Baze podataka 1 – skripta

1. Osnovni pojmovi

Realni sistem (RS) je sve što nas okružuje i što možemo da percipiramo kao realno.

Sistem je struktuirani skup objekata utvrđenog stanja i ponašanja koji se nalaze u međusobnoj interakciji kako bi ostvarili zadate ciljeve. Čine ga cilj delovanja, resursi, procesi, struktura i okruženje.

Informacioni sistem (IS) je model realnog sistema. Cilj njegove izgradnje je pružanje informacija koje su neophodne za funkcionisanje i upravljanje realnim sistemom. On je infrastrukturna komponenta realnog sistema i dominantno je softverski proizvod. Zadaci IS su obuhvat i akvizicija podataka, skladištenje podataka, prenos podataka, obrada i prezentacija podataka, automatizacija upravljačkih funkcija RS. Činioci IS su: računarsko-komunikaciona infrastruktura, baza podataka, aplikacije za rad sa podacima, projektna i korisnička dokumentacija, krajnji korisnici, tim za obezbeđenje eksploatacije i održavanja. Primeri IS: bankovne aplikacije, sistemi u bolnicama, društvene mreže...

(Realni) Entitet je činilac (resurs) poslovanja u realnom sistemu.

Klasa realnih entiteta je skup realnih entiteta koji poseduju zajedničko svojstvo, formalno $E = \{e_i \mid P(e_i)\}$

Poveznik (veza) reprezentuje odnos između dva ili više realnih entiteta ili već uspostavljenih poveznika.

Klasa poveznika je skup veza između klasa realnih entiteta ili prethodno identifikovanih klasa poveznika, tj. skup poveznika koji poseduju zajedničko svojsvo, formalno $S = \{(e_1, ..., e_m) \mid P(e_1, ..., e_m)\}$

$P(e_i)$, $P(e_i, ..., e_m)$ je svojstvo klase entiteta/poveznika, iskazuje osobine klase E ili S

Obeležje (atribut) je svojstvo klase realnih entiteta/poveznika, proističe iz semantike predikata $P(e_i)$, npr. JMBG, Ime, Prz

(Profesorka rekla da nije toliko bitno Vrste obeležja prema mogućnosti dekomponovanja na celine nižeg reda:

- Elementarno – ne dekomponuje se, predstavlja elementarnu vrednost, npr. Grad, Ulica, Broj, Stan

- Složeno – može se dekomponovati, predstavlja složenu vrednost, npr. Adresa=(Grad, Ulica, Broj, Stan)
- Skupovno – predstavlja skup vrednosti istog tipa

Domen je specifikacija skupa mogućih vrednosti obeležja sa definisanim dozvoljenim relacijama i operacijama nad datim skupom, predstavlja skup mogućih vrednosti obeležja. Prema načinu nastanka može biti predefinisani (primitivni) i korisnički definisani (izvedeni). Svakom obeležju se obavezno pridružuje domen. Ako imamo obeležje A, Dom(A) predstavlja pridruživanje domena tom obeležju (govori o deklaraciji domena – apstraktno pravilo), a dom(A) predstavlja navođenje mogućih vrednosti koje obeležje može da ima (govori o konkretnom skupu vrednosti koji se iz tog domena dobija).

Podatak je uređena četvorka (Entitet, Obeležje, Vreme, Vrednost). Semantičku komponentu podatka čine (Entitet, Obeležje, Vreme), što znači da ako navedemo samo vrednost, a izostavimo entitet, obeležje ili vrednost to onda nije podatak jer ne znamo kontekst. Vreme kao komponenta se može izostaviti ako se uvede konvencija da se podatak odnosi na vreme u kojem se manipuliše podatkom ili ako se vreme identifikuje kao posebno obeležje.

Tip entiteta (TE) je model klase realnih entiteta koji poseduje naziv N i skup obeležja Q = $\{A_1$,..., $A_n\}$ koji predstavlja podskup skupa obeležja klase realnih entiteta, npr. Radnik({Mbr, Ime, Prz, Zan, JMBG}). **Pojava tipa entiteta** je model jednog realnog entiteta u IS, formalno p(N) = $\{(A_1, a_1),...,(A_n, a_n)\}$, pri čemu za svaki A_i koje pripada Q važi da a_i pripada dom(A_i). Ako se u skup obeležja uvede redosled, tada se p(N) može posmatrati kao torka (a_1,..., a_n). Tip entiteta formalno predstavlja skup pojava tipa entiteta SP(N) = $\{p_i \mid P(p_i)\}$, pri čemu svaka pojava predstavlja tačno jedan realni entitet.

(Obavezno na ispitu) **Identifikator TE** je skup obeležja koji ima ulogu da obezbedi način za jedinstvenu identifikaciju bilo koje pojave tipa entiteta. **Identifikator pojave tipa entiteta** je bilo koja vrednost identifikatora TE označava najviše jednu pojavu tipa entiteta i predstavlja jednu od četri komponente podatka. Identifikator TE može biti eksterni (ne predstavlja podskup skupa obeležja tog TE, npr za TE Radnik RBR_pojave_TE) i interni (predstavlja podskup skupa obeležja tog TE, npr za TE Radnik JMBG).

(Obavezno na ispitu) **Ključ TE** je minimalni interni identifikator TE, formalno imamo skup obeležja Q = $\{A_1,..., A_n\}$ tipa entiteta N i X je podskup od Q takav da važe:

- Svojstvo jednoznačne identifikacije – ne postoje dve pojave TE N sa istom x-vrednošću (za X) i svaka pojava TE mora imati zadatu x-vrednost
- Svojstvo minimalnosti – ne postoji pravi podskup skupa X za koji važi svojstvo jednoznačne identifikacije

Svaki TE poseduje bar jedan ključ. TE predstavlja uređenu strukturu N(Q, C), pri čemu je N naziv TE, Q skup obeležja TE, C skup ograničenja TE i K skup ključeva TE (on je podskup od C i zbog toga C nikad nije prazan skup).

