih programa za opsluzivanje sistemskih poziva za razmenu podataka)

- inicijalizacija vrednosti tekuceg pokazivaca na pocetak datoteke ili na kraj (u slucaju O_APPEND)
- eventualno inicijalno rezervisanje sistemskih bafera i pocetno punjenje bafera - obuhvata upisivanje pokazivaca na bafere u odgovarajuci zapis u TOD kao i punjenje bafera pocetnim blokovima datoteke u slucaju otvaranja datoteke za citanje
- predaja pozivajucem okruzenju podataka o uspesnosti izvedene operacije i fajl deskriptora (logicke oznake datoteke)

Upravljanje sistemskim baferima obuhvata:
- rezervisanje i otpustanje bafera datoteke koje se vrsi koriscenjem servisa jezgra OS pri cemu otvorenoj datoteci mora biti dodeljen najmanje jedan bafer.
- evidentiranje smestanja blokova u bafere - u zapisu iz TOD ili u referenciranoj strukturi podataka koju odrzava jezgro OS

Tehnike rezervisanja i otpustanja bafera :
- fiksna (staticka) dodela bafera datoteci
- dinamicka dodela bafera datoteci (udruzivanje) - buffer pooling

Kod fiksne dodele bafera datoteci, fiksni broj bafera (>=1) se ekskluzivno dodeljuje pri otvaranju a svi baferi se otpustaju pri zatvaranju datoteke.Cesto se datoteci otvorenoj samo za citanje ili pisanje dodeljuje 1 bafer a datoteci otvorenoj i za citanje i za pisanje dodeljuju se 2 bafera (1 za citanje a drugi za pisanje).
Kod dinamicke dodele bafera datoteci, svi sistemski baferi OS-a su na raspolaganju svim otvorenim datotekama, baferi se dodeljuju i otpustaju dinamicki prema potrebi (pri realizaciji citanja ili pisanja).Po kriterijumu izbora slobodnog bafera za dodelu ili izbora bafera za otpustanje koristi se najduze nekorisceni bafer (Least recently used).

Visestruko otvaranje iste datoteke nije dozvoljeno kod nekih OS a kod nekih je dozvoljeno.
Kod prve grupe OS-a, datoteka se ekskluzivno otvara i zakljucava od strane jednog procesa.Za svaku datoteku u TOD moze postojati najvise jedan zapis, ostali procesi mogu samo eventualno otvoriti read-only kopiju datoteke za potrebe uvida u trenutno stanje datoteke.Ocuvana je konzistentnost podataka u visekorisnickom rezimu rada.Bitno je snizen stepen moguceg paralelizma u obradi

i koriscenja podataka datoteke.
Kod druge grupe OS-a, datoteka se moze otvoriti od strane vise razlicitih procesa istovremeno (open) ili od strane jednog procesa multiplikovano - sistemski pozivi dup i dup2.Svako otvaranje datoteke formira poseban zapis u TOD koji je uvezan sa odgovarajucim zapisom u TLI iz TP,precizno se evidentira koji proces je izvrsio koje otvaranje.Konzistentnost podataka u visekorisnickom rezimu ne mora biti ocuvana.Prisutan je visoki stepen moguceg paralelizma u obradi podataka. Primer je Unix.

Read

- Sistemski poziv za ucitavanje niza bajtova iz otvorene datoteke na osnovu zadatih parametara (logicka oznaka datoteke, odredisna promenljiva u OM, broj bajtova za ucitavanje).Unix : int read (int fd, void *var, int size).Imamo stvarni prenos dela sadrzaja datoteke u pozivajuce okruzenje.Prenosi se zadati broj bajtova od tekuceg pokazivaca. Zadaci:
- provera da li je datoteka sa oznakom fd otvorena i da li dozvoljava zahtevani nacin pristupa
- izracunavanje rednog broja prvog bloka i duzine niza izvornih blokova u kojima se nalaze trazeni podaci - na osnovu rednog broja bajta u tekucem pokazivacu koji se pretvara u par (redni broj prvog bloka u nizu, redni broj bajta u bloku)
- provera da li se trazeni blokovi vec nalaze u baferima
- ako ne, onda imamo :
    - izbor ili alokacija sistemskih bafera za smestanje blokova uz eventualno oslobadjanje prethodnog sadrzaja bafera
    - izracunavanje apsolutnih adresa trazenih blokova
    - inicijalizacija fizickog prenosa blokova sa jedinice diska
    - obrada statusa uspesnosti fizickog prenosa blokova
- prenos trazenog sadrzaja iz bafera u ciljnu promenljivu var
- uvecavanje vrednosti tekuceg pokazivaca na prvi sledeci bajt nakon poslednje prenetog bajta
- predaja pozivajucem okruzenju podataka o uspesnosti izvedene operacije i broju stvarno preuzetih bajtova

Write

- Sistemski poziv za zapisivanje niza bajtova u otvorenu datoteku na osnovu zadatih parametara (logicka oznaka datoteke, izvorna promenljiva u OM, broj bajtova za zapisivanje). Unix : int write (int fd, void *var, int size). Imamo stvarni prenos niza bajtova iz pozivajuceg okruzenja u datoteku.Prenosi se zadati broj bajtova od tekuceg pokazivaca. Zadaci:
- provera da li je datoteka sa oznakom fd otvorena, i da li dozvoljava zahtevani nacin pristupa
- izracunavanje rednog broja prvog bloka i duzine niza ciljnih blokova u koje treba smestiti izvorne podatke - na osnovu rednog broja bajta u tekucem pokazivacu koji se pretvara u par (redni broj prvog bloka u nizu, redni broj bajta u bloku)
- provera da li je potrebno imati prethodno dopremljen sadrzaj ciljnih blokova.U O_RDWR nacinu otvaranja tipicno jeste a u O_APPEND ili O_WRONLY nije.
- ako da, onda provera da li se trazeni blokovi vec nalaze u baferima a ako ne onda imamo izbor ili alokaciju sistemskih bafera za smestanje blokova uz eventualno oslobadjanje prethodnog sadrzaja bafera, izracunavanje apsolutne adrese trazenih blokova, inicijalizaciju fizickog prenosa blokova sa jedinice diska, obradu statusa uspesnosti fizickog prenosa blokova
- prenos trazenog sadrzaja iz izvorne promenljive var u bafere uz eventualno oslobadjanje sadrzaja kompletno napunjenih bafera - iniciranje fizickog prenosa blokova na jedinicu diska, buffer flushing
- uvecavanje vrednosti tekuceg pokazivaca na prvi sledeci bajt nakon poslednje prenetog bajta
- predaja pozivajucem okruzenju podataka o uspesnosti izvdene operacije i broju stvarno predatih bajtova

Seek

- Pozicioniranje na zeljenu lokaciju na osnovu zadatih parametara (logicka oznaka datoteke, pomak, referentna tacka za pomak). Unix : int lseek (int fd, int offset, int moveDirection).Upravlja se sadrzajem tekuceg pokazivaca u cilju obezbedjenja direktnog pristupa zeljenom bajtu datoteke. Zadaci:
- provera da li je datoteka sa oznakom fd otvorena
- postavljanje nove vrednosti tekuceg pokazivaca pomeranjem za zadati broj

bajtova - offset(prema kraju datoteke ako je offset > 0, prema pocetku datoteke ako je offset < 0 i bez pomeranja ako je offset = 0) u odnosu na referentnu tacku zadatu sa moveDirection (0 (SEEK_SET) - za pocetak datoteke, 1 (SEEK_CURR) - za trenutnu poziciju pokazivaca, 2 (SEEK_END) - za kraj datoteke).
- predaja pozivajucem okruzenju podatka o uspesnosti izvedene operacije

Close
- Sistemski poziv za zatvaranje otvorene datoteke na osnovu zadate logicke oznake datoteke.Unix : int close (int fd). Donosi uredan prestanak operativne upotrebe datoteke - u opstem slucaju garantuje da ce datoteka ostati memorisana na disku u konzistentnom stanju.Zatvaranje datoteke nastaje automatski zavrsetkom programa koji je otvorio datoteku, odnosno eksplicitno pokretanjem sistemskog poziva close. Zadaci :
- oslobadjanje sadrzaja zauzetih bafera modifikovanog sadrzaja - obuhvata iniciranje fizickog prenosa blokova na jedinicu diska.Dirty buffer - bafer u kojem je izmenjeno ili potpuno novo stanje bloka u odnosu na njegov sadrzaj na disku
- oslobadjanje svih zauzetih bafera i vracanje OS-u - ova i prethodna aktivnost nisu obavezne kod svih O.Unix close ne podrzava praznjenje i oslobadjanje bafera pri zatvaranju datoteke.
- prenos sadrzaja TOS sa ATD na jedinicu diska - azuriranje sadrzaja STD sa ATD
- unistavanje TOS i odgovarajuceg zapisa u TOD - raskidanje svih uspostavljenih veza prema TLI, TP i STD.
- predaja pozivajucem okruzenju podatka o uspesnosti izvedene operacije

Sync
- sistemski poziv za "praznjenje" dirty bafera. Unix : void sync(void). Izvrsava se na zahtev ili automatski u zadatim intervalima.

Delete
- Sistemski poziv za brisanje imena i unistavanje datoteke na osnovu zadatog naziva sa putanjom. Unix : brisanje imena fajla - int unlink(char *naziv).Moze u strukturi kataloga biti kreirano vise imena - linkova, brisanje poslednjeg linka brise i unistava datoteku.

Zadaci fizickog brisanja i unistavanja datoteke:
- dealociranje prostora datoteke na disku
- unistavanje STD sa ATD
- unistavanje zapisa o datoteci u direktorijumu
- vracanje podatka o uspesnosti operacije

Truncate
- Sistemski poziv za brisanje sadrzaja datoteke dealocira prostor datoteke na disku ali sadrzava STD.
int truncate (const char *path, off_t length)
int ftruncate (int fd, off_t length)
"Odsecanje" sadrzaja datoteke do na zadati broj bajtova.Brise sadrzaj datoteke i dealocira oslobodjeni prostor datoteke na disku.Uvek zadrzava STD.

Metode pristupa

Podrzavaju usluge visokog nivoa. -> pogledati pocetak ove prezentacije
Zahtevaju razresavanje pitanja :
- organizovanja i memorisanja polja, slogova i blokova
- nacina adresiranja i nacina memorisanja logickih veza
- mogucih vrsta usluga na nivou sloga ili bloka
- podrske razlicitih vrsta organizacije datoteka
- podrske opstih postupaka upravljanja sadrzajem datoteka
Servisi metoda pristupa mogu biti ugradjeni u :
1. operativni sistem
2. programski jezik sa pridruzenim paketima funkcija
3. sistem za upravljanje bazama podataka

9.PREZENTACIJA

Osnovna struktura datoteke

Datoteka kao struktura slogova organizovana je nad tipom sloga kao linearnom strukturom atributa. Format sloga obuhvata pravila za strukturiranje i interpretaciju sadrzaja sloga. Opsta struktura sloga datoteke kao FSP ukljucuje podatke iz LSP i podatke o organizaciji FSP na eksternom memorijskom uredjaju.Svaki slog predstavlja niz polja sa vrednostima atributa.
Opsta struktura sloga datoteke kao FSP obuhvata :
- polja vrednosti atributa primarnog kljuca $k_i(S)$, i = 1...
- polja vrednosti ostalih atributa $p_i(S)$, i = 0...
- polje statusa sloga - indikator aktuelnosti sloga u LSP s(S)
- polja pokazivaca za memorisanje veza u LSP $u_i(S)$, i = 0...
- kontrolna polja kod slogova varijabilne duzine $f_i(S)$, i = 0...

Skracena notacija : k(S) p(S) s(S) u(S) f(S)

Osnovna struktura datoteke:

- k(S) predstavlja jedinu obaveznu grupu polja koja se sastoji od najmanje jednog polja.Cesto se posmatra kao linearna struktura slogova uredjena u rastucem ili opadajucem redosledu vrednosti primarnog kljuca.Redosled polja u formatu sloga ne mora biti isti kao u opstoj strukturi sloga.Pozicija kontrolnih polja uslovljena je njihovom semantikom.

Format polja sloga je uslovljen specifikacijom domena odgovarajuceg atributa.

Vrste polja u slogovima:

- polja konstantne duzine - nije potrebno memorisati informaciju o granicama polja
- polja promenljive duzine - potrebno je memorisati informaciju o granicama polja, koristi se kontrolno polje f(S). Tehnike oznacavanja granica polja:
    - navodjenjem aktuelne duzine polja u kontrolnom polju neposredno ispred sadrzaja polja
    - navodjenjem specijalne oznake kraja polja u kontrolnom polju neposredno

nakon sadrzaja polja

Vrste slogova prema duzini:

- slogovi konstantne duzine - sva polja u svakom slogu su konstantne duzine i nije potrebno memorisati informaciju o granicama sloga

- slogovi promenljive duzine - postoji barem jedno polje promenljive duzine u slogu i potrebno je memorisati informaciju o granicama sloga za sta se koristi kontrolno polje f(S).Tehnike oznacavanja granica sloga:

    - navodjenjem aktuelne duzine sloga u kontrolnom polju neposredno ispred ostalog sadrzaja kompletnog sloga

    - navodjenjem specijalne oznake kraja sloga u kontrolnom polju neposredno nakon ostalog sadrzaja sloga

    - uvodjenjem posebne indeksne strukture sa rednim brojevima bajtova koji ukazuju na pocetke slogova

Karakteristike slogova konstantne duzine:

-pojavljuju se u praksi

-homogena struktura

-jednostavnije pristupanje podacima i azuriranje podataka

-laksa i preciznija procena performansi obrade podataka

-manja efikasnost upotrebe memorijskog prostora

Karakteristike slogova promenljive duzine:

-izuzetno cesto se pojavljuju u praksi

-nehomogena struktura

-teze pristupanje podacima i azuriranje podataka

-teza i nepreciznija procena performansi obrade podataka

-veca efikasnost upotrebe memorijskog prostora

Vrste slogova prema ponavljanju vrednosti:

- slogovi s ponavljajucim grupama - moraju uvek biti slogovi varijabilne duzine.Karakterise ih visestruko pojavljivanje vrednosti atributa u jednom slogu kada je dozvoljeno da jedna vrednost atributa bude predstavljena kao niz vrednosti istog tipa.

- slogovi bez ponavljajucih grupa - nije dozvoljeno visestruko pojavljivanje vrednosti atributa.Moguce je uvek projektovati tip sloga bez ponavljajucih grupa primenom odgovarajucih projektantskih tehnika.

Polja pokazivaca u strukturi sloga predstavljaju adrese lokacija u memorijskom prostoru.Vrste adresa lokacija:
1. apsolutna (masinska) adresa - strukturirana prema adresnom prostoru jedinice diska.Prakticno se ne koristi u organizaciji datoteka.Stvara zavisnost od fizickih karakteristika uredjaja i ne zahteva transformaciju.
2. relativna adresa - predstavlja redni broj lokacije a moze biti pracen rednim brojem podlokacije.Vrlo cesto se koristi u organizaciji datoteke i obezbedjuje nezavisnost od fizickih karakteristika uredjaja.Zahteva jednu ili vise transformacija do apsolutne adrese na nivou metode pristupa ili sistemskih poziva.
3. simbolicka (asocijativna) adresa - predstavlja vrednost kljuca.Cesto se koristi u organizaciji datoteka i zahteva transformaciju u relativnu adresu na nivou metode pristupa.

Struktura datoteke kao niza blokova

Blok (logicki blok) kao organizaciona jedinica podataka predstavlja niz slogova i ima konstantan kapacitet tipicne velicine : 2KB, 4KB, 8KB, 16KB, a najcesce predstavlja celobrojni umnozak kapaciteta fizickog bloka. Uobicajeno jedan blok predstavlja niz od 2 na n fizickih blokova, a nije nemoguce da kapacitet bloka bude jednak kapacitetu fizickog bloka ili cak manji od kapaciteta fizickog bloka.
Opsta struktura bloka - zaglavlje bloka i niz slogova:
Ai

Zaglavlje bloka	Ai na 1	...	Ai na j	...	Ai na f
	S1	...	Sj	...	Sf

Ai - adresa bloka (najcesce iskazana kao relativna)
Ai na j - relativna adresa j-tog sloga u i-tom bloku (i,j)
f - faktor blokiranja - broj slogova u bloku
Zaglavlje bloka je neobavezna kategorija, obuhvata podatke vezane za FSP datoteke, npr. razlicita polja pokazivaca, broj slogova u bloku, indeks na pocetke slogova...

Vrste blokova:

- blokovi sa slogovima promenljive duzine - vise slogova moze biti smesteno u jedan blok a dozvoljeno je i da velicina jednog sloga premasi kapacitet bloka, tada se vrsi ulancavanje blokova jednog sloga.
- blokovi sa slogovima konstantne duzine - homogena struktura bloka i datoteke. Svaki blok datoteke sadrzi uvek isti broj slogova.

f - faktor blokiranja datoteke, B - ukupan broj blokova datoteke, N - ukupan broj slogova u LSP datoteke. $B = \lceil (N + x) / f \rceil$ gde je x broj dodatno upotrebljenih specijalnih slogova.

Proracun potrebnog kapaciteta datoteke je moguc u slucaju primene blokova sa slogovima konstantne duzine. Ks - kapacitet sloga - predstavlja zbir kapaciteta svih polja, Kb - kapacitet bloka - unapred zadata konstanta, Kz - kapacitet zaglavlja bloka(zavisi od organizacije) - $f = \lfloor (Kb - Kz) / Ks \rfloor$, Kd - kapacitet datoteke (zavisi od organizacije) Kd = B*Kb + Wd gde je Wd kapacitet STD za datoteku

Struktura datoteke kao niza blokova je linearna struktura blokova datoteke gde svaki blok datoteke obuhvata niz slogova datoteke.

Strogo strukturirana datoteka je strogo tipizovana datoteka sa pridruzenom semantikom, organizovana kao struktura nad skupom slogova koja se preslikava u strukturu nad skupom blokova, a dalje se preslikava u strukturu nad nizom bajtova pa fizickih blokova.