Primarni ključ je jedan izabrani ključ iz skupa ekvivalentnih ključeva TE, često se označava podvlačenjem.

Tip poveznika (TP) povezuje dva ili više TE ili prethodno definisane tipove poveznika tj. to je model veza između realnih entiteta ili veza, predstavlja uređenu strukturu $N(N_1, N_2,..., N_m, Q, C)$, gde je N naziv TP, $N_1, N_2,..., N_m$ su povezani tipovi (u ovom skupu može više puta da se pojavi isti povezani tip), Q je skup obeležja TP (može biti prazan skup), C je skup ograničenja TP i K je skup ključeva TP (on je podskup od C i zbog toga C nikad nije prazan skup), npr. Sluša(Student, Predmet, {Semestar}, C1), formalno predstavlja skup pojava poveznika $SP(N) = \{(p_1,..., p_m) \mid P(p_1,..., p_m)\}$ gde je p_i jedna pojava TE ili TP. **Pojava tipa poveznika** $N(N_1, N_2,..., N_m, \{B_1,..., B_k\}, C)$ reprezentuje jedan poveznik u realnom sistemu i predstavlja skup podataka $p(N) = (p_1,..., p_m)(N) = \{(B_1, b_1),..., (B_k, b_k)\}$ pri čemu svako obeležje mora imati vrednost iz domena koji je pridružen tom obeležju i skup svih pojava mora zadovoljiti skup ograničenja C.

(Obavezno na ispitu) **Identifikator TP** je niz $(N_1, N_2,..., N_m)$ ili neki njegov neprazan podniz i ima ulogu da obezbedi način za jedinstvenu identifikaciju bilo koje pojave TP. Identifikator pojave TP je niz $(p_1,..., p_m)$ i označava najviše jednu pojavu TP i niz pojava tipova $(p_1,..., p_m)$ ili jeste ili nije u vezi.

(Obavezno na ispitu) Ključ TP je skup obeležja X izveden na osnovu ključeva povezanih tipova $(N_1, N_2,..., N_m)$, najčešće podskup unije ključeva povezanih tipova. $X = \{A_1,..., A_n\}$ je takav da važe:

- Svojstvo jedinstvene identifikacije – ne postoje dve pojave TP N sa istom x-vrednošću (za X)
- Svojstvo minimalnosti – ne postoji pravi podskup skupa X za koji važi svojstvo jedinstvene identifikacije

Struktura podataka je orijentisani graf $G(V, \rho)$ gde je V skup čvorova, pri čemu svaki čvor reprezentuje podatke i svakom čvoru je pridružena semantika i ρ je skup grana, pri čemu svaka grana reprezentuje veze između podataka i svakoj vezi je pridružena semantika. **Vrste struktura podataka prema nivou apstrakcije pridružene semantike** su logička struktura obeležja, logička struktura podataka i fizička struktura podataka. **Vrste struktura podataka prema broju direktnih prethodnika i sledbenika čvora** su linearna struktura podataka, struktura tipa stabla i ciklična struktura.

Logička struktura obeležja (LSO) je struktura nad skupom tipova entiteta, tipova poveznika i njihovih atributa, predstavlja model dela realnog sistema M = (STE, RTE), pri čemu je STE skup

tipova, a RTE relacija koja STE snadbeva strukturom i koja modelira odnose koji postoje između realnih entiteta istih ili različitih klasa. Ona je najviši nivo apstrakcije. Postoje dva pristupa organizaciji LSO:

- I TE i TP su čvorovi - tako mi radimo, STE sadrži skup svih TE i TP sistema, RTE sadrži grane koje prikazuju veze TP sa povezanim tipovima, simboli za TE i TP mogu a ne moraju biti različiti
- TE su čvorovi, a TP su grane – STE sadrži skup svih TE sistema, RTE sadrži grane koje prikazuju sve TP i veze sa njihovim povezanim tipovima, nastaju problemi jer onda TP ne sme da ima skup obeležja i skup ograničenja i jer TP ne može za povezani tip da ima neki drugi TP

Nivo detaljnosti prikaza LSO: globalni prikaz (nivo TE i Tp) i detaljni prikaz (nivo obeležja).

Logička struktura podataka (LSP) se definiše putem posebne relacije nad skupom podataka u granicama zadate LSO – LSO predstavlja kontekst za LSP, šema za LSP.

Tip sloga je linearna struktura skupa obeležja datog TE. **Slog (N-torka)** je linearna struktura nad skupom podataka jednog entiteta. **Pojava TE** je kontekstna LSO, predstavlja isto što i slog.

Datoteka je kontekstna LSO, predstavlja strukturu podataka nad skupom pojava jednog TE, skup slogova kojima kontekst daje tip sloga.

Šema baze podataka – struktura nad skupom TE i TP. **Baza podataka** je kontekstna LSO, predstavlja skup pojava TE i TP nad TE i TP.

LSP se vizuelno i fizički mogu predstavljati pomoću grafova i pomoću tabela.

Fizička struktura podataka (FSP) je LSP smeštena na memorijski medijum, uključuje podatke o načinu smeštanja LSP na medijum.

2. Koncepcija BP

Istorijski, IS su se čuvali na papirima, bez podrške računara. Nakon toga, IS su organizovani nad sistemima datoteka i sačinjavao ga je skup nezavisnih aplikacija, gde je svaka aplikacija imala svoje datoteke. Dolazilo je do toga da podatke o istom entitetu imamo u više datoteka, i samim tim do nedostataka kao što su nepovezanost aplikacija (ručno prepisivanje istih ili sličnih podataka), redundantnost podataka (višestruko memorisanje istih ili sličnih podataka) i čvrste povezanosti između aplikacija i podataka (program vodi računa o fizičkoj strukturi datoteke). Sledeća ideja je da sve aplikacije koriste iste datoteke, što otklanja neke nedostatke, ali imamo probleme prava pristupa, višekorisnički rad i problem ako različite aplikacije zahtevaju različite organizacije datoteka.

Relacione baze podataka imaju neke osnovne ideje:

- Da se svi podaci jednog IS integrišu u jednu veliku “datoteku” - bazu podataka
- Neredudantno memorisanje podataka
- Uvođenje softvera za podršku kreiranja i korišćenja BP – sistem za upravljanje bazom podataka (SUBP)
- Transakcija obrada – ranije bila teška za implementaciju, sada to radi SUBP
- Višekorisnički konkurentni pristup
- Autorizacija korisnika

(Ne bude na ispitu) NoSql baze podataka – nastaju usled povećanog generisanja podataka, gde za neke od tih podataka relacione baze podataka nisu pogodne. Brže su od relacionih, ali nisu toliko sigurne.

Tipovi podataka prema strukturi: strukturirani podaci (postoji šema koja definiše format podataka), polustrukturirani podaci (može se definisati šema koja specificira moguće elemente strukture koji mogu ali i ne moraju da postoje, npr. HTML) i nestrukturirani podaci (ne postoji šema koja definiše format podataka, npr. email).

Sistem za upravljanje bazom podataka (SUBP) omogućava efikasno i pouzdano formiranje, korišćenje i menjanje BP, aplikacije koriste i ažuriraju podatke koristeći isključivo SUBP. On sadrži jezik za opis podataka (Data Definition Language - DDL), jezik za manipulisanje podacima (Data Manipulation Language - DML) i upitni jezik (Query Language - QL). Jezgo SUBP omogućava:

- Obezbeđivanje fizičke organizacije podataka
- Rutine za upravljanje podacima
- Zaštita od neovlašćenog pristupa i uništavanja
- Obezbeđenje višekorisničkog rada
- Obezbeđivanje distribuirane organizacije BP
- Obezbeđivanje zadavanja šeme BP

Šema BP je logička kategorija, program koji koristi usluge SUBP može da vidi samo nju i nad njom koristi LSP, ne sme da vodi računa o FSP - zadatak SUBP je da preslikava LSP u FSP. Uvođenjem šeme BP smanjuje se zavisnost programa i šeme BP od promena FSP, smanjuje se redudantnost i uvode se nove uloge – projektant baze podataka koji definiše šemu BP i administrator baze podataka koji upravlja aktivnostima SUBP, implementira i održava šemu BP.

Podšema BP (eksterna šema) je LSO, dobijena na osnovu dela šeme BP, potrebna i dovoljna za realizaciju jednog ili grupe sličnih transakcionih programa. Nastaje iz razloga što je šema BP često jako kompleksna i podložna promenama. Projektuje se u procesu razvoja IS. Poželjna je takva organizacija transakcionih programa da koriste BP isključivo preko šema. Poželjno je da SUBP

preslikava podšemu u šemu BP, ali je to često zadatak transakcionog programa. Uvođenjem podšeme dobijaju se i logička i fizička nezavisnost programa od podataka.

Pogled je LSP nad podšemom, slika dela BP kako je vidi programer ili korisnik. **Globalni pogled** je LSP nad šemom BP, slika stanja modelovanog sistema.

Sistem baza podataka u užem smislu obuhvata BP, SUBP, šemu BP, jezike i operacije za kreiranje, ažuriranje i korišćenje BP, sistemski softver i računare na kojima je BP kreirana. U širem smislu obuhvata više BP, više SUBP (homogen SUBP – više instanci istog SUBP, heterogen SUBP – više instanci različitih SUBP), sav prateći softver i hardver.

3. Model podataka

Model podataka (MP) je matematička apstrakcija pomoću koje se gradi šema BP i služi za predstavljanje šeme BP realnog sistema, ograničenja u odnosima između podataka o stanjima realnog sistema i dinamike izmena stanja realnog sistema. Čini ga trojka (S, I, O) gde je S strukturalna komponenta (omogućava modeliranje šeme BP), I integritetna komponenta (omogućava modeliranje ograničenja nad podacima u BP) i O operaciona komponenta (modelira dinamiku izmene stanja podataka u BP i šeme BP).

(Za visoku ocenu) **Nivoi apstrakcije** su određeni modelom podataka i mogu biti nivo intenzije (nivo konteksta, tipa, npr. LSO) i nivo ekstenzije (nivo konkretizacije, pojave tipa, npr. LSP). U oblasti modelovanja sistema postoje META leveli.

Strukturalna komponenta MP ima koncept i primitivan (atomični) koncept. Koncept je apstraktna predstava jedne klase pojmova. Primitivnom konceptu ne možemo dati definiciju, ne može se dekomponovati na koncepte nižeg reda. Strukturalna komponenta sadrži skup primitivnih koncepata (sa skupom datih osobina svakog koncepta, skupom pravila za njihovo korišćenje i opisanom mogućom semantikom), skup formalnih pravila za kreiranje složenih koncepata (omogućava proširivanje inicijalno definisanog MP) i skup unapred kreiranih složenih koncepata. Ako je MP dovoljno bogat semantički on već nudi neke složene koncepte.