Zaglavlje datoteke je potrebno prosirenje osnovne strukture datoteke.Uvodi se specijalni slog na pocetku datoteke sa podacima o organizaciji datoteke i formatu bloka i sloga datoteke.Ukljucuje podatke :

- broj slogova i blokova u datoteci, duzina i format sloga, pozicija polja kljuca u slogu, pokazivaci na pocetke spregnutih struktura slogova ili blokova

Oznaka kraja datoteke - nacini oznacavanja kraja datoteke u osnovnoj strukturi:

1. uvodjenjem specijalnog sloga za oznaku kraja datoteke - zapisuje se na kraju strukture iza poslednjeg sloga u LSP u prvu slobodnu lokaciju memorijskog prostora datoteke
2. uvodjenjem specijalne oznake kraja u polje pokazivaca - navodi se u polju pokazivaca logicki narednog sloga u(S) , kod poslednjeg sloga u LSP
3. vodjenjem posebne evidencije zauzetosti prostora - u pomocnoj strukturi podataka , memorisanjem npr. broja zauzetih lokacija u prostoru dodeljenom

datoteci
4. kraj datoteke je kraj prostora dodeljenog datoteci, ne uvodi se poseban mehanizam za oznacavanje kraja datoteke

Metoda pristupa

Paket programa za podrsku usluga visokog nivoa podrzava :
- upravljanje strogo strukturiranim datotekama - obuhvata upravljanje organizacijom i memorisanjem polja, slogova i blokova
- upravljanje baferima metode pristupa - visi nivo baferisanja u odnosu na nivo sistemskih bafera
- podrska razlicitih vrsta organizacije datoteka - obuhvata :
    - podrsku razlicitih nacina memorisanja logickih veza i adresiranja u strogo strukturiranim datotekama
    - vodjenje brige o kategorijama - zaglavlje datoteke, pocetak i kraj datoteke, tekuci pokazivac odnosno indikator aktuelnosti (iskazan kao relativna adresa bloka ili sloga na kojem se sprovodi operacija)
    - podrsku izgradnje specijalnih pomocnih struktura za poboljsanje efikasnosti obrade podataka
- podrska opstih postupaka upravljanja sadrzajem datoteka - kreiranje, trazenje, pretrazivanje, azuriranje i reorganizacija
- koristi ili ukljucuje usluge niskog nivoa izabranog OS u zavisnosti od mesta i nacina implementacije metode pristupa
- obezbedjuje nezavisnost aplikativnog programa od usluga niskog nivoa OS - obezbedjuje preslikavanje strogo strukturirane datoteke u FSP niza fizickih blokova, obezbedjuje transformacije relativne adrese sloga ili bloka datoteke u relativnu adresu bajta ili fizickog bloka

Upravljanje strogo strukturiranim datotekama obuhvata:
- podrsku organizacije slogova i polja konstantne i promenljive duzine
- podrsku razlicitih (alfanumerickih,datumskih) tipova podataka
- podrsku razlicitih kodnih rasporeda
- konverzije podataka iz tipa podatka programske promenljive u tip podatka

atributa datoteke i obrnuto, i iz tipa podatka atributa strogo strukturirane datoteke u niz bajtova i obrnuto
Kod ovog upravljanja, usluge razmene podataka sa aplikativnim programima mogu biti na nivou sloga i na nivou bloka.Na nivou sloga imamo grupisanje slogova u blokove pri upisu podataka, rastavljanje bloka na slogove pri citanju podataka, odrzavanje tekuceg pokazivaca kao relativne adrese sloga i njegovu transformaciju u oblik (redni broj bloka u datoteci, redni broj sloga u bloku).Na nivou bloka imamo razmenu sadrzaja kompletnih logickih blokova izmedju aplikativnog programa i datoteke, odrzavanje tekuceg pokazivaca kao relativne adrese bloka, u obliku rednog broja bloka u datoteci.
Pristup podacima iz aplikativnog programa moze biti:
- sekvencijalni - slogovima ili blokovima a najcesce slogovima, automatski odrzavaju tj inkrementiraju vrednost tekuceg pokazivaca pri operacijama ucitavanja / zapisivanja podataka
- direktni - slogovima ili blokovima datoteke , zahtevaju eksplicitno zadavanje vrednosti tekuceg pokazivaca - rednog broja sloga ili bloka pri operacijama pozicioniranja
- dinamicki (kombinovani) - kombinacija direktnog i sekvencijalnog
Pozivi rutina metode pristupa:
- otvaranje i zatvaranje datoteke, ucitavanje i ispisivanje sadrzaja sloga ili bloka, pozicioniranje na slog ili blok datoteke, ispitivanje statusa datoteke, kreiranje i brisanje datoteke. Kod poslednjih preuzimaju parametre poziva iz aplikativnog programa kao sto su putanja i naziv datoteke, oznaka datoteke, oznaka promenljive u radnoj zoni programa.Takodje , prosledjuju u pozivajuce okruzenje informacije o statusu izvrsenja rutine metode pristupa

Upravljanje baferima metode pristupa

Okruzenje u kojem je implementirana metoda pristupa brine o zadacima upravljanja baferima kao sto su alociranje i dealociranje bafera, vodjenje evidencije o sadrzaju bafera i "dirty" bitu bloka. Imamo 3 nivoa "baferisanja" podataka datoteke u OM :
- nivo sistemskih bafera kojim upravlja OS
- nivo bafera metode pristupa kojim upravlja okruzenje u kojem je implementirana metoda pristupa

- nivo lokacija promenljivih u aplikativnom programu kojim upravlja aplikativni program

Okruzenja koja ukljucuju metode pristupa:

- operativni sistem
- programski jezik sa pridruzenim paketima funkcija
- sistem za upravljanje bazama podataka

Neki servisi metode pristupa mogu biti implementirani direktno u aplikativnom programu.

Operativni sistem

Najcesce stariji operativni sistemi "mainframe" racunara.Nije bila vidljiva eksplicitna podela na usluge OS niskog i visokog nivoa.Servisi metode pristupa su vidljivi kao monolitna struktura, a prvi SUBP nastaju na temelju eksplicitne upotrebe servisa takvih metoda pristupa. Podrzavali upravljanje blokovima i baferima metode pristupa - Block = Control Interval, dozvoljavaju eksplicitno deklarisanje formata bloka na nivou pojedinacne datoteke. Pozivi servisa metode pristupa cesto su kombinovani sa programskim jezikom Cobol ili PL1.

Programski jezik sa pridruzenim paketima funkcija

Prakticno svaki savremeni programski jezik pruza odredjene usluge metode pristupa koje mogu biti ugradjene u sam jezik ili ukljucene u odredjene pakete funkcija (upakovane i isporucene zajedno sa kompajlerom i razvojnim okruzenjem ili isporucene nezavisno od samog jezika).Eksplicitno koriste usluge niskog nivoa izabranog OS.Najcesce pruzaju samo usluge na nivou sloga datoteke.Upravljanje blokovima i baferima sakriveno je od aplikativnog programa.

Sistem za upravljanje bazama podataka

Svaki SUBP obavezno obezbedjuje usluge metode pristupa, najcesce se ne koriste direktno iz aplikativnog programa vec su na raspolaganju drugim modulima unutar SUBP.Eksplicitno koristi usluge niskog nivoa izabranog OS,mada je moguce u specificnim situacijama da SUBP zaobidje usluge niskog nivoa OS , tada on direktno upravlja fizickom razmenom podataka izmedju datoteke na disku i OM. Podrzava upravljanje blokovima i baferima metode pristupa (Database block), dozvoljava eksplicitno deklarisanje kapaciteta bloka/bafera na nivou instalacije SUBP (uniformno za sve datoteke kojima upravlja SUBP).

Parametri organizacije datoteka

Organizacija podataka je projekat logicke strukture obelezja, odnosno projekat i implementacija FSP u kontekstu isprojektovane LSO i sistemske arhitekture, sa ciljevima da se obezbede zadovoljenje korisnickih zahteva i uslovi za efikasnu obradu podataka.Rezultat organizovanja podataka je sistem baze podataka ili sistem datoteka.

Da bi projektovali i implementirali FSP moramo ispuniti odredjene zadatke:

- izbor nacina dodele lokacija slogovima
- izbor nacina memorisanja logickih veza izmedju slogova u LSP
- projektovanje osnovnih struktura podataka
- projektovanje pomocnih struktura podataka
- proracun i rezervisanje potrebnog prostora na eksternim memorijskim uredjajima
- smestanje slogova sa vezama na eksterne memorijske uredjaje
- proracun, pracenje i analiza performansi postupaka obrade podataka

Kod organizacije datoteke, projektovanje LSO svodi se na projektovanje tipa entiteta N(Q,C) tj. tipa sloga. Izbor vrste organizacije datoteke zavisi od vrednosti parametara :

- nacin dodele lokacija slogovima - uslovljava i nacin evidentiranja slobodnog i zauzetog prostora u datoteci
- nacin memorisanja logickih veza izmedju slogova u LSP

Nacin dodele lokacija slogovima(DLS)

Moguce vrednosti parametara DLS:

(A) svaki novi slog upisuje se na kraj datoteke , kao fizicki susedan u odnosu na poslednji slog datoteke - ako se prostor datoteke dinamicki alocira (povecava pri upisu) onda samo poslednji blok moze biti delimicno popunjen a svi ostali blokovi su kompletno popunjeni, a ako se prostor datoteke staticki alocira onda poslednji upisani slog datoteke deli prostor datoteke na kompletno zauzet i kompletno slobodni deo

(B) svaki novi slog dobija prvu slobodnu lokaciju iz spregnute linearne strukture slobodnih lokacija - prostor datoteke se uvek staticki alocira tj ne izaziva ga svaki

upis novih slogova u datoteku, a indeks na listu slobodnih lokacija memorise se u zaglavlju datoteke
(C) svaki novi slog dobija slobodnu lokaciju cija relativna adresa predstavlja funkciju vrednosti kljuca - prostor datoteke se uvek staticki alocira nezavisno od upisa novih slogova u datoteku.Ovaj nacin je moguc kada su u pitanju iskljucivo slogovi konstantne duzine.Funkcija transformacije vrednosti kljuca moze biti : hash(analiticka) transformacija ili tabelarno zadata uz upotrebu pomocne strukture.
Nacin memorisanja logickih veza(MLV) izmedju slogova u LSP
Moguce vrednosti parametara MLV:
(1) fizickim pozicioniranjem - logicki susedni slogovi se smestaju u fizicki susedne lokacije
(2) pomocu pokazivaca kao relativnih adresa - pokazivac memorise relativnu adresu logicki susednog sloga.

(2a) polja pokazivaca ugradjena u osnovnu strukturu - svaki slog osnovne strukture datoteke prosiruje se barem jednim poljem pokazivaca

(2b) polja pokazivaca ugradjena u pomocne strukture - uvodi se barem jedna pomocna tzv indeksna struktura , cesto oblika stabla sa formatom sloga u obliku para ili n-torke (polja identifikatora sloga, polja pokazivaca)
(3) logicke veze se ne memorisu - u FSP ne postoje podaci o logicki susednim slogovima, jedino se mogu generisati putem posebnih programa na zahtev korisnika - programi za uredjivanje datoteka.

Vrste organizacija datoteka

Vrste organizacija datoteka:
1. osnovne organizacije - organizacija datoteke svodi se na osnovnu organizaciju. FSP nad skupom slogova organizovana je u jednoj memorijskoj zoni , cesto je to i jedna datoteka operativnog sistema
2. slozene organizacije - dobijaju se kombinovanjem osnovnih organizacija.FSP ukljucuje barem 2 memorijske zone - mogu biti i barem 2 datoteke OS-a.Osnovna FSP moze biti rasporedjena u jednu ili vise od jedne zone.Mogu se pojaviti

pomocne strukture podataka smestene u posebnim zonama.
U osnovne organizacije spadaju :
- serijska, sekvencijalna, spregnuta, rasuta sa jedinstvenim memorijskim prostorom (direktna, relativna, staticka rasuta, dinamicka rasuta)
U slozene organizacije spadaju :
- rasute sa zonom prekoracenja, staticke indeksne (indeks-sekvencijalne), dinamicke indeksne (organizacije sa B-stablom)
Rasute sa zonom prekoracenja - primarna zona (osnovna struktura) - osnovna rasuta organizacija, zona prekoracenja(nastavak osnovne organizacije) - spregnuta ili serijska organizacija
Staticke indeksne - primarna zona(osnovna struktura) - sekvencijalna organizacija, zona prekoracenja(nastavak osnovne strukture) - spregnuta organizacija, zona indeksa (pomocna struktura) - spregnuta organizacija (sprezanje u obliku n-arnog stabla trazenja)
Dinamicke indeksne - primarna zona(osnovna struktura) - serijska ili spregnuta organizacija, zona indeksa(pomocna struktura) - spregnuta organizacija(sprezanje u obliku jedne od varijanti B stabla)
Navedene vrste organizacija javljaju se u praksi kao fizicke organizacije datoteka - u sistemima datoteka gde se svaka datoteka u sistemu javlja kao jedna ili vise posebnih OS datoteka, i kao fizicke organizacije tabela - u sistemima baza podataka gde svaka tabela BP moze biti distribuirana u vise datoteka podataka kojima upravlja SUBP, a u jednoj datoteci podataka kojom upravlja SUBP moze biti smesteno vise tabela BP

Opste procedure nad datotekama

Vrste postupaka (operacija) nad LSP datoteke:
- formiranje datoteke
- pristupanje u datoteci
- trazenje u datoteci
- pretrazivanje u datoteci
- obrada datoteka

- azuriranje datoteke
- reorganizacija datoteke

Formiranje datoteke je postupak kreiranja FSP datoteke sa smestanjem slogova na eksterni memorijski uredjaj saglasno projektovanoj organizaciji a na osnovu sadrzaja neke druge strukture podataka , preuzimanjem podataka iz nekih drugih datoteka ili direktnim zadavanjem podataka od strane korisnika. Dve vrste datoteka:

- datoteke koje se formiraju u posebnom postupku, najcesce - sekvencijalna, spregnuta, staticka rasuta, staticka indeksna
- datoteke koje se formiraju u redovnom postupku azuriranja, najcesce - serijska, indeksne sa B stablima, dinamicka rasuta

Pristupanje u datoteci je postupak pozicioniranja na zeljenu lokaciju sloga ili bloka datoteke. Vrste pristupa:

- sekvencijalni pristup - automatsko odrzavanje relativne adrese tekuceg pokazivaca.Operacija se odnosi na neposredno susednu lokaciju u odnosu na lokaciju na kojoj je obavljena prethodna operacija
- direktni pristup - eksplicitno zadavanje relativne adrese tekuceg pokazivaca koji ukazuje na lokaciju nad kojom ce se realizovati neka operacija
- dinamicki pristup - kombinacija sekvencijalnog i direktnog pristupa

NAPOMENA - sekvencijalni i direktni pristup i sekvencijalna i direktna organizacija nisu isto.

Trazenje u datoteci - algoritam : dom(K) -> Ind x A x S

To je postupak koji za zadatu vrednost argumenta trazenja (vrednost kljuca iz domena, a $\in$ dom(K)) generise i vraca po potrebi u program:

- indikaciju uspesnosti trazenja Ind = {true, false} - ako je slog nadjen trazenje je uspesno a ako slog nije pronadjen trazenje je neuspesno
- relativnu adresu mesta zaustavljanja trazenja - iz skupa svih adresa u adresnom prostoru datoteke A, koji po potrebi ukljucuje i adresu prve naredne lokacije nakon kraja datoteke
- sadrzaj sloga na mestu zaustavljanja trazenja - ili specijalnu vrednost ako je rec o nepostojecem slogu

Nisu uvek svi nabrojani izlazni parametri potrebni aplikativnom programu.Specificni algoritmi trazenja na izlazu ce generisati samo vrednosti onih

parametara koji su stvarno neophodni aplikativnom programu. Moguce svrhe primene algoritma trazenja:
- da bi se utvrdilo ima li trazenog sloga u datoteci ili nema (npr radi provere ispunjenosti uslova za upis novog ili brisanje postojeceg sloga sa zadatom vrednoscu kljuca)
- da bi se utvrdila adresa na kojoj se trazeni slog nalazi (jer je potrebno direktno pozicioniranje na datu adresu)
- da bi se preneo sadrzaj trazenog sloga u aplikativni program (npr. radi daljih potreba obrade podataka)

```
POCETAK TRAZENJA
  generisanje pocetne relativne adrese trazenja
  postoji potreba za nastavak trazenja  <- DA
  RADI petlja_trazenja DOK postoji potreba za nastavak trazenja
        citanje sadrzaja sloga s tekuce adrese
        AKO argument trazenja = vrednost kljuca tekuceg sloga TADA
          trazenje uspesno, postoji potreba za nastavak trazenja  <- NE
        INACE
          AKO postoje uslovi za nastavak trazenja TADA
            generisanje naredne relativne adrese trazenja
          INACE
            trazenje neuspesno, postoji potreba za nastavak trazenja  <- NE
          KRAJ AKO
        KRAJ AKO
    KRAJ RADI petlja_trazenja
KRAJ TRAZENJA
```

Metode trazenja s obzirom na vrstu postupka:
- linearno trazenje - moguce u sekvencijalnim, serijskim i rasutim organizacijama
- binarno trazenje - iskljucivo moguce u sekvencijalnim organizacijama
- trazenje pracenjem pokazivaca - iskljucivo moguce u spregnutim organizacijama (u osnovnim strukturama i uz koriscenje pomocnih struktura)
- trazenje metodom transformacije argumenta u adresu - h: dom(K) -> A, moguce u rasutim organizacijama

Vrste trazenja s obzirom na predistoriju trazenja:
- trazenje slucajno odabranog sloga (tso) - izbor pocetne adrese trazenja je unutrasnje pitanje algoritma i ne zavisi od mesta zaustavljanja prethodnog trazenja, niti od toga da li je postojalo prethodno trazenje.Moguce je u svim organizacijama datoteka
- trazenje logicki narednog sloga (tln) - pocetna adresa trazenja predstavlja adresu na kojoj je zaustavljeno prethodno trazenje, a moguce je ako je prethodno postojalo barem jedno trazenje.Svaka naredna adresa trazenja moze biti samo adresa logicki narednog sloga.Moguce je u organizacijama u kojima se vode podaci o logicki narednom slogu.