Integritetnu komponentu MP čini:

- Skup tipova ograničenja sa skupom datih osobina svakog tipa ograničenja, skupom pravila za njihovo korišćenje i opisanom mogućom semantikom, pomoću njih se iskazuju ograničenja koja se odnose na moguće vrednosti obeležja ili moguće odnose između pojava povezanih tipova, npr. ograničenje ključa
- Skup formalnih pravila za izvođenje zaključaka o važenju ograničenja
- Skup formalnih pravila za kreiranje novih tipova ograničenja

Proveru važenja ograničenja možemo ugraditi u transakcione programe ili specifikaciju šeme BP sa implementacijom u SUBP, teži se ka tome da se sva ograničenja ugrade u šemu BP, a pojedina i u transakcione programe.

Operacijska komponenta MP omogućava modeliranje dinamike izmene stanja u sistemu BP i predstavlja skup tipova operacija sa skupom datih osobina svakog tipa operacije, skupom pravila za njihovo korišćenje i opisanom mogućom semantikom. Definiše:

- Jezik za definiciju podataka (Data Definition Language – DDL) - tipovi operacija za kreiranje i modifikaciju specifikacija šeme BP, fizičke strukture BP, prava pristupa i zaštite BP, novih tipova operacija za upravljanje podacima
- Jezik za manipulisanje podacima (Data Manipulation Language – DML) - tipovi operacija za izmenu stanja BP
- Upitni jezik (Query Language - QL) - tipovi operacija za iskazivanje upita nad BP

Specifikacija operacije sadrži aktivnost (specifikacija akcije nad podacima u BP) i selekciju (specifikacija dela BP ili dela šeme BP nad kojim se sprovodi ta aktivnost).

Operacijska komponenta može biti:

- Proceduralna (navigaciona) - definiše šta i kako treba da se uradi, selekcija vrši izbor jednog objekta iz BP, proceduralnost sa programskim petljama i uslovnim grananjima
- Specifikaciona (deklarativna) - definiše šta ali ne i kako treba da se uradi, selekcija vrši izbor iz skupa objekata iz BP, neproceduralnost

(Na ispitu može doći nabrajanje modela podataka, ništa više iz toga) **Modeli podataka** mogu biti: Mrežni, Hijerarhijski, ER, Relacioni, Logički i verovatnosni logički modeli, Objektno-orijentisani, XML model, Model ključ/vrednost, Model zasnovan na grafovima, Model zasnovan na dokumentima, Model zasnovan na familijama kolona, Model zasnovan na nizovima i matricama...

4. ER model

Osnovni pojmovi ER modela su obeležje i domen, tipe entiteta i pojava tipa entiteta, tip poveznika i pojava tipa poveznika. Semantički bogatiji od relacionog, viši nivo apstrakcije, prevodi se na relacioni. Ima dijagramsku reprezentaciju.

(od 5. do 16. slajda sve isto kao i u prezentaciji Osnovni pojmovi)

Strukturalnu komponentu ER modela čine primitivni koncepti:

- Vrednost – bilo koja konstanta iz bilo kog skupa

- Domen – specifikacija skupa mogućih vrednosti obeležja sa definisanim dozvoljenim relacijama i operacijama nad datim skupom, mogu biti predefinisani (ugrađen u definiciju modela podataka, zavisi od softverskog okruženja, npr N, R, integer...) i korisnički definisani (definiše se korišćenjem već postojećeg domena putem pravila za definisanje domena, može predstavljati skup atomičnih ili složenih podataka, npr DOCENA)
- Obeležje - ako na dijagramu vidimo više podvučenih obeležja za neki tip entiteta to znači da svi oni zajedno čine primarni ključ

Izvedeni koncepti sturkturalne komponente ER modela su podatak, tip entiteta, pojava tipa entiteta, tip poveznika, pojava tipa poveznika.

(od 24. do 40. slajda sve isto kao u prezentaciji Osnovni pojmovi)

ER dijagrami su tehnika za predstavljanje modela statičke strukture realnog sistema. Prikazuje tipove entiteta, tipove poveznika, njihove veze, obeležja i domene. Obeležja primarnog ključa se podvlače. Nivoi detaljnosti prikaza ER dijagrama su nivo naziva tipova i nivo naziva obeležja i domena. Možemo da imamo dva tipa poveznika između istih tipova entiteta, tipove poveznika koji povezuju n tipova entiteta (n-arni tipovi poveznika), rekurzivni binarni tip poveznika...

Integritetna komponenta opisuje ograničenja, tipovi ograničenja u ER modelu podataka su ograničenje domena, ograničenje vrednosti obeležja, ograničenje pojave tipa, kardinalitet tipa poveznika, ograničenje ključa (integritet tipa) za TE i TP

Domen predstavlja strukturu D(id(D), Predef), gde je D naziv domena, id(D) ograničenje domena i Predef predefinisana vrednost domena (nije obavezna).

Ograničenje domena se definiše primenom izabranog pravila za specificiranje korisnički definisanog domena. Pravila koja postoje su pravila nasleđivanja (mi radimo ovo), pravila tipa sloga, pravila tipa skupa, pravila tipa izbora. Ograničenje “nasleđenog” domena je struktura id(D) = (Tip, Dužina, Uslov), gde je:

- Tip - tip podatka (oznaka primitivnog domena ili prethodno korisnički definisanog domena), jedina obavezna komponenta specifikacije, nasleđuju se sva ograničenja, relacije i operacije definisane nad izabranim tipom
- Dužina - dužina tipa podataka, navodi se samo tipove podataka (primitivne domene) koji to zahtevaju
- Uslov - logički uslov koji svaka vrednost domena mora da zadovolji, u ER modelu podataka mora biti definisana sintaksa za zadavanje logičkih uslova
- Predef – predefinisana vrednost, ako se ne unese vrednost obeležja uzima se ona, mora da zadovolji ograničenje tipa, dužine i uslova

Interpretacija moguća za bilo koju vrednost – konstantu.

Oznaka Δ u primerima znači da nema tog uslova.