Pretrazivanje u datoteci - algoritam dom(LogUslov) -> P(S) ili P(A)
To je postupak koji za zadatu vrednost argumenta pretrazivanja (zadati logicki uslov) generise i vraca po potrebi u program skup slogova koji zadovoljavaju logicki uslov pretrazivanja (P(S) - partitivni skup skupa slogova S) ili skup adresa slogova koji zadovoljavaju logicki uslov pretrazivanja (P(A) - partitivni skup skupa adresa iz adresnog prostora).Pretrazivanje je uspesno ako je skup slogova koji zadovoljava zadati uslov neprazan.Logicki uslov zahteva definisanje sintakse zapisivanja.Predstavlja logicki izraz gde je moguca upotreba logickih izraza i operanada, relacionih izraza i operanada, tipskih izraza i operanada(aritmetickih, alfanumerickih, datumskih) a u ulozi operanda mogu se pojaviti atributi tipa sloga, konstante i funkcije primenjene nad izrazima.Neki specijalni tipovi logickih uslova su uslovi konjuktivnog tipa i uslovi disjunktivnog tipa.
Sekundarni kljuc - niz obelezja strukture po kojem se vrsi pretrazivanje.

Azuriranje datoteke je postupak dovodjenja LSP datoteke u sklad sa izmenjenim stanjem klase entiteta u realnom sistemu.Osnovne operacije:
- upis novog sloga u datoteku - zahteva prethodno neuspesno trazenje i moze iziskivati premestanje odredjenog broja drugih slogova
- modifikacija vrednosti neprimarnih atributa sloga - zahteva prethodno uspesno trazenje i uobicajeno se zabranjuje modifikacija vrednosti obelezja primarnog kljuca po kojem je uspostavljena osnovna organizacija.
- brisanje postojeceg sloga iz datoteke - zahteva prethodno uspesno trazenje i moze iziskivati premestanje odredjenog broja drugih slogova

Vrste brisanja:
1. logicko brisanje sloga iz datoteke - svodi se na izmenu vrednosti polja statusa sloga iz statusa aktuelnog u status neaktuelnog.Neaktuelni slog i dalje zauzima lokaciju u memorijskom prostoru. Lokacije neaktuelnih slogova oslobadjaju se reorganizacijom.
2. fizicko brisanje sloga iz datoteke - dovodi do izmene sadrzaja bloka u kojem se nalazio izbrisani blok.Moze izazvati pomeranje drugih slogova iz jednih u druge lokacije.Dovodi do oslobadjanja jedne lokacije sloga u memorijskom prostoru pri cemu to ne mora obavezno biti lokacija izbrisanog sloga.
Obrada datoteka je algoritamski iskazani niz operacija nad LSP jedne ili vise datoteka sa ciljem svrsishodne transformacije podataka datoteka.Moguca je primena operacija:
- pristupa slogovima (jedina obavezna vrsta operacija u obradi datoteka), trazenja i pretrazivanja, azuriranja (unosa,brisanja,modifikacije), generisanja (izracunavanja) novih podataka

Uloge datoteka u obradi

Podela prema vrsti primenjenih operacija u obradi:
- ulazna datoteka - datoteka u kojoj se iskljucivo vrse citanja
- izlazna datoteka - datoteka u koju se iskljucivo zapisuju novi slogovi u obradi
- ulazno - izlazna datoteka - datoteka u kojoj se vrse i citanja i azuriranja slogova

Podela prema ulozi u trazenjima slogova:
- vodeca datoteka - datoteka koja iskljucivo generise argumente trazenja ili pretrazivanja slogova tokom obrade.Barem jedna ulazna datoteka u obradi mora biti vodeca.
- obradjivana datoteka - datoteka u kojoj se iskljucivo vrse trazenja ili pretrazivanja na osnovu generisanih argumenata
- vodeca i obradjivana datoteka - datoteka sa obe uloge, vodeca za neku drugu obradjivanu i obradjivana u odnosu na neku drugu vodecu.

Vrste obrade prema nacinima trazenja slogova u obradjivanoj datoteci:
- direktna obrada - u svakom narednom koraku obrade zahteva se trazenje slucajno odabranog sloga
- redosledna (sekvencijalna) obrada - u svakom narednom koraku obrade zahteva se trazenje logicki narednog sloga i/ili sekvencijalni pristup fizicki susednoj lokaciji

Reorganizacija datoteke obuhvata ponovno formiranje datoteke u cilju dovodjenja u sklad FSP sa novim stanjem LSP.Do reorganizacije dolazi zato sto operacije azuriranja vrse izmene u LSP koje FSP nekada ne prati na odgovarajuci nacin sto dovodi do degradacije performansi rada sa datotekom. Primeri:
- nagomilavanje neaktuelnih , logicki izbrisanih slogova koji zauzimaju lokacije u FSP
- nagomilavanje lanaca slogova
- neizbalansiranost podataka s obzirom na postojecu indeksnu strukturu
- prevelika fragmentacija slobodnog prostora

Organizacije koje traze povremenu reorganizaciju:
- sekvencijalna
- spregnuta
- staticka rasuta
- staticka indeksna

Organizacije koje ne traze povremenu reorganizaciju:
- serijska - reorganizacija nije neophodna
- indeksna sa B stablom - reorganizacija se sprovodi dinamicki i lokalizovana je
- dinamicka rasuta - reorganizacija se sprovodi dinamicki i lokalizovana je

Performanse obrade datoteke

Idealna organizacija datoteke:
- zahteva tacno onoliko lokacija koliko sadrzi slogova - faktor popunjenosti 100%
- zahteva najvise jedan pristup za tso i tln
- zahteva najvise jedan pristup za pretrazivanje - po bilo kom zadatom uslovu
- zahteva jedan pristup za bilo koju operaciju azuriranja
- nikada ne zahteva reorganizaciju

Izbor vrste organizacije datoteke predstavlja kompromisno resenje.Nemoguce je da jedna vrsta organizacije zadovolji sve navedene zahteve.Favorizacija jednih cesto defavorizuje druge zahteve.Uzimaju se u obzir potrebe i uticajnost aplikativnih programa.Favorizuju se zeljene mere performansi u odnosu na zauzece memorijskog prostora.Cena memorisanja po jedinici kapaciteta je sve

niza.
Ukupno vreme trazenja ili pretrazivanja slogova zavisi od:
- broja i vremena pristupa blokovima na jedinici diska - broj je dominantno opredeljen vrstom organizacije datoteke (u slucaju pretrazivanja i prirodom logickog uslova) a za velike datoteke zavisi od karakteristika fajl sistema OS.Vreme je dominantno opredeljeno karakteristikama diska (~ 10ms).
- vremena prenosa bloka sa diska u OM - dominantno opredeljeno karakteristikama diska i spreznog podsistema (< 1 ms).
- broja i vremena uporedjivanja argumenta sa vrednoscu kljuca - broj je dominantno opredeljen vrstom organizacije datoteke (u slucaju pretrazivanja i prirodom logickog uslova).Vreme je dominantno opredeljeno karakteristikama OM i CPU (~ 10ns)
Mere za ocenu performansi:
(A) - broj pristupa blokovima, broj uporedjivanja argumenta i vrednosti kljuca
(B) - srednji broj, broj najgorem slucaju (apsolutni broj)
(C) - trazenje logicki narednog sloga, trazenje slucajno odabranog sloga, operacije azuriranja (upis, brisanje, modifikacija)
(D) - uspesna operacija, neuspesna operacija

10.PREZENTACIJA

Serijska organizacija datoteke

Osnovna struktura:
Slogovi su smesteni jedan za drugim u sukcesivne memorijske lokacije.Fizicka struktura ne sadrzi informacije o vezama izmedju slogova logicke strukture datoteke.Ne postoji veza izmedju vrednosti kljuca sloga i adrese lokacije u koju je smesten.Redosled memorisanja slogova najcesce prema hronoloskom redosledu njihovog nastanka.Slogovi mogu a ne moraju biti blokirani.

Formiranje serijske datoteke

Serijska datoteka se generise najcesce u postupku obuhvata podataka.Slogovi se

formiraju prenosom podataka sa razlicitih izvora - izvorna dokumenta, uredjaji i softveri za ocitavanje vrednosti.Upisuju se jedan za drugim u sukcesivne memorijske lokacije a svaki novi slog se upisuje na kraj datoteke.Rezultat obuhvata podataka je neblokirana ili blokirana serijska datoteka.
Obuhvat podataka je proces sa zadatkom da obezbedi inicijalno memorisanje ispravnih podataka.Osnovna aktivnost je upis podataka na medijum a izvrsava je ili covek-operater koristeci program sa odgovarajucim UI-jem zaa formatiranje podataka i odgovarajuce U/I uredjaje, ili specijalizovani softver sa odgovarajucim hardverskim uredjajima.
UI programa za obuhvat podataka definise:
1. opis formata dokumenta - raspored polja
2. pravila navigacije (pomeranje kursora izmedju polja) - daje mogucnost pristupa poljima putem misa ili tastature i redosled obilaska polja u navigaciji putem tipke <TAB> , <ENTER> ili automatskoj
3. opisi i formatiranje sadrzaja polja - definisu:
    - fiksni tekst i tekstualno uputstvo sa opisom polja
    - tip polja (alfa, numericko, alfanumericko, datumsko)
    - nacin vizuelizacije sadrzaja - tekstualno polje, padajuca lista, combo-box,radio-grupa, polje skrivenog sadrzaja, format maska za numericke ili datumske vrednosti, zadavanje vizuelnih atributa polja na formi
    - maksimalni broj znakova koji je moguce uneti
    - nacin poravnavanja sadrzaja polja (levo,desno,centrirano)
    - nacin preloma sadrzaja tekstualnih polja (bez preloma, prelom na 1 znak,prelom na celu rec)
4. specijalne kontrole sadrzaja polja - podrazumevaju :
    - obaveznost unosenja barem jednog znaka u polje
    - obaveznost kompletnog popunjavanja sadrzaja polja
    - kontrola na dozvoljeni opseg vrednosti
    - kontrola po modulu - samo za numericka polja (broju od n cifara pridruzuje se jedna kontrolna cifra koja se izracunava primenom posebnog algoritma na broj od n cifara)
    - provera da li se uneti podatak nalazi u tabeli / listi dozvoljenih vrednosti

5. dozvoljene operacije nad sadrzajima polja :

- rucni unos sadrzaja polja
- modifikacija postojeceg sadrzaja polja
- brisanje sadrzaja jednog, izabranih, ili svih polja
- dupliciranje sadrzaja jednog, izabranih ili svih polja u cilju povecava produktivnost operatora ili kada se isti ili skoro isti sadrzaj polja vise puta ponavlja

Vreme obavljanja obuhvata podataka moze biti:

- u realnom vremenu - na mestu i trenutku nastanka podataka
- u odlozenom rezimu - nakon odredjenog intervala vremena od nastanka podataka, po pravilu ga realizuje operater koji nije evidentirao izvorne podatke na osnovu manuelno izradjenih dokumenata.Na kraju imamo verifikaciju.

Verifikacija je postupak sustinske provere ispravnosti unetih podataka - npr. drugi operater ponovo unosi jednom vec unete podatke koristeci isti izvorni dokument, koriscenje inteligentnih softverskih resenja.

Trazenje sloga u serijskoj datoteci

Imamo trazenje slucajno odabranog sloga jer ne postoji funkcionalna veza izmedju vrednosti kljuca i adrese lokacije sloga pa je trazenje logicki narednog isto sto i trazenje slucajno odabranog.Primena metode linearnog trazenja pocinje od pocetka datoteke i zahteva pristupanje sukcesivno memorisanim blokovima i slogovima.

Ukupan broj pristupa kod uspesnog trazenja Ru je $1 <= Ru <= B$

Ukupan broj pristupa kod neuspesnog trazenja Rn je B , B je ukupan broj blokova datoteke

Ukupan broj blokova datoteke racuna se kao $B = \lceil (N + 1) / f \rceil$ gde je N - broj slogova, f - faktor blokiranja a +1 imamo zbog specijalnog sloga sa oznakom kraja datoteke.

Ocekivani (srednji) broj pristupa kod uspesnog trazenja Ru (nadvuceno) = $B/N * (N - f*(B-1)/2)$, a kada $f|N$ tj $B = N/f + 1$ ili kada je $N>>1$ onda je Ru (nadvuceno) = $B/2$.

Ukupan broj uporedjivanja argumenta trazenja i vrednosti kljuca kod uspesnog trazenja Uu je $1 <= Uu <= N$ a ocekivani broj je Uu (nadvuceno) = $N+1 / 2$. Kod neuspesnog trazenja ukupan broj ovih uporedjivanja je $Un = N$.

Obrada serijske datoteke

Vrste :

- direktna
- redosledna - ukoliko se ide na sekvencijalni pristup slogovima u hronoloskom redosledu

Moze se koristiti kao vodeca u rezimu direktne obrade.Moze se koristiti kao vodeca u rezimu redosledne obrade datoteke ciji kljuc sadrzi kao svoj strani kljuc i kada je uredjena saglasno neopadajucim vrednostima tog stranog kljuca.

Program koji vrsi redoslednu obradu serijske datoteke ucitava sukcesivne slogove vodece datoteke.Svaki naredni slog vodece datoteke sadrzi logicki narednu vrednost kljuca obradjivane serijske datoteke.Te vrednosti kljuca se koriste kao argumenti za trazenje u serijskoj datoteci metodom linearnog trazenja.

U rezimu direktne obrade sukcesivni slogovi vodece datoteke sadrze slucajno odabrane vrednosti kljuca obradjivane serijske datoteke a trazenje je ponovo linerano.Trazenje logicki narednog i slucajno odabranog sloga serijske datoteke se obavlja identicno krecuci od prvog sloga datoteke.

Vodeca datoteka generise Nv argumenata trazenja za serijsku datoteku od kojih Nv na u slogova inicira uspesna trazenja a Nv na n neuspesna trazenja, pa je Nv = Nv na u + Nv na n. Ukupan prosecni broj trazenja je Ruk (nadvuceno) = Nv na u * Ru (nadvuceno) + Nv na n * Rn (nadvuceno). Ovo mozemo pisati i kao :

Ruk (nadvuceno) priblizno = Nv na u * B/2 + Nv na n * B. Broj pristupa se ne razlikuje za slucaj direktne i redosledne obrade.

Azuriranje serijske datoteke

Upis novog sloga u prvu slobodnu lokaciju na kraju datoteke, mora mu prethoditi jedno neuspesno trazenje.Jednostavan je ali zahteva veliki broj pristupa:

Ri = Rn + 1 kada ne vazi da f | (N + 1) a Rn + 2 kada to vazi - svaki f-ti put je neophodno prosiriti datoteku novim blokom. Ocekivani broj pristupa je Ri (nadvuceno) = Rn + 1 + 1/f

Brisanje postojeceg sloga - mora mu prethoditi jedno uspesno trazenje.Najcesce samo logicko izmenom statusa aktuelnosti sloga jer bi fizicko brisanje zahtevalo veliki broj pristupa.

Modifikacija sadrzaja postojeceg sloga - mora mu prethoditi jedno uspesno trazenje. Ocekivani broj pristupa za logicko brisanje ili modifikaciju sadrzaja sloga

je Rd (nadvuceno) = Ru (nadvuceno) + 1.

Oblasti primene i ocena karakteristike:

Pogodne su kao male datoteke kada mogu cele stati u OM zbog veoma velikog broja pristupa potrebnog za pronalazenje logicki narednog ili slucajno odabranog sloga.Druge vrste organizacije donose samo mala poboljsanja u efikasnosti obrade malih datoteka. Serijska organizacija podataka u kombinaciji sa indeksnim strukturama je veoma pogodna za direktnu obradu i to je onda osnovna fizicka struktura relacionih baza podataka.Serijska datoteka kao rezultat obuhvata podataka je polazna osnova za izgradnju datoteka sa drugim vrstama organizacije podataka.

Sekvencijalna organizacija datoteke

Osnovna struktura:

Slogovi su smesteni sukcesivno jedan za drugim.Logicki susedni slogovi smestaju se u fizicki susedne lokacije. Postoji informacija o vezama izmedju slogova logicke strukture podataka datoteke i ugradjena je u fizicku strukturu.Realizovana je kao linearna logicka struktura podataka smestanjem sloga sa vecom vrednoscu kljuca u lokaciju sa vecom adresom.Rastuce uredjenje po vrednostima kljuca => slog sa najmanjom vrednoscu kljuca smesta se u prvu lokaciju.

Naziva se i fizicki sekvencijalnom organizacijom.

Veza izmedju memorisanih vrednosti kljuca k(S) i adresa lokacija nije ugradjena u strukturu datoteke i ne predstavlja bilo kakvu matematicku funkciju.Slogovi se smestaju u blokovima od po f(>=1) slogova, pozeljno je da faktor blokiranja bude sto veci.Savremeni OS i programski jezici podrzavaju samo sekvencijalni nacin pristupa pa je korisnicima ostavljeno da naprave svoje sopstvene sekvencijalne metode pristupa.

Formiranje sekvencijalne datoteke - najcesce sortiranjem serijske datoteke saglasno rastucim ili opadajucim vrednostima kljuca.

Trazenje sloga u sekvencijalnoj datoteci - logicki narednog ili slucajno odabranog Trazenje slucajno odabranog sloga - moguca je primena metoda linearnog trazenja, binarnog trazenja i trazenja po m putanja.Nema prakticnog smisla ako je

datoteka velika i smestena na eksterni memorijski uredjaj.Ima prakticnog smisla ako je datoteka cela smestena u OM.U tom slucaju moze je predstavljati neka linearna struktura nad skupom slogova ili blok neke druge datoteke npr. indeks-sekvencijalne

Trazenje logicki narednog sloga - linearnom metodom trazenja pocevsi od prvog, fizicki susedni blokovi se ucitavaju u OM. U centralnoj jedinici se vrsi uporedjivanje argumenata trazenja i vrednosti kljuca sukcesivnih slogova dok se trazeni slog ne pronadje, argument trazenja ne postane manji od vrednosti kljuca sloga ili ne dodje do kraja datoteke.Trazenje novog logicki narednog sloga zapocinje od sloga na kojem se prethodno trazenje zaustavilo (tekuceg sloga datoteke).