Nula (nedostajuća) vrednost je specijalna vrednost obeležja koja se označava sa ω (u praksi je to NULL) i označava da vrednost obeležja nije poznata, ne postoji ili nije informativna. Nekada postoji potreba da vrednost obeležja umesto vrednosti iz domena ima nula vrednost.

Specifikacija obeležja A $\in$ Q datog tipa N je struktura (id(N, A), Predef), gde je id(N, A) ograničenje vrednosti obeležja, a Predef je predefinisana vrednost obeležja.

Ograničenje vrednosti obeležja se definiše za svako obeležje tipa i predstavlja strukturu id(N, A) = (Domen, Null), gde je Domen oznaka pridruženog domena obeležja (svakom obeležju moramo pridružiti domen), a Null $\in$ {T, ⊥} pri čemu je T dozvola dodele nula vrednosti obeležju, a ⊥ je zabrana dodele nula vrednosti obeležju (specificiramo da li obeležje sme ili ne sme imati null vrednost). Domen i Null su obavezne komponente (SUBP sređuje za nas da ne moramo da pišemo null, samo not null), a može da se navede i Predef – ako se navede onda će on da važi, a ako se ne navede onda će da važi Predef od domena ili prvog sledećeg nasleđenog domena za koji je Predef definisan. Narušavanje ograničenja dovodi do pokušaja narušavanja konzistentnosti BP. Interpretacija moguća za bilo koju vrednost obeležja.

Ograničenje pojave tipa definiše ograničenja na moguće vrednosti podataka unutar iste pojave TE ili TP i predstavlja skup ograničenja vrednosti obeležja kome je pridodat logički uslov. Formalno, za tip N id(N) = ({id(N, A) | A $\in$ Q'}, Uslov), gde je:

- Q' prošireni skup obeležja tipa - za TE to je isto što i Q, a za TP to je Q $\cup$ K_p, gde je K_p skup obeležja primarnog ključa TP
- Uslov je logički uslov koji svaka pojava tipa mora da zadovolji, može da sadrži bilo koje obeležje iz Q', u ER modelu podataka mora biti definisana sintaksa za zadavanje logičkih uslova

Interpretacija moguća za bilo koju pojavu tipa nad skupom obeležja nad kojim je definisano.

Kardinalitet TP prema povezanom tipu predstavlja par (a, b), pri čemu a može imati vrednost 0 ili 1 (minimalni kardinalitet), a b može imati vrednost 1 ili N, N>=2 (maksimalni kardinalitet). Definiše se za svaki povezani tip i ograničava u koliko pojava TP može učestvovati jedna, bilo koja pojava povezanog tipa – minimalno a i maksimalno b.

(Naučiti kako se formalno čitaju dijagrami kao na prezentacijama, dolazi na test)

Postoje tri opšte grupe maksimalnih kardinaliteta: M:N, N:1, 1:1. Ovo utiče na to kako će biti formiran ključ TP.

Integritet TP – niz naziva povezanih tipova ili njegov neprazan podskup, ograničenje ključa. Ključevi povezanih tipova ulaze u skup obeležja TP.

(Na testu sigurno) Kada imamo M:N uniraju se ključevi (+), kada imamo N:1 ključ iz strane N se propagira u stranu 1 - ključ TP je je ključ iz strane 1, kada imamo 1:1 nije uniranje (!!!) nego su oba ključevi, ali zasebno (,).

(Često na testu rekurzivni TP sa više obeležja kao ključevi).

Gerund (mešoviti tip) je TE dobijen transformacijom TP tj. dobija se kada TP učestvuje kao povezani tip u nekom drugom TP. On je istovremeno i TP za neke povezane tipove a i TE u nekim drugim TP. Koristimo ga kada:

- Ne mogu proizvoljne kobinacije pojava nekih tipova biti sadržane u pojavi posmatranog TP
- Postoji pravilo koje kombinacije pojava tih tipova mogu biti sadržane u pojavi posmatranog TP

(Pogledati primere na slajdovima)

(Neće biti u zadacima) Agregacija obezbeđuje objedinjavanje složenijih ER struktura gde se cela ER struktura posmatra kao jedan TE, koristi se često kad je šema složena a neki deo nje nam ne treba. Najjednostavniji primer agregacije je gerund.

Slabi TE je TE čije su pojave zavisne od pojava nekog drugog TE. Postoje dve vrste zavisnosti slabih TE:

- Egzistencijalna zavisnost - između pojava dva TE, postoji kad je minimalni kardinalitet tipa poveznika jednak 1, nema poseban simbol, regularni TE je onaj koji nije u egzistencijalnoj zavisnosti
- Identifikaciona zavisnost – poseban slučaj egzistencijalne zavisnosti, i minimalni i maksimalni kardinalitet su 1, uvodi klasifikaciju tipova na neidentifikacioni TE i identifikacioni TE (predstavlja identifikacionu zavisnost slabog TE, ukazuje da se svaka pojava zavisnog TE može identifikovati samo uz pomoć identifikatora nadređenog TE, ključ se formira kao ključ nadređenog+njegovi ključevi)

Regularni TE može učestvovati kao Id-zavistan povezani tip u nekom drugom TP. Id-zavisni TE može učestvovati i kao Id-zavistan i kao regularan TE u više različitih TP.