Broj pristupa pri uspesnom i neuspesnom trazenju je $0 <= R <= B - i$, i - redni broj tekuceg bloka u odnosu na pocetak. Broj poredjenja argumenata trazenja i vrednosti kljuceva slogova pri uspesnom i neuspesnom trazenju je $1 <= U <= N - i + 1$ gde je i - redni broj tekuceg sloga (broj za 1 veci od duzine puta izmedju prvog i tekuceg sloga)

Obrada sekvencijalne datoteke moze biti redosledna i direktna.

Direktna obrada ima smisla ako je sekvencijalna datoteka mala, tako da se moze smestiti u OM.Performanse obrade se malo razlikuju od performansi obrade serijske datoteke. Ruk (nadvuceno) = Nv na u * Ru (nadvuceno) + Nv na n * Rn (nadvuceno) odnosno moze se pisati Ruk (nadvuceno) priblizno = Nv na u * B/2 + Nv na n * B/2

Vodeca datoteka u direktnoj i redoslednoj obradi ima cestu upotrebu.Sukcesivno ucitava fizicki susedne slogove pocevsi od prvog pa do poslednjeg.Ukupan broj pristupa kada se sekvencijalna datoteka koristi kao vodeca u obradi :

$Ruk = B = \lceil (N + 1) / f \rceil$

Redosledna obrada je iterativan proces gde vodeca datoteka generise logicki naredne vrednosti kljuca za trazenje u obradjivanoj sekvencijalnoj datoteci.Svaki korak obrade je ustvari trazenje logicki narednog sloga i vrsi se metodom linearnog trazenja.Svaki blok datoteke ucitava se u OM samo jednom.Vodeca datoteka sadrzi Nv ($Nv >= 1$) slogova i ukljucuje vrednost kljuca vecu ili jednaku najvecoj vrednosti kljuca u obradjivanoj datoteci.

Ukupan broj pristupa Ruk = B, srednji broj pristupa po 1 trazenju R (nadvuceno) =

B / Nv, sto je R (nadvuceno) manji, obrada je efikasnija.Pozeljniji je veci faktor blokiranja f i veci broj trazenja Nv.
Ukupan broj uporedjivanja U >= N + Nv a srednji broj uporedjivanja je U (nadvuceno) >= N/Nv + 1

Azuriranje sekvencijalne datoteke

Upis novog sloga - zahteva jedno neuspesno trazenje.Trazi se mesto upisa novog sloga - lokacija sloga sa prvom vecom vrednoscu kljuca od datog.Svi slogovi sa vrednostima kljuca vecim od vrednosti kljuca novog sloga pomeraju se za jednu lokaciju udesno.
Brisanje postojeceg sloga - treba nam prethodno pronalazenje sloga odnosno uspesno trazenje.Svi slogovi sa vecom vrednoscu kljuca pomeraju se za jednu lokaciju ulevo ako se vrsi fizicko brisanje.
Modifikacija sadrzaja sloga - treba nam prethodno pronalazenje sloga odnosno uspesno trazenje.
Kod upisa i brisanja imamo ozbiljan problem ukupnog broja pristupa.
U rezimu direktne obrade - u proseku pomera se polovina od ukupnog broja slogova za jednu lokaciju udesno pri upisu i ulevo pri brisanju sloga.Primenjuje se kada je kompletna datoteka smestena u OM.
U rezimu redosledne obrade imamo poseban iterativni postupak - kreiranje potpuno nove datoteke na osnovu postojece.Primeren je kada se datoteka ne moze kompletno smestiti u OM.Datoteke i uloge u obradi:
- Ds - obradjivana, ulazna(stara) sekvencijalna datoteka
- Dn - obradjena, izlazna (nova) sekvencijalna datoteka
- Dp - vodeca datoteka promena, serijska, ulazna
- Dg - datoteka gresaka, izlazna

Format sloga datoteka Ds i Dn je identican (k(Si), p(Si)).
Format sloga datoteke promena Dp : (k(Si), Pp(Si),Sp(Si)) gde je Sp(Si) - polje statusa izvrsene operacije sa mogucim vrednostima : n - novi slog, m - podaci za modifikaciju, b - slog za brisanje.
Format sloga datoteke gresaka Dg : (k(Si), p(Si), Sg(Si)) gde je Sg(Si) - polje opisa greske a moguce vrednosti ukazuju na : pokusaj upisa vec postojeceg sloga u datoteku ili pokusaj brisanja ili modifikacije nepostojeceg sloga datoteke.

Postupak:
Imamo sekvencijalni pristup sa ucitavanjem slogova Ss(Ds) i Sp(Dp) zatim uporedjivanje vrednosti kljuceva tekucih slogova pa onda generisanje novih slogova Sn(Dn) na osnovu sadrzaja tekucih slogova Ss i Sp i na kraju upis slogova Sn u datoteku Dn.
Duzina intervala izmedju dva azuriranja odredjuje se tako da se tokom njega nakupi toliki broj promena koji bi opravdao pristupanje svim slogovima stare i generisanje nove datoteke.Duzi interval => veca efikasnost obrade, ali i duze vreme neusaglasenosti sadrzaja datoteke sa realnim stanjem.
Datoteka promena Dp sadrzi Nv = Nv na n + Nv na b + Nv na m slogova (Nv na n za upis, Nv na b za brisanje i Nv na m za modifikaciju), i Bv blokova, gde je Bv = ⌈(Nv + 1) / f ⌉.
Postojeca datoteka Ds sadrzi Bs blokova : Bs = ⌈(N + 1) / f ⌉
Nova datoteka Dn sadrzi Bn blokova : Bn = ⌈(N + Nv na n - Nv na b + 1) / f ⌉
Srednji broj pristupa pri azuriranju datoteke za jedno trazenje logicki narednog sloga je R (nadvuceno) = (Bv + Bs + Bn) / Nv

Oblasti primene i ocena karakteristika

Prednosti:
1. najpogodnija fizicka organizacija za redoslednu obradu
- rezim redosledne obrade se cesto koristi u praksi u paketnoj obradi podataka.To je posledica cinjenice da su logicki susedni slogovi smesteni u fizicki susedne lokacije.Ucitavanjem jednog bloka u OM, pribavlja se f slogova koji najverovatnije ucestvuju u narednim koracima obrade.Pozeljno je da f bude sto veci.Kada Nv -> N , tada R (nadvuceno) -> 1/f , pa se s povecanjem f poboljsava efikasnost obrade.
2. ekonomicno koriscenje memorijskog prostora
3. mogucnost koriscenja i magnetne trake i magnetnog diska kao medijuma

Nedostaci:
1. nepogodnost za direktnu obradu
2. potreba sortiranja pri formiranju
3. relativno dugotrajan postupak azuriranja

11.PREZENTACIJA

Rasute organizacije datoteka

Rasuta organizacija datoteke se cesto u najsirem znacenju naziva i direktnom jer se slogu ili grupi slogova pristupa direktno na osnovu poznavanja adrese memorijske lokacije u koju su smesteni.Adresa lokacije dobija se transformacijom vrednosti identifikatora sloga u adresu.Identifikator je skup obelezja cije vrednosti jednoznacno odredjuju slogove datoteke.On moze a ne mora pripadati skupu obelezja tipa sloga datoteke, pa moze biti interni ili eksterni. Interni identifikator je po pravilu primarni kljuc datoteke a kod eksternog identifikatora, vrednosti identifikatora pridruzuju se svakom slogu eksterno van konteksta sadrzaja datoteke.
Transformacija vrednosti identifikatora u adresu je funkcija koja svakom identifikatoru iz domena pridruzuje jednu adresu iz skupa adresa lokacija memorijskog prostora datoteke. h : dom(K) -> A
Vrste transformacija vrednosti identifikatora u adresu:
1. deterministicka - funkcija h je injektivna, svakoj vrednosti identifikatora odgovara jedna adresa a svakoj adresi odgovara najvise jedna vrednost identifikatora.
2. probabilisticka - svakoj vrednosti identifikatora odgovara jedna adresa a jednoj adresi moze odgovarati vise rezultata transformacije.Ovo je metoda za generisanje pseudoslucajnih brojeva.
FSP ne sadrzi informaciju o vezama izmedju slogova logicke strukture datoteke.U dve fizicki susedne lokacije mogu a ne moraju biti memorisani logicki susedni slogovi.Slogovi su na slucajan nacin rasuti po memorijskom prostoru datoteke.
Baket je tradicionalni naziv za blok kod rasutih datoteka.Faktor blokiranja sada zovemo faktor baketiranja (b >= 1). Transformacijom h vrednost identifikatora pretvara se u adresu baketa.
Vrste rasutih datoteka s obzirom na nacin alokacije memorijskog prostora:
1. staticke - velicina adresnog prostora odredjuje se i kompletno alocira

unapred,staticki i ne moze se menjati tokom eksploatacije.
2. dinamicke - velicina dodeljenog adresnog prostora menja se tokom azuriranja, saglasno potrebama
Istorijski gledano, staticke rasute datoteke nastale su znatno ranije od dinamickih.

Opsti postupak formiranja staticke rasute datoteke

Statickoj rasutoj datoteci se u postupku formiranja dodeljuje Q = q*b lokacija, N<=Q , nakon formiranja se Q vise ne moze menjati. Uvodimo pojam faktora popunjenosti - q = N / Q , 0 < q <= 1
Redosled smestanja slogova u datoteku je nevazan i u slucaju deterministicke i probabilisticke transformacije. Slogovi se upisuju saglasno hronoloskom redosledu nastanka.Upisu sloga prethodi neuspesno trazenje na osnovu obavljene transformacije identifikatora u adresu.Slog se smesta u baket sa izracunatom adresom.

Direktna i relativna organizacija datoteke

Direktna organizacija datoteke:
Imamo eksterni identifikator i trivijalnu deterministicku transformaciju h : A -> A. Vrednost identifikatora je u isto vreme i adresa baketa : Ai = Ki - najjednostavniji vid deterministicke transformacije. Preslikavanje veza izmedju sadrzaja slogova i adresa ne pripada direktnoj organizaciji datoteke. Vrste direktnih datoteka s obzirom na vrstu upotrebljavanih adresa:
- vrednosti identifikatora su masinske adrese baketa
- vrednosti identifikatora su relativne adrese baketa

Direktna datoteka sa masinskim adresama - direktna datoteka u uzem smislu reci Koristi se adresa baketa na disku , oblika (u,c,t,s).Ima samo istorijski znacaj.
Nedostaci:
- cvrsta povezanost programa sa fizickim karakteristikama memorijskog uredjaja.Program zavisi od karakteristika eksternog memorijskog uredjaja i fizicki alociranog prostora na uredjaju.Ne moze se koristiti putem programskih jezika trece generacije.
- odsustvo logicke veze izmedju vrednosti identifikatora i sadrzaja

sloga.Odrzavanje veze izmedju sadrzaja sloga i adrese je zadatak krajnjeg korisnika ili u boljem slucaju samog programa.Problemi u slucaju reorganizacije ili brisanja pa upisa sloga.

Direktna datoteka sa relativnim adresama - relativna organizacija datoteke
Koriste se relativne adrese slogova.Lokacije alociranog memorijskog prostora se numerisu rednim brojevima od 1 do Q (alternativno od 0 do Q-1). Redni broj sloga u memorijskom prostoru predstavlja eksterni identifikator sloga.Resava se problem cvrste povezanosti slogova datoteke sa fizickim karakteristikama memorijskog uredjaja.
Glavni nedostatak je odsustvo logicke veze izmedju vrednosti identifikatora i sadrzaja sloga.
Imamo **relativnu metodu pristupa** - deo sistema za upravljanje podacima koji obavlja transformaciju relativne adrese u masinsku i ostale operacije.Ova metoda pristupa pruza programu usluge na nivou bloka.Ne vrsi blokiranje i rastavljanje blokova na slogove, prihvata samo vrednost faktora blokiranja jednaku 1. Sa stanovista relativne metode pristupa je **1 blok = 1 slog**. Aktivnosti blokiranja i rastavljanja blokova na slogove moraju se realizovati u okviru samog programa ukoliko za tim postoji realna potreba cime se moze postici veca efikasnost iskoriscenja memorijskog prostora.
Ova metoda pristupa se javlja kod :
- sistema za upravljanje datotekama mainframe racunarskih sistema - kompletna podrska - usluge na nivou bloka.
- savremenih SUBP - moguca podrska ali nije tipicno u upotrebi.
- standardnih biblioteka programskih jezika - podrzana transformacija relativnih adresa u apsolutne, moguca kompletna podrska - usluga na nivou sloga ili podrska samo direktnog pristupa lokacijama datoteke (rutinske operacije rada sa slogovima datoteke resavaju se kroz aplikativni program, koriste se pozivi rutina koji obezbedjuju rad sa datotekom kao bajt ili znak - orijentisanom strukturom).
One za razliku od sistema za upravljanje datotekama mainframe racunarskih sistema ne prave razliku izmedju serijske,sekvencijalne, relativne, spregnute ili rasute osnovne organizacije datoteke.Pruzaju usluge na nivou sloga ili cak samo na nivou bajt-orijentisane strukture.Za svaku datoteku podrzavaju metode direktnog i sekvencijalnog pristupa.Za upravljanje svakom datotekom podrzavaju

iste sistemske pozive i odgovarajuce bibliotecne funkcije (otvaranje,pozicioniranje,citanje,upisivanje i zatvaranje).Aplikativnom programu ostaje da obezbedi svu funkcionalnost koju zahteva odredjena vrsta organizacije.
Formiranje relativne datoteke najcesce u posebnom postupku na osnovu vodece serijske ili neke druge vrste datoteke - sukcesivno ucitavanje slogova vodece datoteke, pridruzivanje vrednosti identifikatora - relativne adrese slogu (najcesce automatski kroz aplikativni program), smestanje sloga u lokaciju s pridruzenom relativnom adresom.
Trazenje sloga u relativnoj datoteci - trazenje i logicki narednog i slucajno odabranog sloga obavlja se zadavanjem relativne adrese lokacije metodi pristupa (primena trivijalne metode transformacije identifikatora u adresu).Na osnovu adrese, metoda pristupa prenosi blok u radnu zonu programa. Uspesno trazenje imamo ako i samo ako trazeni slog postoji u prenetom bloku.Broj pristupa za uspesno ili neuspesno trazenje je Ru = 1 i Rn = 1.
Azuriranje relativne datoteke u rezimu direktne obrade:
Upis novog sloga - novom slogu se pridruzuje vrednost identifikatora. Slog se upisuje u datoteku ako postoji slobodna lokacija u bloku.Pre upisa, moze se izvrsiti provera da li slog sa istom vrednoscu kljuca vec postoji u istom bloku.
Brisanje - realizuje se kao logicko.Procita se sadrzaj adresiranog bloka, nakon provere vrednosti kljuca menja se sadrzaj statusnog polja sloga.Modifikovani sadrzaj bloka se ponovo upisuje u datoteku.
Ocena karakteristika relativne datoteke:
Trazenje slucajno odabranog sloga je najefikasnije moguce uz pretpostavku da korisnik brine o vezi izmedju sadrzaja sloga i relativne adrese.
Trazenje logicki narednog sloga je znacajno efikasnije nego u slucaju serijske datoteke a znacajno manje efikasno nego u slucaju sekvencijalne datoteke.
Za oba ova trazenja je potreban uvek samo jedan pristup datoteci.
Uvodjenje relativne adrese lokacije kao identifikatora resava problem cvrste povezanosti slogova datoteke sa karakteristikama memorijskog uredjaja.
Nepostojanje veza izmedju vrednosti identifikatora i sadrzaja sloga u relativnoj organizaciji datoteke je prepreka za sire koriscenje relativnih datoteka u praksi.
Relativna metoda pristupa se moze primeniti kao:

- osnov za izgradnju spregnute datoteke (relativna adresa lokacije logicki

narednog sloga smesta se u polje pokazivaca tekuceg sloga)

- osnov za izgradnju rasutih datoteka sa probabilistickom transformacijom kljuca u adresu (probabilisticka transformacija brine o vezi izmedju vrednosti identifikatora i relativne adrese)

- osnov za izgradnju indeksnih datoteka (formira se pomocna struktura podataka koja obezbedjuje memorisanje veze izmedju sadrzaja sloga i relativne adrese)

Staticka rasuta organizacija datoteke

Opste karakteristike rasute datoteke sa probabilistickom transformacijom:
Kljuc cesto uzima vrednosti iz veoma velikog opsega mogucih vrednosti ogranicenog samo brojem pozicija p i brojem dozvoljenih vrednosti v koju svaka pozicija moze imati. Moze uzimati jednu od v na p ili (v na p) - 1.
Kod velicine adresnog prostora dodeljenog datoteci imamo broj lokacija Q = B*b gde je B broj baketa a b broj lokacija u baketu, i N koji predstavlja broj aktuelnih slogova u datoteci - po pravilu je mnogo manji od broja mogucih vrednosti kljuca v na p >> Q >> N.
Metode probabilisticke transformacije se uvode kako bi se prevazisli nedostaci do kojih dovodi deterministicka transformacija vrednosti kljuca u adresu. Ciljevi :
- sto ravnomernija raspodela slogova u adresnom prostoru
- pseudoslucajna transformacija vrednosti kljuca u adresu
- pravilno dizajniranje potrebog adresnog prostora
Koraci probabilisticke transformacije vrednosti kljuca - h : dom(K) -> A:
1. pretvaranje nenumericke u numericku vrednost kljuca:
- k(S) $\in$ {0,1,....(v na p) - 1}, ocekivano je osnova brojnog sistema v = 10 a p je broj cifara numericke vrednosti kljuca
2. pretvaranje numericke vrednosti kljuca k(S) u pseudoslucajan broj T(k(S)) ili skraceno T , T $\in$ {0,1,....,(v na n) - 1}, gde je n dozvoljeni broj cifara relativne adrese A $\in$ {1,...,B}.Racuna se kao n = $\lceil \log_v B \rceil$, 1 <= n <= p
3. dovodjenje vrednosti pseudoslucajnog broja T u opseg dozvoljenih vrednosti relativne adrese {1,...,B}. A = 1 + $\lfloor k*T \rfloor$, k = B / v na n, 0.1 < k <= 1, A $\in$ {1,...,B}.

4. pretvaranje relativne u masinsku adresu - opsti zadatak metode pristupa.Rezultat primene prva 3 koraka je relativna adresa.
Tri cesto upotrebljavane metode probabilisticke transformacije:
- metoda ostatka pri deljenju
- metoda centralnih cifara kvadrata kljuceva
- metoda preklapanja

Metoda ostatka pri deljenju

- relativna adresa A se dobija na sledeci nacin:

Pseudoslucajni broj T dobijamo kada uzmemo celobrojni ostatak pri deljenju numericke vrednosti kljuca sa celim brojem m takvim da vazi v na (n-1) < m <= v na n. T = k(S) (mod m).
Adresa A se racuna kao - A = 1 + $\lfloor B/m * T \rfloor$, k = B/m. Ako izaberemo da je m = B, onda vazi : A = 1 + k(S) (mod B)
Kako bi rezultat transformacije bio sto slucajniji broj a transformacija sto ravnomernija , m ne treba da bude paran broj jer tada neparne vrednosti kljuca daju neparne vrednosti relativne adrese a parne vrednosti kljuca daju parne adrese. m ne treba da bude stepen osnove brojnog sistema jer bi tada cifre najmanje tezine kljuca predstavljale relativnu adresu bez obzira na vrednost ostalih cifara.Najpogodnije je da m predstavlja prost broj ili neparan broj sa relativno velikim prostim ciniocima.
Pogodna je za primenu kada se vrednosti kljuca javljaju u paketima - pojedini intervali u opsegu dozovljenih vrednosti kljuca gusto su zaposednuti aktuelnim vrednostima kljuca, izmedju njih su intervali sa neaktuelnim vrednostima kljuca. Slogovi sa sukcesivnim vrednostima kljuca iz paketa dobijaju adrese fizicki susednih baketa sto rezultuje ravnomernim zaposedanjem baketa.

Metoda centralnih cifara kvadrata kljuca

Vrednost kljuca se dize na kvadrat.Uzima se onoliko centralnih cifara kvadrata vrednosti kljuca koliko pozicija treba da sadrzi relativna adresa - formira se pseudoslucajni broj T.Vrsi se centriranje i normiranje pseudoslucajnog broja na zadati opseg relativnih adresa.Vrednost kljuca k(S) $\in$ {0,1,...,(v na p) - 1} u polinomijalnom obliku :
k(S) = Σ od i =0 do p-1 (ai * v na i), gde ai $\in$ {0,1,...v-1}

Kvadrat u vrednosti kljuca u polinomijalnom obliku je :
(k(S))na2 = Σ od i = 0 do 2p - 1 (ci * v na i) gde ci $\in$ {0,1,...,v-1}
Iz niza cifara kvadrata kljuca (C_{2p-1},C_{2p-2},....,C_1,C_0) izdvaja se podniz od n centralnih cifara (C_{t+n-1},C_{t+n-2},...,C_{t+1},C_t) gde je t pozicija najlakse cifre podniza $t = \lfloor p - n/2 \rfloor$.
Formira se pseudoslucajni broj T : T = Σ od i = 0 do n-1 (C_{t+i} * v na i)
Relativna adresa maticnog baketa A: A = 1 + $\lfloor$B/(v na n) * T$\rfloor$

Metoda preklapanja

Cifre kljuca premestaju se kao pri savijanju tj. preklapanju hartije.Vrsi se sabiranje preklopljenih vrednosti po modulu v na n. Pogodna je za primenu kada je broj pozicija vrednosti kljuca p mnogo veci od broja pozicija relativne adrese n.Preklapanje se izvodi po osama koje zdesna u levo odredjuje broj pozicija n relativne adrese.Broj segmenata za preklapanje se odredjuje kao $\lceil p/n \rceil$.
Pseudoslucajan broj T se racuna kao :

$$T = \left(\sum_{k=0}^{q} \sum_{i=0}^{n-1} c_r v^i + \sum_{k=1}^{q} \sum_{i=0}^{n-1} c_s v^i \right) (\mathrm{mod}\, v^n)$$

$$q = \left\lfloor \frac{p}{2n} \right\rfloor, \; r = 2kn + i, \; s = 2kn - i - 1$$

$$c_r = \begin{cases} c_r, & za \;\; r < p \\ 0, & za \;\; r \geq p \end{cases} \quad \text{i} \quad c_s = \begin{cases} c_s, & za \;\; s < p \\ 0, & za \;\; s \geq p \end{cases}$$

Relativna adresa maticnog baketa racuna se kao: A = 1 + $\lfloor$B/(v na n) * T$\rfloor$

<u>Karakteristike probabilisticke transformacije:</u>

Pojava sloga sinonima nastaje kada dve razlicite vrednosti kljuca mogu transformacijom dobiti istu relativnu adresu.Slogovi sinonimi - slogovi koji transformacijom dobiju iste relativne adrese:
k(Si) $\neq$ k(Sj) , h(k(Si)) = h(k(Sj)).
Maticni baket je baket cija relativna adresa predstavlja rezultat transformacije.Slogovi se uvek smestaju u maticni baket dok se ne popuni.
Primarni slog je slog koji je smesten u maticni baket.
Metoda transformacije particionira skup mogucih vrednosti kljuca na B skupova

sinonima.
Prekoracilac je slog koji ne moze biti smesten u maticni baket usled njegove popunjenosti.Mora se smestiti u neki drugi baket pa se definise poseban postupak za smestanje prekoracilaca.Pojava prekoracilaca je nepozeljna jer zahteva poseban postupak za pronalazenje nove slobodne lokacije za smestaj prekoracilaca, a i dovodi do produzenja vremena pristupa pri kasnijem trazenju slogova prekoracilaca.
Verovatnoca pojave sinonima zavisi od raspodele vrednosti kljuca unutar opsega dozvoljenih vrednosti, odabrane metode transformacije i faktora popunjenosti memorijskog prostora. q = N / Q = N / (b*B)
Broj prekoracilaca ce biti manji sto su slogovi ravnomernije rasporedjeni po baketima , sto je faktor popunjenosti manji , sto je faktor baketiranja veci. Pojava prekoracilaca je neminovnost.
Transformacija vrednosti kljuca u adresu je serija od N nezavisnih eksperimenata. Verovatnoca dodele bilo koje adrese vrednosti kljuca je u idealnom slucaju jednaka i za bilo koji baket iznosi 1/B. Broj slogova u jednom skupu sinonima je slucajna velicina s binomnom raspodelom:

$$P(i) = \binom{N}{i}\left(\frac{1}{B}\right)^{i}\left(1-\frac{1}{B}\right)^{N-i}$$

Za velike vrednosti B i N vazi da 1/B -> 0 a N/B = q*b je konstantan. Imamo Poasonovu raspodelu:

$$P(i) = \frac{(qb)^{i}}{i!}e^{-qb}$$

Ocekivani broj sinonima u svakom skupu sinonima:

$$\overline{S} = \sum_{i=0}^{N} iP(i) = \frac{N}{B} = qb$$

Broj prekoracilaca po baketu L je slucajna velicina sa binomnom ili Poasonovom raspodelom i uzima vrednosti u opsegu $L \in \{0,1,...N\text{-}b\}$.

Ocekivani broj prekoracilaca po baketu:

$$\bar{L} = \sum_{i=b+1}^{N}(i-b)P(i) = \sum_{i=0}^{N}(i-b)P(i) - \sum_{i=0}^{b}(i-b)P(i)$$

$$\bar{L} = b(q-1) - \sum_{i=0}^{b}(i-b)P(i)$$

Ukupan ocekivani broj prekoracilaca : L(nadvuceno)*B - L je nezavisna slucajna promenljiva za svaki baket.

Ocekivani broj prekoracilaca po jednom slogu:

$$\frac{\bar{L}B}{N} = \frac{1}{qb}\sum_{i=b+1}^{N}(i-b)P(i) = \frac{q-1}{q} - \frac{1}{qb}\sum_{i=0}^{b}(i-b)P(i)$$

Izbor faktora popunjenosti ostvaruje veliki uticaj na karakteristike rasuto organizovane datoteke.Za malo q verovatnoca pojave vise slogova u jednom skupu sinonima je takodje mala ali je malo i iskoriscenje memorijskog prostora.Za veliko q (blizu 1) iskoriscenje memorijskog prostora je dobro, ali je velika verovatnoca pojave sinonima i prekoracilaca. U praksi se bira q <= 0.8

Izbor faktora baketiranja utice na ocekivani broj prekoracilaca po jednom baketu pri datom faktoru popunjenosti q.Sa porastom faktora baketiranja b verovatnoca pojave prekoracilaca opada, ali raste vreme razmene sadrzaja baketa izmedju diska i OM (putem jednog baketa ucitava se veci broj potencijalno nepotrebnih slogova u OM).Sa smanjenjem faktora baketiranja b, povecava se ocekivani broj prekoracilaca, za isti N i q povecava se broj baketa B i poboljsava se preciznost transformacije. U praksi, bira se b <= 10.

Postupci za smestaj prekoracilaca

- smestaj svih prekoracilaca unutar jedinstvenog adresnog prostora - izbor posebnog postupka za pronalazenje prazne lokacije za smestaj prekoracioca
- smestaj svih prekoracilaca u posebnu zonu adresnog prostora - datoteka poseduje dve zone, primarnu i zonu prekoracenja (prekoracioci se smestaju u neku od slobodnih lokacija unutar zone prekoracenja), izbor vrste fizicke organizacije zone prekoracenja
- kombinacija prethodna 2 nacina - lokalne zone prekoracenja u okviru primarne zone, glavna zona prekoracenja - posebna zona

Projektovanje rasute datoteke

- pri projektovanju se utvrdjuju faktor popunjenosti memorijskog prostora q, faktor baketiranja b, metoda transformacije vrednosti kljuca u adresu i postupak za smestanje prekoracilaca.
- ova opredeljenja se donose s obzirom na raspodelu vrednosti kljuca unutar opsega mogucih vrednosti, velicinu sloga, obim i karakter azuriranja datoteke i ocekivani srednji broj pristupa datoteci pri uspesnom i neuspesnom trazenju

Za zadati b, q i N, broj baketa je $B = \lceil N / bq \rceil$

Aktivnosti formiranja rasuto organizovane datoteke realizuju se potpuno uz pomoc metode pristupa ako je podrzava ili delom uz pomoc aplikativnog programa i delom uz pomoc relativne metode pristupa - (aplikativni program: transformacija vrednosti kljuca, smestanje prekoracilaca, formiranje baketa od vise slogova, izdvajanje slogova iz baketa), (relativna metoda: direktni upis i citanje sadrzaja baketa)

Formiranje rasute datoteke

1. inicijalno alociranje prazne datoteke - staticka alokacija kompletnog praznog prostora datoteke, izbor postupka prepoznavanja slobodnih lokacija unutar baketa (putem sadrzaja statusnog polja, upisom specijalnih znakova u lokaciju, putem indeksa slobodnih lokacija i njihovog sprezanja)
2. upisivanje slogova u rasutu datoteku - saglasno opstem postupku formiranja rasutih datoteka, na osnovu sadrzaja vodece datoteke kojoj se sekvencijalno pristupa ili direktnim upisivanjem slogova u realnom vremenu

Imamo 2 nacina formiranja:

(A) formiranje u jednom prolazu - slogovi se upisuju u hronoloskom redosledu nastanka, bilo citanjem sadrzaja vodece datoteke u sekvencijalnom pristupu, bilo direktnim unosom podataka u realnom vremenu

(B) formiranje u dva prolaza - slogovi se ucitavaju iz vodece datoteke i upisuju u rasutu

- (I) prvi prolaz - upisuju se samo oni slogovi koji ce biti smesteni u maticne bakete
- (II) drugi prolaz - upisuju se preostali slogovi - prekoracioci, saglasno izabranom postupku za smestanje prekoracilaca

Formiranje u 2 prolaza ima smisla kod datoteka sa jedinstvenim adresnim

prostorom jer se time umanjuje ukupan broj prekoracilaca.

Trazenje sloga u rasutoj datoteci

- trazenje logicki narednog = slucajno odabranog sloga - vrsi se metodom transformacije argumenta u adresu baketa. Ako se slog ne nadje u maticnom baketu a maticni baket ima prekoracilaca, trazenje se nastavlja saglasno izabranom postupku za smestanje i trazenje prekoracilaca (linearna metoda, ponovna transformacija, metoda pracenja pokazivaca).

Za zadati q i b, efikasnost trazenja zavisi od primenjenog postupka za smestaj prekoracilaca.

Vrste statickih rasutih organizacija:

1. rasute datoteke sa jedinstvenim adresnim prostorom :
- datoteka sa linearnim trazenjem lokacije za smestaj prekoracilaca - sa otvorenim nacinom adresiranja (s fiksnim korakom k, k>=1 ili sa slucajno odabranim korakom k, k>=1)
- datoteka sa sprezanjem prekoracilaca u jedinstvenom adresnom prostoru - primarnoj zoni

2. rasute datoteke sa zonom prekoracenja
- sa serijskom zonom prekoracenja
- sa spregnutom zonom prekoracenja

Rasuta s linearnim trazenjem prekoracilaca

1. s <u>fiksnim korakom k = 1</u>:
- ukoliko je maticni baket popunjen, slog se smesta u prvu narednu slobodnu lokaciju. $A_0 = h(k(S))$, $A_n = 1 + A_{n-1} \bmod B$, za $n >= 1$
- trazenje slucajno odabranog sloga - prekoracioca vrsi se linearnom metodom a zaustavlja se na :
    - pronadjenom slogu ako je uspesno
    - prvoj slobodnoj lokaciji ako je neuspesno
    - ponovnim nailaskom na maticni baket ako je neuspesno a cela datoteka kompletno popunjena
- upis novog sloga - prekoracioca - pronalazenjem prve slobodne lokacije iza maticnog baketa
- brisanje postojeceg sloga :
    - logicko - potrebna su 3 statusa sloga (aktuelan, neaktuelan i slobodna

lokacija)

- fizicko - uz lokalnu reorganizaciju memorijskog prostora. Kada ne postoje prekoracioci, brisanjem primarnog sloga oslobadja se lokacija.Kada postoje prekoracioci, svi prekoracioci (ne samo za dati maticni baket) pomeraju se za jednu poziciju prema maticnom baketu, pri pomeranju prekoracilac ne sme da predje ispred svog maticnog baketa.

- glavni nedostaci:

(A) efekat nagomilavanja prekoracilaca - prekoracioci iz jednih baketa izazivaju pojavu prekoracilaca iz drugih baketa.Sve vise raste verovatnoca zauzeca prve prazne lokacije iza sve duzeg lanca zauzetih lokacija.

(B) neefikasno trazenje - trazenje se vrsi i u baketima koji ne sadrze slogove iz istog skupa sinonima

(C) neefikasno neuspesno trazenje - zaustavlja se tek nailaskom na prvu slobodnu lokaciju

2. <u>s fiksnim korakom k > 1</u>:

- ukoliko je maticni baket popunjen, slog se smesta u narednu slobodnu lokaciju, udaljenu za k > 1 pozicija. $A_0 = h(k(S))$, $A_n = 1 + (A_{n-1} + k - 1) \bmod B$, za $n >= 1$

k i B moraju biti uzajamno prosti brojevi kako bi se obezbedio u najgorem slucaju siguran obilazak svih mogucih baketa (u cilju pronalaska prve slobodne lokacije)

- glavna motivacija je odlaganje efekta nagomilavanja prekoracilaca

- pozitivan efekat je prekidanje dugackih lanaca zauzetih lokacija - pokusaj da se koliko je to moguce ocuva priblizno jednaka verovatnoca zauzeca bilo koje prazne lokacije.

3. <u>sa slucajno odabranim korakom k >= 1</u>:

- Rasuta datoteka sa slucajnim trazenjem.Ukoliko je maticni baket popunjen, slog se smesta u narednu slobodnu lokaciju udaljenu za k >= 1 pozicija s obzirom na poziciju maticnog baketa. k se odredjuje na slucajan nacin i predstavlja rezultat druge probabilisticke transformacije primenjene na vrednost identifikatora - kljuca sloga.

- imamo dve probabilisticke transformacije:

$A_0 = h_1(k(S)) \in \{1,...,B\}$

$k = h_2(k(S)) \in \{1,...,B-1\}$

$A_n = 1 + (A_{n-1} + k - 1) \bmod B$

$A_n = 1 + (A_{n-1} + h_2 (k(S)) - 1) \bmod B$, za $n >= 1$

k i B moraju biti uzajamno prosti brojevi kako bi se obezbedio u najgorem slucaju siguran obilazak svih mogucih baketa u cilju pronalaska prve slobodne lokacije.Posto je k slucajna velicina, bira se da B bude prost broj.Cesto je $k = h_2 (k(S)) = 1 + k(S) \bmod (B - 1)$

- glavna motivacija - izbegavanje efekta nagomilavanja prekoracilaca
- pozitivan efekat je prekidanje dugackih lanaca zauzetih lokacija - bolji pokusaj da se koliko je to moguce ocuva priblizno jednaka verovatnoca zauzeca bilo koje prazne lokacije

Rasuta sa otvorenim nacinom adresiranja

Pogodne su za upotrebu u slucaju manje popunjenosti , q <= 0.7 kao i u slucaju nizeg intenziteta azuriranja.