IS-A hijerarhija je poseban kocept koji pripada EER (proširenom) ER modelu gde određeni podskup neke klase određuju još neka obeležja. Uvode se pojmovi superklasa (nadtip) i potklasa (podtip) koji predstavljaju posebne vrste tipova. To su pojmovi vezani za postupak specijalizacije tj. generalizacije koja se primenjuje kada neki skup entiteta ili poveznika (superklasa) poseduje prepoznatljive podskupove (potklase) sa samo sebi svojstvenim obeležjima ili sa samo sebi svojstvenim vezama ka drugim klasama entiteta ili poveznika i svojstvena je semantičkim modelima podataka. Pojmovi superklase i potklase se uvode:

- Da bi model statičke strukture realnog sistema bio semantički bogatiji
- Da bi se izbegle nula vrednosti u ekstenziji (ako bi modelovali sa jednim TE imali bismo još puno dodatnih specifičnih obeležja kod kojih bi na dosta mesta bilo NULL)
- Da bi se izbeglo definisanje TP koje nema mnogo smisla

Specijalizacija se vrši na osnovu vrednosti nekog skupa klasifikacionih obeležja. U TE superklase ostaju primarni ključ i sva zajednička obeležja, dok u TE potklase idu samo svojstvena, specifična obeležja.

Navođenje kardinaliteta ka superklasi je obavezno (a, b), dok ka potklasama nije (uvek je (1, 1)). Ako je a 0 onda imamo totalnu IS-A hijerarhiju, a ako je 0 onda imamo parcijalnu. Ako je b 1 onda imamo nepresečnu IS-A hijerarhiju, a ako je N onda imamo presečnu.

Bitne karakteristike:

- Potklasa nasleđuje osobine superklase
- Ključ svake potklase je primarni ključ superklase (nasleđivanje ključeva), potklase mogu imati i svoje sopstvene ključeve (ne unira se sa ključem superklase)
- Identifikaciona zavisnost svake potklase prema superklasi
- Potklasa može imati ulogu superklase u nekoj drugoj IS-A hijerarhiji
- Nad jednim tipom se može napraviti više različitih IS-A hijerarhija koristeći različite kriterijume

Kategorizacija je poseban koncept, TP u EER modelu koji je veznan za postupak klasifikacije (tipizacije), svojstvene semantičkim modelima podataka. Uvodi se pojam kategorije – poseban tip (TE ili TP - gerunda) gde se jedan TE povezuje sa više kategorija (barem dve) i svaka pojava posmatranog TE pripada najviše jednoj kategoriji. Ne postoji id-zavisnost posmatranog TE od kategorija ili obrnuto (međusobno su nezavisni). Može, a ne mora postojati skup klasifikacionih obeležja kategorije. Minimalni kardinalitet ka posmatranom TE definiše tip kategorizacije – ako je 0 onda je parcijalna kategorizacija, a ako je 1 onda je totalna kategorizacija. Maksimalni kardinalitet ka posmatranom TE je uvek 1.

(Na testu može biti da damo npr. primer parcijalne-nepresečne kategorizacije)

N-arni TP je poveznik koji povezuje više od dva druga tipa. Za svaki od n povezanih tipova, za bilo koju odabranu pojavu tipa utvrđuje se koliko se minimalno i koliko se maksimalno puta javlja kao komponenta u pojavama TP.

(Ne moramo znati varijante u dijagramskom označavanju)

5. Relacioni model podataka

(od 1. do 6. slajda ponavljanje sa ranijih prezentacija)

Nastaje zbog želje da se uklone nedostaci klasičnih modela podataka - čvrsta povezanost logičkih i fizičkih aspekata, strukturalna kompleksnost i navigacioni jezik i zbog institiranja na matematičkim osnovama u razvoju MP – teorija skupova i relacija, predikatski račun I reda.

(Ne udubljujemo se u mrežni i hijerarhijski model)

Nezavisnost programa od podataka – potpuno odvajanje prezentacionog od formata memorisanja, relacija kao skup n-torki, apstraktni opis relacije je šema relacije N(R, C) gde je R skup obeležja, a C skup ograničenja, pri čemu je skup ključeva K podskup od C. Često se u početnim fazama projektovanja šema relacije posmatra kao N(R, K).

Strukturalna jednostavnost:

Relacija je osnovna logička stuktura u relacionom modelu podataka i ona ne sadrži podatke o fizičkoj organizaciji, zasniva se na matematičkom konceptu relacije (skupa torki), ima homogenu i uniformnu strukturu (redovi i kolone) i najčešće se prikazuje u vidu tabele što je čini lako razumljivom.

Šema relacije- opis tabele (npr. imena kolona, tipovi podataka)

Relacija- konkretni podaci u tabeli (redovi)

Za selekciju podataka, relacioni modeli koriste asocijativno adresiranje (preko sadržaja, ne lokacije). Zbog toga, dolazi do korišćenja simboličkih adresa (vrednost ključa), svaki podatak u BP se pronalazi na osnovu naziva relacije, zadatih obeležja i vrednosti ključa itd...

Za povezivanje podataka, relacioni modeli koriste simboličke adrese, rešenje putem prostiranja ključa(uvodi se pojam stranog ključa) i rešenje putem kreiranja posebne tabele sa prostiranjem ključeva. U oba slučaja, transakcioni program ne vodi računa o pretvaranju simboličke u relativnu adresu.

Deklarativni jezik:

Temelji se na primenjenim tehnikama povezivanja podataka sa prostiranjem ključa. Koriste se dva alata za upitni jezik: 1. relaciona algebra- definisana na osnovama teorije skupova i skupovnih (unija, presek, razlika ,join, projekcija, selekcija, itd.) i 2.relacioni račun nad torkama(SQL) i nad domenima - definisani na osnovama predikatskog računa I reda

(12 principa relacionog MP pročitati)

Strukturalna komponenta

Primitivni koncepti nivoa intenzije- domen, obeležje, šema BP, skup obeležja

Primitivni koncept nivao ekstenzije- vrednost, podataka, torka, BP

Torka reprezentuje jednu pojavu entiteta ili poveznika. Pomoću torke se svakom obeležju, iz nekog skupa obeležja, dodeljuje konkretna vrednost. Torka je preslikavanje t : U→DOM. Postoji i restrikcija torke t[X]- dobije se uslov po kojem se prikazuju samo određeni delovi torke.