Rasuta sa sprezanjem u primarnoj zoni

Imamo primenu tehnike sprezanja i slozeniju fizicku strukturu.Ukoliko je maticni baket popunjen, slog se smesta u prvu slobodnu lokaciju iz lanca slobodnih lokacija.Sprezu se dvostruko baketi sa slobodnim lokacijama a bitan nam je i specijalni baket s pokazivacem na pocetak lanca.Vrsi se sprezanje svih sinonima u odnosu na maticni baket.Za svaki maticni baket imamo po jedan lanac sinonima.Pokazivac na pocetak lanca je u zaglavlju maticnog baketa. Pokazivac na sledeci u lancu sinonima je ugradjen u svaki slog.

Format baketa i sloga:

Ai

pokazivac na lanac baketa sa slobodnim lokacijama

u	b	n	e	Ai na 1			Ai na 2			Ai na 3		
				K(S1)	P(S1)	U(S1)	K(S2)	P(S2)	U(S2)	K(S3)	P(S3)	U(S3)

u(S2) - pokazivac na sledeci u lancu sinonima maticnog baketa

u - pokazivac na lanac sinonima maticnog baketa

b - pokazivac na prethodni u lancu baketa sa slobodnim lokacijama

n - pokazivac na naredni u lancu baketa sa slobodnim lokacijama

e - broj slobodnih lokacija u baketu

- trazenje slucajno odabranog sloga:
    - transformacija vrednosti kljuca u adresu A = h(k(S)) i pristupanje maticnom baketu
    - pracenje lanca sinonima, zapocinjuci od pokazivaca *u* - metodom pracenja pokazivaca
    - zaustavlja se na pronadjenom slogu ako je uspesno ili na kraju lanca sinonima ako je neuspesno
- upis novog sloga:
    - nakon neuspesnog trazenja uvezivanjem u lanac sinonima, izborom prve prazne lokacije iz lanca baketa sa slobodnim lokacijama na koju ukazuje E
- brisanje postojeceg sloga - fizicko, uklanjanjem sloga iz lanca sinonima uz potrebno prevezivanje.Oslobadjanje lokacije uz eventualno vracanje baketa u lanac baketa sa slobodnim lokacijama.
- glavna motivacija je izbegavanje efekta neefikasnog trazenja (B) i (C) - trazenje se vrsi samo u baketima koji sadrze slogove iz istog skupa sinonima.Neuspesno trazenje zaustavlja se dolaskom do kraja lanca spregnutih slogova - sinonima.
- pozitivan efekat je poboljsana efikasnost trazenja (narocito neuspesnog) u odnosu na datoteke sa otvorenim nacinom adresiranja
- negativan efekat je i dalje moguc efekat nagomilavanja prekoracilaca (A) kao i komplikovanija fizicka struktura

Rasuta sa sprezanjem u zoni prekoracenja

Uvodimo zonu prekoracenja - spregnuta datoteka.Imamo primenu tehnike sprezanja i slozeniju fizicku strukturu. Ukoliko je maticni baket popunjen, slog se smesta u prvu slobodnu lokaciju iz lanca slobodnih lokacija u zoni prekoracenja.Sprezu se jednostruko baketi sa slobodnim lokacijama u zoni prekoracenja, bitan nam je specijalni baket s pokazivacem na pocetak lanca.Vrsi se sprezanje svih prekoracilaca.Za svaki maticni baket imamo po jedan lanac prekoracilaca.Pokazivac na pocetak lanca nalazi se u zaglavlju maticnog baketa.Pokazivac na sledeci u lancu prekoracilaca ugradjen je u svaki slog u zoni prekoracenja.

Imamo dimenzionisanje spregnute zone prekoracenja.Tipican faktor blokiranja je f = 1.Mala je verovatnoca da se dva prekoracioca iz istog lanca sinonima nadju u susednim lokacijama. Kapacitet odnosno maksimalni broj blokova je :

$$Z = \lceil \bar{L}B + k\sigma\sqrt{B} \rceil$$

- ocekivana vrednost ukupnog broja prekoracilaca uvecana za standardnu devijaciju pomnozenu sa fakorom sigurnosti k (normalna raspodela, tipicno k = 3). Formiranje je uvek u jednom prolazu.Svi prekoracioci su u zoni prekoracenja koja je odvojena od primarne zone.
- trazenje slucajno odabranog sloga:
    - transformacija vrednosti kljuca u adresu A = h(k(S)) i pristupanje maticnom baketu
    - pracenje lanca prekoracilaca, zapocinjuci od pokazivaca na pocetak lanca u maticnom baketu ukoliko slog nije pronadjen u maticnom baketu a postoji lanac prekoracilaca, metodom pracenja pokazivaca.
    - zaustavlja se na pronadjenom slogu ako je uspesno ili na kraju lanca prekoracilaca ako je neuspesno.
- upis novog sloga - nakon neuspesnog trazenja u maticni baket ako ima mesta.Ako u maticnom baketu nema mesta onda se uvezuje u lanac prekoracilaca izborom prve prazne lokacije iz lanca baketa sa slobodnim lokacijama u zoni prekoracenja na koju ukazuje pokazivac na pocetak lanca.
- brisanje postojeceg sloga - fizicko, uklanjanjem sloga iz maticnog baketa uz eventualno prebacivanje prvog prekoracioca u maticni baket ili uklanjanjem sloga iz lanca prekoracilaca uz potrebno prevezivanje.
- glavna motivacija je izbegavanje efekta neefikasnog trazenja (B) i (C) - trazenje se vrsi samo u baketima koji sadrze slogove iz istog skupa sinonima, a neuspesno trazenje se zaustavlja dolaskom do kraja lanca spregnutih slogova - sinonima. Motivacija je takodje i uklanjanje efekta nagomilavanja prekoracilaca (A) - svi prekoracioci u zoni prekoracenja - spregnuta datoteka
- pozitivan efekat je poboljsana efikasnost trazenja (narocito neuspesnog) - u odnosu na datoteke sa jedinstvenim adresnim prostorom

Rasuta sa serijskom zonom prekoracenja

Uvodimo zonu prekoracenja - serijska datoteka.Ukoliko je maticni baket popunjen, slog se smesta u prvu slobodnu lokaciju u serijskoj zoni prekoracenja.Jednostavna je struktura jer nema dodatnih polja pokazivaca.Pogodna je u slucaju manjeg ocekivanog ukupnog broja prekoracilaca

jer se ne isplati sprezanje prekoracilaca u zoni prekoracenja.

Ocena trazenja sloga u rasutoj datoteci

Ocekivani broj prekoracilaca po jednom slogu:

$$\frac{\overline{L}B}{N} = \frac{q-1}{q} - \frac{1}{qb}\sum_{i=0}^{b}(i-b)P(i)$$

pri zadatom q i B, ne zavisi od broja slogova N, odnos N / B je konstantan.

Ocekivani broj pristupa pri uspesnom i neuspesnom trazenju zavisi od q i b a ne od N. Karakteristike velikih rasutih datoteka mogu se procenjivati poredjenjem sa malim datotekama ali sa istim q i b.

Ocekivani broj pristupa za uspesno trazenje slucajno odabranog sloga kod svih vrsta je :

$$\overline{R}_u = \frac{1}{N}\sum_{i=1}^{N} R_i^u$$

Ocekivani broj pristupa za neuspesno trazenje slucajno odabranog sloga kod svih vrsta osim kod datoteke sa slucajnim trazenjem prekoracilaca je :

$$\overline{R}_n = \frac{1}{B}\sum_{i=1}^{B} R_i^n$$

prebroji se potreban broj pristupa za svaki maticni baket.

Ocekivani broj pristupa za neuspesno trazenje slucajno odabranog sloga kod datoteke sa slucajnim trazenjem prekoracilaca je :

$$\overline{R}_n = \frac{1}{T}\sum_{i=1}^{T} R_i^n \quad \text{ili} \quad \overline{R}_n = \frac{1}{B(B-1)}\sum_{i=1}^{B(B-1)} R_i^n$$

- prebroji se potreban broj pristupa za svaku neaktuelnu vrednost kljuca , T = v na p - N ili svaki maticni baket i svaku mogucu vrednost koraka k $\in$ {1,...,B-1}

Obrada rasute datoteke sa probabilistickom transformacijom

Nepogodne za koriscenje u ulozi osnovne vodece datoteke.Mogu se koristiti kao obradjivane i vodece u rezimu direktne obrade.Ne mogu se koristiti kao vodece u rezimu redosledne obrade posto fizicka struktura ne sadrzi informaciju o logickoj strukturi podataka.Mogu se obradjivati i u rezimu redosledne i u rezimu direktne

obrade.Performanse redosledne i direktne obrade rasute datoteke su iste zbog iste efikasnosti trazenja i logicki narednog i slucajno odabranog sloga.Ocekivani broj pristupa je :

$$\overline{R}_{uk} = N_v^u \overline{R}_u + N_v^n \overline{R}_n$$

broj slogova vodece datoteke je :

$$N_v = N_v^u + N_v^n$$

Ru (nadvuceno) - ocekivani broj pristupa pri uspesnom trazenju 1 sloga
Rn (nadvuceno) - ocekivani broj pristupa pri neuspesnom trazenju 1 sloga

Oblasti primene rasutih datoteka

- u svim mreznim SUBP, u pojedinim relacionim SUBP, u interaktivnoj obradi podataka, u rezimu paketne obrade.

Prednost je mali ocekivani broj pristupa pri trazenju slucajno odabranog sloga.
Nedostaci:

- potreba da se unapred odredi velicina datoteke - pogodne samo za datoteke ciji se sadrzaj redje menja.Intenzivno upisivanje novih slogova dovodi do degradacije performansi obrade.
- problem izbora probabilisticke transformacije - ravnomerna raspodela broja sinonima po baketima i pri formiranju i pri azuriranju.
- broj pristupa pri trazenju moze biti nepredvidivo velik.

12.PREZENTACIJA

Indeksne datoteke

Ideja je postojanje indeksa kao pomocnih struktura podataka, realizovanih u posebnoj datoteci kao stablo trazenja.Sadrze parove (vrednost kljuca, relativna adresa sloga/bloka) za preslikavanje argumenta trazenja u adresu sloga kao i za brz pristup slucajno odabranom i logicki narednom slogu.Smestaju se kompletni slogovi u posebnu datoteku - zonu podataka (primarnu zonu) koja moze biti razlicito organizovana.

Vrste indeksnih datoteka:
1. staticke - istorijski prve, danas redje u prakticnoj upotrebi. Staticka alokacija memorijskog prostora se definise prilikom projektovanja organizacije.Staticki indeks se formira i nakon formiranja se ne azurira.
2. dinamicke - prakticno nezaobilazne u realnim projektima.Dinamicka alokacija memorijskog prostora.Dinamicki indeks se azurira paralelno sa azuriranjem zone podataka.Pozeljne karakteristike se odrzavaju u vremenu.
Efikasnost organizacija u upotrebi:
- sekvencijalne datoteke - ideal redosledne obrade i trazenja logicki narednih slogova
- rasute datoteke - ideal direktne obrade i trazenja slucajno odabranih slogova
- indeksne datoteke - struktura kompromisa, solidna podrska direktne i redosledne obrade kao i obe vrste trazenja

Indeks-sekvencijalna organizacija datoteke

Staticka indeks-sekvencijalna datoteka
- Sadrzi 3 memorijske zone ili osnovne organizacije datoteke:
1. primarna zona ili zona podataka - sekvencijalna organizacija
2. zona indeksa - spregnuta ogranizacija (n-arno stablo)
3. zona prekoracenja - spregnuta organizacija (lanci prekoracilaca)

Primarna zona:
Slogovi su uredjeni saglasno rastucim vrednostima kljuca.Grupisani su u blokove pri cemu je pozeljan sto veci faktor blokiranja. Kreira se u postupku formiranja staticke indeks-sekvencijalne datoteke.Nikada se naknadno ne azurira.Ciljevi:
- iskoristiti pozeljne osobine sekvencijalne organizacije u redoslednoj obradi podataka
- izbeci efekat losih performansi azuriranja sekvencijalno organizovane datoteke

Zona indeksa:
- Puno stablo trazenja i spregnuta struktura - reda n (n >= 2), visine h
- Cvor stabla = blok, sadrzi od 1 do n elemenata. Imamo parove (k_e,A_e) , e $\in$ {1,...,n} gde je n faktor blokiranja u zoni indeksa, k_e = k(S) - vrednost kljuca sloga S,

A_e je adresa bloka primarne zone u kojem je slog S u slucaju lista ili adresa podstabla , tj. drugog cvora stabla trazenja koji takodje sadrzi k_e u slucaju neterminalnog cvora. Elementi u cvoru uredjeni saglasno rastucim vrednostima kljuca k_e - cvor je sekvencijalno organizovana struktura.
Koristi se retko popunjeni indeks - reprezentativne vrednosti kljuca svakog bloka primarne zone propagirane u stablo. Vrednosti kljuca k_e u stablu su najmanje ili najvece vrednosti kljuca iz svakog bloka primarne zone.Elementi listova stabla sadrze po jednu vrednost kljuca iz svakog bloka. Elementi cvorova na visim nivoima hijerarhije stabla sadrze po jednu vrednost kljuca iz svakog direktno podredjenog cvora. Vrednosti kljuca ponavljaju se u cvorovima na svim nizim nivoima hijerarhije.
Neterminalni cvor sa m (1 <= m <= n) elemenata poseduje m direktno podredjenih cvorova. Adresa A_e je pokazivac tj relativna adresa.
Vrste zone indeksa:
- zona indeksa sa propagacijom najvecih vrednosti kljuca iz svakog bloka - u slucaju poslednjeg bloka , propagira se ne aktuelna najveca vrednost kljuca, vec najveca dozvoljena vrednost kljuca
- zona indeksa sa propagacijom najmanjih vrednosti kljuca iz svakog bloka - u slucaju prvog bloka, propagira se ne aktuelna najmanja vrednost kljuca, vec najmanja dozvoljena vrednost kljuca
- broj cvorova Ci na i-tom nivou hijerarhije stabla (i = 1,...,h):

$$C_i = \left\lceil \frac{B}{n^{h-i+1}} \right\rceil$$

B - broj blokova u primarnoj zoni (B >= 1)
- visina stabla :

$$h = \lceil \log_n B \rceil$$

- ukupni broj cvorova stabla C :

$$C = \sum_{i=1}^{h} \left\lceil \frac{B}{n^{h-i+1}} \right\rceil$$

- kapacitet stabla - koliko parova (k_e,A_e) se moze upisati u cvorove : K = n*C

- ukupni broj elemenata E u stablu pristupa:

$$E = \sum_{i=1}^{h} \left\lceil \frac{B}{n^{h-i}} \right\rceil$$

Stablo trazenja obezbedjuje relativno brz pristup za trazenje slucajno odabranog sloga.

Zona prekoracenja:

Sadrzi kompletne slogove datoteke kao i primarna zona koji se upisuju u zonu prekoracenja pri upisu novih slogova i nazivaju se prekoracioci. Svaki blok primarne zone moze imati svoje prekoracioce.

Posmatra se list u stablu trazenja sa m elemenata (1 <= m <= n):

(k_1,A_1) ,..., (k_{e-1},A_{e-1}) , (k_e,A_e) , (k_{e+1},A_{e+1}) ,... ,(k_m,A_m)

A_e, e $\in$ {1,...,m} - adrese sukcesivnih blokova primarne zone. Ako stablo sadrzi najvece vrednosti kljuca , slog sa k(S) je u bloku sa adresom A_1 ako je k(S) < k_1 a u bloku sa adresom A_e, ako je k_{e-1} < k(S) <= k_e, e $\in$ {2,...,m}.

Kada blok u primarnoj zoni nije kompletan (m ≠ n) - upis novog sloga dovodi samo do pomeranja slogova u bloku.

Kada je blok u primarnoj zoni kompletan (m = n) - upis svakog novog sloga dovodi do upisa jednog od slogova koji pripadaju bloku sa adresom A_e, e $\in$ {1,...,m} u zonu prekoracenja.Ako uzmemo da je $k_{e(max)}$ trenutno maksimalna vrednost kljuca u bloku sa adresom A_e. Ako je k(S) < $k_{e(max)}$ - novi slog se upisuje u blok a svi slogovi sa vrednoscu kljuca vecom od k(S) pomeraju se za jednu lokaciju ka kraju bloka.Slog sa vrednoscu kljuca $k_{e(max)}$ se upisuje u zonu prekoracenja.Ako je k(S) > $k_{e(max)}$ novi slog se upisuje u zonu prekoracenja.

Sprezu se logicki neposredno susedni prekoracioci iz jednog bloka, faktor blokiranja je fz = 1. Za svaki blok primarne zone imamo najvise jedan lanac spregnutih prekoracilaca.Slogovi u svakom lancu prekoracilaca uredjeni su rastucem (alternativno opadajucem) redosledu. Dodatno, imamo lanac slobodnih blokova.

Postoje dva nacina za pokazivac na pocetak lanca:

1. direktno povezivanje sa listom stabla trazenja - pokazivac na pocetak lanca smesta se u odgovarajuci list.

2. indirektno povezivanje sa listom stabla trazenja - pokazivac na pocetak lanca smesta se u prateci deo odgovarajuceg bloka u primarnoj zoni.

Indeks-sekvencijalna metoda pristupa

Podrzana je najcesce sistemima za upravljanje datoteka ugradjenim u OS mainframe racunara, a redje savremenim SUBP-ovima. Obezbedjuje formiranje,trazenje,azuriranje i reorganizaciju, kao i sekvencijalni, direktni i dinamicki nacin pristupa indeks-sekvencijalnoj datoteci.

Formiranje IS datoteke:

Program redosledno ucitava slogove ulazne sekvencijalne datoteke, smesta blokove u primarnu zonu IS datoteke.Alternativno, vec formirana sekvencijalna datoteka proglasava se primarnom zonom IS datoteke.

Formiranje zone indeksa:

Iterativan postupak, po nivoima stabla trazenja - s dna na gore -> s leva na desno Prvo se formiraju svi listovi - cvorovi nivoa h, zatim cvorovi nivoa h-1,itd do cvorova nivoa 1.U svaki cvor na i-tom nivou hijerarhije (i = h-1, h-2,...,1) upisuju se najvece (alternativno najmanje) vrednosti kljuca iz n sukcesivnih cvorova na nivou hijerarhije i+1. Propagacija najvecih vrednosti - u poslednji element krajnjeg desnog cvora upisuje se maksimalna dozvoljena vrednost kljuca.Propagacija najmanjih vrednosti - u prvi element krajnjeg levog cvora upisuje se minimalna dozvoljena vrednost kljuca.

Formiranje zone prekoracenja:

Alocira se prazna zona prekoracenja, svi blokovi se sprezu u lanac slobodnih blokova. Pocetak lanca upisuje se u zaglavlje zone prekoracenja.

Trazenje logicki narednog sloga:

Vrsi se kombinovanom primenom metode linearnog trazenja i metode trazenja pracenjem pokazivaca. Pocinje u prvom bloku primarne zone, svako naredno trazenje se nastavlja od tekuceg sloga datoteke u bloku primarne zone - linearna metoda.Po dolasku do poslednjeg sloga bloka, trazenje se nastavlja u lancu prekoracilaca ako postoji - metoda pracenja pokazivaca. Indirektno povezivanje prekoracilaca - nastavak trazenja direktno u zoni prekoracenja. Direktno

povezivanje prekoracilaca - pristup direktno nadredjenom listu i nastavak trazenja u zoni prekoracenja.

Direktno povezivanje prekoracilaca:

- Pristupa se blokovima primarne zone , prekoraciocima i listovima stabla trazenja.

Broj pristupa R i pri uspesnom i pri neuspesnom trazenju jednog logicki narednog sloga:

$$0 \le R \le B + \left\lceil \frac{B}{n} \right\rceil + Z - (i + j + k)$$

Z - ukupni broj slogova u zoni prekoracenja

i - redni broj tekuceg bloka datoteke u odnosu na pocetak primarne zone

j = ⌈ i/n ⌉ - redni broj tekuceg lista stabla trazenja

k - broj slogova zone prekoracenja kojima se vec pristupilo

Indirektno povezivanje prekoracilaca:

- Broj pristupa R i pri uspesnom i pri neuspesnom trazenju jednog logicki narednog sloga : 0 <= R <= B + Z - (i + k). Trazenje logicki narednog sloga je efikasnije.

Trazenje slucajno odabranog sloga:

Pracenjem pokazivaca u stablu pristupa - zapocinje u korenu i stize do lista u stablu trazenja. Uvazava organizaciju sa propagacijom maksimalnih ili minimalnih vrednosti kljuca iz svakog bloka.

Direktno povezivanje prekoracilaca - dolaskom do odgovarajuceg elementa u listu donosimo odluku o nastavku trazenja (u bloku primarne zone ili pracenjem lanca prekoracilaca u zoni prekoracenja). Broj pristupa R i pri uspesnom i pri neuspesnom trazenju jednog slucajno odabranog sloga - h+1 <= R <= h+z.

Indirektno povezivanje prekoracilaca - prati se pokazivac odgovarajuceg elementa u listu i prelazi se u blok podataka u primarnoj zoni a po potrebi se nastavlja trazenje pracenjem lanca prekoracilaca. Broj pristupa R i pri uspesnom i pri neuspesnom trazenju jednog slucajno odabranog sloga - h+1 <= R <= h+1+z gde je z duzina lanca prekoracilaca za 1 blok primarne zone.

Nesto efikasnije trazenje je u datoteci sa direktnim povezivanjem prekoracilaca.

Obrada IS datoteke:

Moguca je efikasna obrada i u rezimu redosledne i u rezimu direktne obrade.Pogodne su za koriscenje u ulozi vodece datoteke u oba rezima.

Redosledna obrada putem vodece datoteke od Nv slogova se odvija naizmenicnim pristupanjem blokovima primarne zone i njihovim lancima prekoracilaca.Adresa prvog bloka primarne zone nalazi se u zaglavlju datoteke. Ukupan broj pristupa pri redoslednoj obradi Ruk za slucaj direktnog povezivanja je Ruk = B + Z + $\lceil B/n \rceil$ a za slucaj indirektnog povezivanja Ruk = B + Z.
Ocekivani broj pristupa pri uspesnom ili neuspesnom trazenju jednog logicki narednog sloga:
- za slucaj direktnog povezivanja:

$$\overline{R} = \frac{B + Z + \lceil B/n \rceil}{N_v}$$

- za slucaj indirektnog povezivanja:

$$\overline{R} = \frac{B + Z}{N_v}$$

Redosledna obrada je nesto efikasnija kod datoteke sa indirektnim povezivanjem prekoracilaca. Pri uobicajenim vrednostima reda stabla n razlika je neznatna.
Direktna obrada putem vodece datoteke od Nv slogova je Nv = Nv na u + Nv na n.
Ocekivani ukupni broj pristupa:

$$\overline{R}_{uk} = \overline{R}_u N_v^u + \overline{R}_n N_v^n$$

Ru(nadvuceno) - ocekivani broj pristupa pri uspesnom trazenju
Rn(nadvuceno) - ocekivani broj pristupa pri neuspesnom trazenju

Ocekivani broj prekoracilaca po bloku primarne zone : z (nadvuceno) = Z / B.
Za datoteku sa direktnim povezivanjem prekoracilaca je ista verovatnoca zaustavljanja trazenja na bilo kom slogu. Direktna obrada je nesto efikasnija kod datoteke sa direktnim povezivanjem prekoracilaca.

Azuriranje IS datoteke:

Vrsi se u rezimu direktne obrade. Upis novog sloga vrsi se nakon neuspesnog trazenja.Ako se neuspesno trazenje zaustavilo u bloku primarne zone, vrsi se pomeranje slogova sa vecom vrednoscu kljuca od vrednosti kljuca novog sloga za jednu lokaciju ka kraju bloka.Novi slog se upisuje u lokaciju koju je zauzimao slog sa prvom vecom vrednoscu kljuca,a slog sa do tada najvecom vrednoscu kljuca u bloku upisuje se u zonu prekoracenja.Prekoracilac se upisuje u lokaciju ciju adresu

sadrzi indeks slobodnih lokacija i povezuje se sa ostalim prekoraciocima iz bloka. Ako se neuspesno trazenje zaustavilo na nekom od prekoracilaca, novi slog se upisuje u prvu slobodnu lokaciju i uvezuje se sa ostalim prekoraciocima.
Brisanje sloga:
- logicko - cesce, Rd = Ru + 1. Lokacija logicki izbrisanog sloga moze se upotrebiti za upis novog sloga u specijalnom slucaju - kada se vrednost kljuca novog sloga nalazi tacno u odgovarajucim granicama.
- fizicko - zahteva pomeranje slogova sa azuriranjem lanca prekoracilaca.Zahteva veci broj pristupa datoteci.
Modifikacija sadrzaja postojeceg sloga vrsi se nakon neuspesnog trazenja pri cemu je potreban samo jedan pristup da bi se modifikovani slog upisao u datoteku.

Reorganizacija IS datoteke:

Znacajna degradacija performansi trazenja slogova i obrade datoteke u vremenu nastaje usled upisa slogova u zonu prekoracenja i logickog brisanja slogova.
Periodicna reorganizacija datoteke je uklanjanje negativnih posledica azuriranja.
Postupak:
- ponovno formiranje primarne zone redoslednom obradom - trazenjima logicki narednih slogova u postojecoj primarnoj zoni i zoni prekoracenja
- generisanje novog stabla trazenja
- formiranje nove, prazne zone prekoracenja

Interval vremena izmedju dve reorganizacije moze biti :
1. fiksan - npr. jednom mesecno
2. dinamicki odredjen - na osnovu stepena popunjenosti zone prekoracenja, npr. kada se zona prekoracenja popuni do 80% svog obima, a dimenzionisana je da primi npr. 10% slogova primarne zone

Distribuirani slobodni prostor ublazava problem degradacije performansi obrade zbog upisa novih slogova - blokovi podataka se pri formiranju datoteke popunjavaju samo delimicno (npr. 60% ili 80%), time se obezbedjuje prostor za upis novih slogova.Produzava se interval vremena izmedju dve reorganizacije.
IS datoteka se moze azurirati i kao sekvencijalna u rezimu redosledne obrade - po zavrsetku azuriranja generise se novo stablo trazenja.Ovakav postupak predstavlja istovremeno i reorganizaciju datoteke.

Oblasti primene i ocena karakteristika

- prednosti :

1. kada iste podatke treba obradjivati i u rezimu redosledne i u rezimu direktne obrade
2. intenzivno se koristi u paketnoj obradi
3. moze se koristiti i u interaktivnoj obradi

- performanse redosledne obrade u pocetku ne zaostaju za performansama redosledne obrade sekvencijalne datoteke
- performanse direktne obrade ne zaostaju znacajnije za performansama direktne obrade rasute datoteke
- glavni nedostatak :

Upis slogova u zonu prekoracenja dovodi do degradacije performansi obrade.Performanse redosledne obrade mogu se drzati pod kontrolom - pogodnim dimenzionisanjem zone prekoracenja.Performanse trazenja slucajno odabranog sloga i direktne obrade znacajnije se degradiraju.Jedino resenje: periodicno reorganizovanje datoteke koje je nepogodno ako se mora cesto sprovoditi u slucaju datoteka velikog obima.

- FSP zasnovane na statickoj IS organizaciji intenzivno su koriscene u mreznim sistemima baza podataka.Relacioni i objektno-relacioni SUBP ih retko podrzavaju.
- Ponekad se istice vise njihov istorijski znacaj, nazivaju se i klasicnim indeks-sekvencijalnim datotekama.Preteca su modernih indeksnih datoteka sa B-stablima.
- Osnovna ideja za primenu IS datoteka:
    - kada se podaci ne azuriraju intenzivno i u vecem obimu
    - kada je potrebno obezbediti vrlo efikasnu redoslednu obradu i u isto vreme solidne performanse direktne obrade
    - brz pristup slucajno odabranom slogu u sekvencijalnoj strukturi vrsi se koriscenjem stabla trazenja kao funkcije koja preslikava vrednost kljuca u adresu

13.PREZENTACIJA

Osnovno B-stablo

B-stablo:
- puno stablo i stablo trazenja, gusto popunjeni dinamicki indeks. Visina h - svi listovi su na jednakoj udaljenosti od korena, a put od korena do bilo kog lista je iste duzine. Rang r (r >= 2) - uvodi ogranicenje na dozvoljeni broj elemenata u svakom cvoru. Red n = 2r+1
- cvor je blok zone indeksa i sadrzi niz elemenata.Element je propagirana vrednost kljuca iz primarne zone.Svaki cvor sadrzi maksimalno 2r elemenata, a svaki cvor izuzev korena sadrzi minimalno r elemenata dok koren sadrzi minimalno 1 element.Svaki cvor sa m elemenata koji ne predstavlja list poseduje m+1 direktno podredjenih cvorova.

Format cvora B-stabla:
- niz elemenata, gde je svaki element trojka (k_e,A_e,P_e), e $\in$ {1,...,m}.

k_e - vrednost kljuca sloga Si (i $\in$ {1,2,...,N})

A_e - pridruzeni podatak

P_e - pokazivac ka podstablu s vecim vrednostima kljuca od k_e

Zaglavlje bloka	P_0	k_1	A_1	P_1	k_2	A_2	P_2	...	k_m	A_m	P_m	Neiskorišćeni prostor

Uslovi stabla trazenja:
- $(\forall i \in \{1,..., m - 1\})(k_i < k_{i+1})$
- $(\forall k \in K(P_0))(k < k_1)$
- $(\forall i \in \{1,..., m - 1\})(\forall k \in K(P_i))(k_i < k < k_{i+1})$
- $(\forall k \in K(P_m))(k_m < k)$

Popunjenost B-stabla - za isti broj slogova N i rang r, B-stablo moze posedovati razlicite visine i razlicite brojeve cvorova. Ekstremni slucajevi popunjenosti B-stabla:
- poluprazno (polupuno) B-stablo - svi cvorovi osim korena sadrze po r elemenata,

koren sadrzi samo 1 element a stablo ne moze biti popunjeno manje od polupraznog.

- kompletno (popunjeno) B-stablo - svi cvorovi sadrze po 2r elemenata a stablo ne moze biti vise popunjeno od kompletnog

Broj cvorova i elemenata na razlicitim nivoima hijerarhije B-stabla ranga r:

Nivo	Visina	Kompletno B-stablo		Poluprazno B-stablo	
		Broj čvorova	Broj elemenata	Broj čvorova	Broj elemenata
0	1	1	2r	1	1
1	2	$(2r + 1)^1$	$2r(2r + 1)^1$	2	2r
2	3	$(2r + 1)^2$	$2r(2r + 1)^2$	$2(r + 1)^1$	$2r(r + 1)^1$
...	...	...	...	...	...

Broj cvorova stabla kod kompletnog stabla:

$$C_{kp} = \sum_{i=1}^{h} (2r+1)^{i-1} = \sum_{i=0}^{h-1} (2r+1)^{i} = \frac{(2r+1)^h - 1}{2r}$$

Broj cvorova stabla kod polupraznog stabla:

$$C_{pp} = 1 + 2\sum_{i=1}^{h-1} (r+1)^{i-1} = 1 + 2\sum_{i=0}^{h-2} (r+1)^{i} = 1 + 2\frac{(r+1)^{h-1} - 1}{r}$$

<u>Broj cvorova i visina stabla kod kompletnog stabla:</u>

N = 2rCkp

$$h_{kp} = \log_{2r+1}(N+1), \quad h_{min} = \lceil \log_{2r+1}(N+1) \rceil$$

Cmin = $\lceil N/2r \rceil$

Visina ne moze biti manja od hmin, ukupan broj cvorova ne moze biti manji od Cmin

<u>Broj cvorova i visina stabla kod polupraznog stabla:</u>

N = 1 + r(Cpp - 1)

$$h_{pp} = 1 + \log_{r+1}\frac{N+1}{2}, \quad h_{max} = 1 + \left\lfloor \log_{r+1}\frac{N+1}{2} \right\rfloor$$

Cmax = 1 + $\lfloor (N-1)/r \rfloor$

Visina ne moze biti veca od hmax, ukupan broj cvorova ne moze biti veci od Cmax

hmin <= h <= hmax, Cmin <= C <= Cmax

Formiranje datoteke sa B-stablom

Indeksna metoda pristupa:
- kod operativnih sistema - OS mainframe racunara poseduju metode pristupa za formiranje, koriscenje i azuriranje indeksnih datoteka sa B-stablima
- kod programskih jezika - koriscenje specijalizovanih biblioteka za indeksnu metodu pristupa ili korisnici sami pisu svoje metode
- SUBP poseduju sopstvene indeksne metode pristupa i koriste ih u izgradnji fizickih struktura baza podataka

Struktura datoteke s B-stablom:
- zona indeksa je spregnuta struktura sa B-stablom - dinamicki, gusto popunjeni indeks gde se svaka vrednost kljuca primarne zone propagira u zonu indeksa. Dinamicko azuriranje - prati azuriranje primarne zone.Omogucava trazenje u primarnoj zoni.
- primarna zona je serijska datoteka - iskoriscenje dobrih osobina serijske datoteke pri azuriranju pod pretpostavkom da se trazenja ne vrse direktno u serijskoj datoteci.

Formiranje datoteke sa B-stablom:
1. formiranje primarne zone:
- preuzimanjem postojece (serijske) datoteke ili sukcesivnim ucitavanjem slogova ulazne (serijske) datoteke

2. formiranjem zone indeksa:
- upisivanje zaglavlja stabla trazenja u zonu indeksa - inicijalno, nedefinisane vrednosti pokazivaca na koren stabla i krajnji levi cvor stabla, inicijalizuje se pokazivac na prvi blok u lancu praznih blokova
- formiranje lanca praznih blokova (cvorova)
- sukcesivno citanje slogova primarne zone i formiranje B-stabla dinamickim upisivanjem novih elemenata
- upisivanje prvog elementa - formiranje korena stabla

Upis novog elementa u B-stablo:
Prethodi mu neuspesno trazenje elementa (tso) koje zapocinje u korenu i uvek se zavrsava u listu - uporedjivanjem argumenta trazenja s vrednostima kljuca u

svakom cvoru i pracenjem pokazivaca na cvorove na jednom pristupnom putu od korena do lista.Zapocinje na mestu zaustavljanja neuspesnog trazenja u listu pri cemu list moze biti potpuno ili delimicno popunjen.
Ako je aktuelni broj elemenata u listu m_e imamo dve mogucnosti:
(A) Delimicno popunjen list: $m_e < 2r$:
- novi element upisuje se na poziciju zaustavljanja trazenja.Elementi s vecom vrednoscu kljuca u cvoru pomeraju se za jednu poziciju udesno.
(B) Potpuno popunjen list: $m_e = 2r$:
- podela lista na dva lista - alocira se novi desni list. Formira se u OM uredjeni niz od 2r + 1 elemenata gde prva polovina niza ostaje u levom (postojecem) listu, srednji element upisuje se u nadredjeni cvor pa moze izazvati deljenje nadredjenog cvora a druga polovina niza upisuje se u desni (novi) list
- ukoliko se deli koren, dolazi do formiranja novog korena i povecavanja visine stabla za 1.

Trazenje u datoteci s B-stablom:

Trazenje logicki narednog sloga:
- modifikovani simetricni postupak prolaska kroz B-stablo sa naizmenicnim pristupanjem listovima i njihovim nadredjenim elementima.
- vrsi se od tekuceg elementa stabla - inicijalno, tekuci element stabla je element sa najmanjom vrednoscu kljuca u krajnjem levom listu
- uporedjuju se argument trazenja **a** i vrednosti kljuca elemenata stabla k_e - trazenje se uspesno zavrsava kada je $a = k_e$, neuspesno trazenje se zavrsava nailaskom na element sa $k_e > a$ ili nailaskom na kraj krajnjeg desnog lista
Trazenje slucajno odabranog sloga:
- zapocinje u korenu stabla a eventualno se nastavlja u podredjenim cvorovima.Zavrsava se u cvoru u kojem je element pronadjen ili u listu.Na svakom nivou hijerarhije stabla pristupa se najvise jednom cvoru
- uporedjuju se argument trazenja **a** i vrednosti kljuca elemenata stabla k_e
- ako se pri trazenju nadje element za koji je $a = k_e$ onda se trazenom slogu pristupa na osnovu adrese Ae.