Relacija predstavlja konačan skup torki i reprezentuje skup realnih entiteta ili poveznika i ne mogu se pojaviti dve iste torke. Uobičajena repreyentacija relacije je pomoću tabele.

Relaciona šema baze podataka: (S,I) gde je S skup šema relacija, a I skup međurecalionih ograničenja.

Relaciona baza podataka je jedna pojava nad zadatom relacionom šemom baze podataka (S,I) gde svakoj šemi relacije iz skupa S odgovara jedna njena pojava i skup relacija s mora da zadovoljava sva međurelaciona ograničenja iz skupa I.

Baza podataka RBP= {ri | i $\in$ {1,..., n}} se nad šemom (S, I) nalazi u:

> **formalno konzistentnom stanju** ako ($\forall$ ri $\in$ RBP)(r_i zadovoljava sva ograničenja odgovarajuće šeme (R_i , O_i)) i RBP zadovoljava sva međurelaciona ograničenja iskazana putem I
>
> **suštinski konzistentnom stanju** ako se nalazi u formalno konzistentnom stanju i predstavlja vernu sliku stanja realnog sistema

Integritetna komponenta

Definisana je putem tipova ograničenja.

Karakteristike tipa ograničenja:

- formalizam za zapisivanje (definicija)
- pravilo za interpretaciju (validaciju)
- oblast definisanosti (tip logičke strukture obeležja nad kojom se ograničenje definiše)
- oblast interpretacije (tip logičke strukture podataka nad kojom se ograničenje interpretira)
- skup operacija nad bazom podataka koje mogu dovesti do narušavanja ograničenja datog tipa
- skup mogućih akcija kojima se obezbeđuje očuvanje validnosti baze podataka, pri pokušaju narušavanja ograničenja datog tipa

Kontrola ograničenja je centralna i ne može je zaobići ni jedan program ili korisnik. U slučaju pokušaja narušavanja ograničenja nekom operacijom ažuriranja, SUBP izaziva grešku i prekida operaciju Ili aktivnim mehanizmom dovodi stanje BP u konzistentno.

Oblasti definisanosti u relacionom MP:

vanrelaciono ograničenje-definiše se izvan konteksta šeme relacije

jednorelaciono ograničenje- definiše se nad tačno jednom šemom relacije

višerelaciono ograničenje-definiše se nad skupom ili nizom šema relacija, koji sadrži bar dva člana

Oblasti interpretacije u relacionom MP:

ograničenje vrednosti- interpretira se nad tačno jednom vrednošću nekog obeležja

ograničenje torke- interpretira se nad jednom torkom bilo koje relacije

relaciono ograničenje-interpretira se nad skupom torki bilo koje relacije

međurelaciono ograničenje-interpretira se nad barem dve, bilo koje relacije (bilo koja relacija označava jednu relaciju iz baze podataka ili relaciju koja je nastala primenom izraza relacione algebra nad jednom ili više drugih relacija.)

Tipovi ograničenja u RMP: ograničenje domena, ograničenje vrednosti obeležja, ograničenje torke, ograničenje ključa, funkcionalna zavisnost...

Do 64. slajda se ponavlja iz 4. lekcije

Ograničenje torke predstavlja skup ograničenja vrednosti obeležja, kojem je pridodat logički uslov. id(N) = id(R) = ({id(N, A) | A $\in$ R}, Uslov)

Ključ šeme relacije- minimalni podskup skupa obeležja šeme relacije, na osnovu kojeg se jedinstveno može identifikovati svaka torka relacije nad datom šemom. Odlikuje ga jedinstvenost (UNIQUE(N,X))- nema dve različite torke koje imaju iste vrednosti na atributima X, i minimalnost-ne možeš izbaciti nijedan atribut iz X a da ono i dalje ostane ključ.

Ograničenje ključa (Integritet entiteta) Key(N, X), X $\subseteq$ R, za sva obeležja ključa nula vrednosti su zabranjene. Svaka šema relacije mora posedovati makar jedan ključ. Ekvivalentni ključevi sus vi ključevi skupa K, a primaarni ključ je jedan izabrani od svih ekvivalentnih ključeva i svaka šema mora da poseduje tačno jedan. Koristi se u ulozi asocijativne (simboličke) adrese za povezivanje podataka u relacijama.

Skup svih ograničenja šeme relacije- kada šemu relacije treba implementirati u datom SUBP, zadaje se kao unija skupa ključeva, ograničenja jedinstvenosti i ograničenja torke N(R, K U Uniq U {id(R)})

Zavisnost sadržavanja (Inclusion Dependency)

Ova zavisnost se zapisuje kao:

Ni[X]$\subseteq$Nj[Y]

To znači da vrednosti iz atributa X u šemi Ni(referencirajuća šema relacije) moraju biti sadržane među vrednostima atributa Y u šemi Nj(referencirana šema relacije). Date su šeme relacija Ni(Ri,Oi) i Nj(Rj,Oj)

Dati su nizovi atributa:

- X=(A1,...,An), atributi iz šeme Ri
- Y=(B1,...,Bn), atributi iz šeme Rj

Domeni moraju biti kompatibilni. Zavisnost sadržavanja Ni[X]⊆Nj[Y] znači: 1. Svaka kombinacija vrednosti u kolonama X relacije Ni **mora postojati i u kolonama Y** relacije Nj

- Ovo se koristi npr. da se obezbedi **strani ključ** (foreign key)

Funkcionalna zavisnost f: X→Y Ako je poznata X vrednost, poznata je i Y vrednost i svakoj X vrednosti odgovara samo jedna Y vrednost. Primer: MBR→IME ako dve torke imaju istu vrednost za MBR, moraju imati istu vrednost i za IME.