- ako se pri trazenju u cvoru dodje do elementa sa $k_e > a$, onda se trazenje nastavlja u odgovarajucem podstablu

Uspesno trazenje zavrsava u bilo kojem cvoru:

- ako imamo samo jedan bafer u OM za stablo pristupa onda je broj pristupa - Ru = h + 1
- ako imamo h bafera u OM za stablo pristupa onda je broj pristupa - 0 <= Ru <= h + 1

Neuspesno trazenje zavrsava se uvek u listu:

- ako imamo samo jedan bafer u OM za stablo pristupa onda je broj pristupa - Rn = h
- ako imamo h bafera u OM za stablo pristupa onda je broj pristupa - 0 <= Rn <= h

Ako je celo stablo u OM onda je Ru = Rn = 0.

Obrada datoteke s B-stablom:

Moze se koristiti i kao vodeca i kao obradjivana, i u rezimu redosledne i u rezimu direktne obrade. U svim slucajevima, pokazuje solidne performanse.

Parametri koji uticu na ukupan broj pristupa datoteci Ruk na v su:

1. da li je stablo kompletno ili poluprazno - broj cvorova kompletnog stabla je najmanji moguci a polupraznog je najveci moguci
2. da li je rezervisano h bafera za ceo pristupni put ili samo 1 bafer za cvorove B-stabla:
   - u slucaju h bafera, ceo pristupni put staje u OM pa se svakom cvoru stabla pristupa tacno jedanput
   - u slucaju jednog bafera , svakom neterminalnom cvoru mora se pristupati barem onoliko puta koliko elemenata poseduje
3. da li su sukcesivno trazeni slogovi u primarnoj zoni uvek smesteni u fizicki susedne lokacije ili uvek u lokacije razlicitih blokova

Ukupan broj pristupa datoteci u ulozi vodece datoteke:

$$\left\lceil \frac{N}{2r} \right\rceil + \left\lceil \frac{N}{f} \right\rceil \leq R_{uk}^{v} \leq \left\lfloor \frac{N+1}{r+1} \right\rfloor + \left\lfloor \frac{N-r}{r+1} \right\rfloor + N$$

Redosledna obrada - putem vodece datoteke sa Nv = Nv na u + Nv na n slogova.

Ukupan broj pristupa datoteci kao obradjivanoj :

$$\left\lceil \frac{N}{2r} \right\rceil + \left\lceil \frac{N_v^u}{f} \right\rceil \le R_{uk}^r \le \left\lfloor \frac{N+1}{r+1} \right\rfloor + \left\lfloor \frac{N-r}{r+1} \right\rfloor + N_v^u$$

Ocekivani broj pristupa pri uspesnom ili neuspesnom trazenju jednog logicki narednog sloga je R (nadvuceno) = Ruk na r / Nv

Direktna obrada - putem vodece datoteke sa Nv = Nv na u + Nv na n slogova. Relativno brz pristup slucajno odabranom slogu. Ukupan broj pristupa:

$$R_{uk}^d = R_u N_v^u + R_n N_v^n$$

Azuriranje datoteke s B-stablom:

-Upis novog i brisanje postojeceg sloga - vrsi se u rezimu direktne obrade, izvodi ga metoda pristupa na zahtev aplikativnog programa.Vrsi se upis, odnosno logicko brisanje elementa iz primarne zone i odgovarajuceg elementa iz stabla pristupa.
-Upis novog elementa u B-stablo na isti nacin kao u slucaju formiranja B-stabla.
-Brisanje elementa iz B-stabla - prethodi mu uspesno trazenje.Logicko brisanje sloga u primarnoj zoni. Fizicko brisanje elementa iz B-stabla. Element se sme fizicki izbrisati jedino ako se nalazi u listu.
(A) element za brisanje nalazi se u listu:
- (A1) list sadrzi m_e > r elemenata ili je koren stabla - fizicko oslobadjanje lokacije izbrisanog elementa, po potrebi pomeranje ostalih elemenata u listu za jednu poziciju ulevo, nakon brisanja u listu ostaje m_e - 1 >= r elemenata.
- (A2) list sadrzi m_e = r elemenata i nije koren stabla - fizicko oslobadjanje lokacije izbrisanog elementa nije dozvoljeno jer svaki cvor osim korena mora imati minimalno r elemenata.

- (A21) postoji barem jedan susedni cvor sa m_f > r elemenata - primenjujemo tehniku pozajmljivanja elemenata iz susednog cvora

- (A22) svi susedni cvorovi imaju m_f = r elemenata - primenjujemo tehniku spajanja dva cvora u jedan

Susedni cvorovi - cvorovi Ce i Cf su susedni ako imaju zajednicki direktno nadredjeni cvor C i ako vazi :

$(\exists!k \in K(C))(k_m < k < k_1)$

K(C) - skup vrednosti kljuca u cvoru C, k_m - najveca vrednost kljuca u cvoru Ce, k1 - najmanja vrednost kljuca u cvoru Cf

Tehnika pozajmljivanja elemenata od suseda:

- Formira se u OM uredjeni niz od $m_f + r - 1 + 1$ elemenata - svi elementi susednog cvora , jedan nadredjeni i r - 1 element cvora iz kojeg se brise element.
- $\lfloor (m_f + r) / 2 \rfloor$ elemenata se smesta u levi susedni cvor
- $\lfloor (m_f + r) / 2 \rfloor + 1$-vi element se smesta u nadredjeni cvor
- preostali elementi se smestaju u desni susedni cvor

Tehnika spajanja dva susedna cvora u jedan:

- Formira se u OM uredjeni niz od r + r - 1 + 1 elemenata - r elemenata susednog cvora, jedan nadredjeni i r - 1 element cvora iz kojeg se brise element.Svih 2r elemenata smesta se u levi susedni cvor.Desni susedni cvor postaje prazan odnosno dealocira se.Na kraju fizicki se brise element iz nadredjenog cvora. Ovo brisanje moze izazvati novo pozajmljivanje ili spajanje cvorova na visem nivou u stablu.U ekstremnom slucaju moze doci do spajanja jedina dva direktno podredjena cvora korenu - ovo izaziva dealociranje korena i smanjenje visine stabla za jedan.

(B) element za brisanje ne nalazi se u listu:

- zamena elementom koji sadrzi prvu vecu vrednost kljuca. Taj element predstavlja krajnji levi element u krajnjem levom cvoru desnog podstabla u odnosu na element koji se brise.Nakon zamene (upisivanja u lokaciju izbrisanog elementa) element s prvom vecom vrednoscu kljuca fizicki se brise iz lista.Povratak na slucaj (A) - primena za fizicko brisanje elementa s prvom vecom vrednoscu kljuca

Ocena karakteristika datoteka sa B-stablom

Prednosti:

- pokazuju uravnotezene performanse pri direktnoj i redoslednoj obradi
- ne kvare se performanse obrade u vremenu - kao posledica naknadnih azuriranja
- u odnosu na indeks-sekvencijalne datoteke ne zahtevaju postojanje zone

prekoracenja (struktura primarne zone odgovara strukturi serijske datoteke), maksimalni broj pristupa pri trazenju slucajno odabranog sloga moguce je unapred proceniti, selektivnost indeksa po svakom pristupnom putu je ista
Nedostaci:
- nije najpogodnije resenje za redoslednu obradu - broj pristupa datoteci zavisi od broja slogova vodece datoteke, primena modifikovanog algoritma simetricnog postupka prolaska kroz stablo pri trazenju adrese lokacije logicki narednog sloga - potreba visestrukog pristupanja neterminalnim cvorovima
- deljenje cvorova favorizuje izgradnju polupraznog B-stabla - kada se formiranje datoteke vrsi na osnovu sekvencijalne ulazne datoteke, svi listovi osim krajnjeg desnog uvek su poluprazni, cesta situacija u praksi kada se vrednosti kljuca generisu automatski, inkrementiranjem

B*-stablo, B#- stablo, B+ - stablo

B* - stablo

Strukturalno isto kao osnovno B-stablo.
Tehnika prelivanja:
- ublazavanje problema favorizacije polupraznog stabla
- primenjuje se pri upisu novog elementa u stablo kada je cvor Ce u koji se upisuje element kompletan a bar jedan od njegovih susednih cvorova Cf sadrzi mf < 2r elemenata
- formira se u OM uredjeni niz od mf + 2r + 1 + 1 elemenata - svi elementi susednog cvora, jedan nadredjeni i 2r+1 elemenata cvora u koji se upisuje element, $\lfloor (mf + 2r + 2) / 2 \rfloor$ elemenata smesta se u levi susedni cvor, $\lfloor (mf + 2r + 2) / 2 \rfloor$ + 1-vi element smesta se u nadredjeni cvor a preostali elementi se smestaju u desni susedni cvor
- favorizacija 75% popunjenosti stabla

B# - stablo

To je varijanta B* - stabla, garantuje se minimalna popunjenost 66% - za sve cvorove na svim nivoima hijerarhije, osim na prva 2.

Tehnika distribuiranog deljenja:
Primenjuje se pri upisu novog elementa u stablo kada je cvor Ce u koji se upisuje element kompletan i kompletni su svi njegovi susedi, onda se elementi susednih cvorova Ce i Cf distribuiraju u tri cvora. Formira se u OM uredjeni niz od 2r+2r+1+1 elemenata:
- 2r elemenata susednog cvora , jedan nadredjeni i 2r+1 elemenata cvora u koji se upisuje element
- $\lfloor (4r + 2) / 3 \rfloor$ elemenata smesta se u levi susedni cvor
- $\lfloor (4r + 2) / 3 \rfloor$ + 1-vi element smesta se u nadredjeni cvor
- $\lfloor (4r + 1) / 3 \rfloor$ elemenata se smesta u srednji susedni cvor
- $\lfloor (4r + 2) / 3 \rfloor + \lfloor (4r + 1) / 3 \rfloor$ + 2-gi element smesta se u nadredjeni cvor
- preostalih $\lfloor 4r / 3 \rfloor$ elemenata smesta se u novoalocirani , desni susedni cvor

<u>B+ stablo</u>
Modifikacija osnovnog B-stabla - prevazilazenje glavnog nedostatka osnovnog B-stabla , ili neke od njegovih varijanti, pri trazenju logicki narednog sloga potrebno je pristupati svim cvorovima stabla.
U slucaju B+ stabla , za trazenje logicki narednog sloga dovoljno je pristupati samo listovima.
Strukturalno, drugacije je od osnovnog B-stabla.Vrednosti kljuca svih slogova nalaze se uredjene u listovima.Svi listovi su dvostruko spregnuti (sadrze informaciju o svim logickim vezama izmedju slogova).Cvorovi na visim nivoima stabla trazenja sadrze najmanje vrednosti kljuca iz svakog lista osim iz krajnjeg levog.Vrednosti kljuca cvorova koji ne predstavljaju listove , u nadredjenim cvorovima ne ponavljaju se. Logika formiranja neterminalnih cvorova odgovara strukturi osnovnog B-stabla.
Format cvora B+ stabla:
- niz elemenata, gde je svaki element dvojka (ke,Pe), e $\in$ {1,...,m}, ke - vrednost kljuca sloga Si, (i $\in$ {1,2,...,N}) , Pe - pokazivac ka podstablu , za neterminalni cvor ili u bloku u primarnoj zoni u slucaju lista.

Zaglavlje bloka	P_0	k_1	P_1	k_2	P_2	...	k_m	P_m	Neiskorišćeni prostor

Uslovi stabla trazenja:

- $(\forall i \in \{1,..., m - 1\})(k_i < k_{i+1})$
- $(\forall k \in K(P_0))(k < k_1)$
- $(\forall i \in \{1,..., m - 1\})(\forall k \in K(P_i))(k_i \leq k < k_{i+1})$
- $(\forall k \in K(P_m))(k_m \leq k)$

Formiranje B+ stabla:

Analogan postupak formiranju osnovnog B-stabla, ili neke njegove varijante (B* ili B# stablo) - mogu se koristiti postupci prelivanja , obicnog ili distribuiranog deljenja na uobicajen nacin.

Tehnika deljenja lista - specificnosti:

Formira se u OM uredjeni niz od 2r + 1 elemenata - svi elementi datog cvora i jedan novododati, $\lfloor (2r + 1) / 2 \rfloor$ = r elemenata smesta se u levi susedni cvor , r+1-vi element smesta se u nadredjeni cvor, r+1-vi element i svih preostalih r elemenata smesta se u novoalocirani , desni susedni cvor (ponavljanje istog elementa i u nadredjenom cvoru i u desnom susedu)

Neterminalni cvorovi dele se na isti nacin kao kod B-stabla.

Tehnika prelivanja u listovima - specificnosti:

Formira se u OM uredjeni niz od mf + 2r + 1 + 1 elemenata - svi elementi susednog cvora , jedan nadredjeni i 2r + 1 elemenata cvora u koji se upisuje element, $\lfloor (mf + 2r + 2) / 2 \rfloor$ elemenata smesta se u levi susedni cvor, $\lfloor (mf + 2r + 2) / 2 \rfloor$ + 1-vi element smesta se u nadredjeni cvor, $\lfloor (mf + 2r + 2) / 2 \rfloor$ + 1-vi element i svi preostali elementi smestaju se u desni susedni cvor.

Prelivanje u neterminalnim cvorovima vrsi se na isti nacin kao kod osnovnog B stabla.

<u>Trazenje u B+ stablu:</u>

Trazenje logicki narednog sloga - vrsi se iskljucivo u listovima B+ stabla primenom kombinacije metoda linearnog trazenja i metode pracenja pokazivaca.Vrsi se od tekuceg elementa B+ stabla.

Redosledna obrada - prolazak kroz sve listove B+ stabla i pristupanje primarnoj zoni.

Trazenje slucajno odabranog sloga - pocinje u korenu B+ stabla i uvek se zavrsava

u jednom od listova i za uspesno i za neuspesno trazenje.Ako je a < ke, dalje se prati pokazivac Pe-1, ako je a = ke, prati se pokazivac Pe - stize se do lista, a ako je a > km prati se pokazivac Pm.
Broj pristupa kod uspesnog ako imamo samo 1 bafer u OM za stablo pristupa - Ru = h na + + 1
Broj pristupa kod neuspesnog ako imamo samo 1 bafer u OM za stablo pristupa - Rn = h na +
Azuriranje B+ stabla:
Upis novog elementa u B+ stablo vrsi se na isti nacin kao pri formiranju B+ stabla. Kod brisanja, brise se i element u listu i ako je ponovljen, element sa istom vrednoscu kljuca u nadredjenom cvoru. Brisanje elementa iz cvora sa r elemenata - tehnika pozajmljivanja, tehnika spajanja cvorova.

Indeks-sekvencijalna datoteka sa B-stablom

Modifikacija B+ stabla - prevazilazenje glavnog nedostatka B+ stabla ili neke od njegovih varijanti. Za sva uspesna trazenja sloga, potrebno je pristupati primarnoj zoni.U slucaju IS datoteke sa B - stablom, uspesna trazenja zavrsavaju u listovima. Strukturalno je drugacija od B+ stabla - u listovima B+ stabla ne memorisu se parovi (ke,Pe), vec kompletni slogovi.Pogodna je u situacijama kada kapacitet sloga nije preveliki kako bi se ocuvao solidan rang stabla.Listovi B+ stabla postaju blokovi primarne zone.
Primarna zona - spregnuta struktura, slogovi su uredjeni saglasno rastucim vrednostima kljuca.
Zona indeksa - spregnuta struktura = neterminalni cvorovi B+ stabla.Osnovno B-stablo ili neka od varijanti (B*, B#).Retko popunjeni indeks - u zonu indeksa propagiraju se najmanje vrednosti kljuca svakog bloka primarne zone, osim prvog bloka.
Primarna zona - faktor blokiranja f = 2r.
Primarna zona - broj blokova:
- kompletno stablo - Bmin = $\lceil N/2r \rceil = \lceil N/f \rceil$
- poluprazno stablo - Bmax = $\lceil N/r \rceil = \lceil 2N/f \rceil$

Broj blokova se krece u rasponu Bmin <= B <= Bmax

Formiranje na uobicajen nacin.Moze se poci od ulazne serijske ili ulazne sekvencijalne datoteke.

Obrada - moze se koristiti i kao vodeca i kao obradjivana i u rezimu direktne i u rezimu redosledne obrade.Pogodnija je za primenu u redoslednoj obradi. U redoslednoj obradi , pristupa se (svim) blokovima primarne zone. Pogodno je da broj blokova primarne zone bude manji - za manje kapacitete slogova, postize se veci r i manji B.

Trazenje i azuriranje - postupci analogni onima koji se primenjuju u slucaju B+ stabla. Trazenje logicki narednog sloga vrsi se iskljucivo u blokovima primarne zone primenom kombinacije metoda linearnog trazenja i metoda pracenja pokazivaca.Vrsi se od tekuceg bloka primarne zone.

Trazenje slucajno odabranog sloga pocinje u korenu B-stabla i uvek se zavrsava u bloku primarne zone i za uspesno i za neuspesno trazenje. Broj pristupa kod uspesnog sa samo jednim baferom u OM za stablo pristupa je Ru = h na is + 1 a broj pristupa kod neuspesnog sa samo jednim baferom u OM za stablo pristupa je Rn = h na is + 1.

Ocena karakteristika indeksnih datoteka s B-stablima

Pogodne i za direktnu i za redoslednu obradu - relativno brz pristup slucajno odabranom slogu.Ne prevelik broj pristupa u redoslednoj obradi.Za efikasniju redoslednu obradu koriste se indeks-sekvencijalne datoteke s B-stablom u slucajevima ne prevelikog kapaciteta sloga.

Kompromisno resenje pri izboru fizicke strukture podataka - nije idealna organizacija ni za redoslednu ni za direktnu obradu ali organizacija je prisutna u svim savremenim SUBP bez izuzetka.