$$F \models f$$

Ako sve relacije koje zadovoljavaju F(skup funkcionalnih zavisnosti) zadovoljavaju i f (jedna funkcionalna zavisnost), onda je f logička posledica skupa F. Dva skupa FZ F1 i F2 su ekvivalentna ako F1⊨F2 i F2⊨F1

f = A → B
Jedna zavisnost: ako dva reda imaju isti A, onda moraju imati isti B

Zatvarač u funkcionalnim zavisnostima je **skup atributa koji se mogu funkcionalno odrediti** iz nekog početnog skupa atributa koristeći dati skup funkcionalnih zavisnosti. F^+

Primer:

Neka je:

- F={A→B,B→C}
- Početni skup: X={A}

Tada:

1. Iz A→B, dobijemo B
2. Iz B→C, dobijemo C

Dakle:

F^+={A,B,C}

- **Armstrongova pravila izvođenja**
  - refleksivnost
    - $Y \subseteq X \mid\!- X \rightarrow Y$
  - proširenje
    - $X \rightarrow Y,\ W \subseteq V \mid\!- XV \rightarrow YW$
  - pseudotranzitivnost
    - $X \rightarrow Y,\ YV \rightarrow Z \mid\!- XV \rightarrow Z$
- Izvedena pravila izvođenja
  - uniranje desnih strana
    - $X \rightarrow Y,\ X \rightarrow Z \mid\!- X \rightarrow YZ$
  - dekompozicija desnih strana
    - $X \rightarrow Y,\ V \subseteq Y \mid\!- X \rightarrow V$
  - tranzitivnost
    - $X \rightarrow Y,\ Y \rightarrow Z \mid\!- X \rightarrow Z$

Sistem armstrongovih pravila izvodjenja je korektan, kompletan i minimalan.

Nepotpuna FZ- ako sadrži logičko suvišno obeležje na levoj strain

Primer: BRI+IME→PRZ, zbog BRI→IME - redukuje se u BRI→PRZ

Tranzititivna FZ - ako važi X→Y ∈ F + i Y→Z ∈ F + , a ne važi da je Y→X ∈ F +

Generisanje jednog ključa šeme relacije- polazi se od R i vrši se redukcija

Operacijska komponenta

Operacijsku komponentu čine: jezik za manipulaciju podacima (add, delete, update), jezik za definiciju podataka (kreiranje, brisanje i modifikovanje delova šeme BP) i upitni jezik koji služi za pružanje podatka na uvid korisniku. Upitni jezik sačinjavaju operatori za izražavanje upita, pravila za formiranje operanada upita i pravila za primenu tih operanada.

Vrse teoretskih upitnih jezika u RMP: relaciona algebra (zasnovana na teoriji skupova i skupovnih operacija(unija, presek, razlika)) i relacioni račun (nad torakama i nad domenima)

Selekcija- bira torke iz relacije po nekom kriterijumu σ F (r(R)) = {t ∈ r | F(t)} gde se sa F izražava kriterijum koji mora biti tačan.

Projekcija(restrikcija)- izdvajanje vresnosti pojedinih kolona iz relacije.

Prirodni spoj relacija- spajanje torki različitih relacija po osnovu istih vrednosti zajedničkih obeležja.

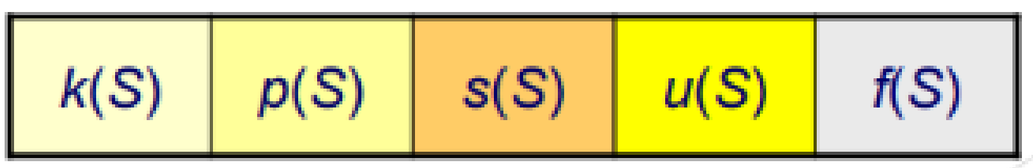
Dekartov proizvod relacija-spajanje formiranjem svih mogućih kombinacija torki iz dve relacije.

Theta spajanje relacija- selektovanje torki po nekom kriterijumu iz dekartovog proizvoda relacija.

6. Metode pristupa i organizacija datoteka

Datoteka je organizovana nad tipom sloga(slog predstavlja linearnu strukturu atributa). Datoteka ima **FSP (fizičku strukturu)**, ali u nju ulaze i podaci o **LSP (logičkoj strukturi)** koja sadrži informaciju o strukturi sloga(tip, dužina), veze izmedju slogova. Svaki slog predstavlja niz polja sa vrednostima atributa. Format sloga je pravilo za strukturiranje i interpretaciju sadržaja sloga.

Opšta struktura sloga datoteke(redosled polja ne mora biti isti)



Gde je k(S)- polja vrednosti atributa primarnog ključa. Jedina obavezna grupa polja.

p(S)- polja vrednosti ostalih atributa

s(S)- polja statusa sloga- indicator aktuelnosti sloga u LSP

u(S)- polja pokazivača za memorisanje veza u LSP

f(S)- kontrola polja kod slogova varijabilne dužine

Format polja sloga je uslovljen specifikacijom domena odgovarajuceg atributa. Postoje dve vrse polja u slogovima:

1. polja konstantne dužine- nije potrebno memorisati informaciju o granicama polja. Sva polja su konstantne dužine. Ovakvi slogovi se pojavljuju u praksi, imaju homogenu strukturu, jednostavnije je pristupiti i aziririati podatke, manja efikasnost upotrebe memorijskog prostora.
2. polja promenljive dužine- potrebno je memorisati informaciju o granicama polja pri čemu se koristi kontrolno polje f(S). To se može postići navođenjem aktuelne dužine polja u kontrolnom polju, neposredno ispred sadržaja polja, navođenjem specijalne oznake kraja polja u kontrolnom polju, neposredno nakon sadržaja polja ili uvođenjem posebne indeksne strukture sa rednim brojevima bajtova koji ukazuju na početke slogova. Barem jedno polje je promenljive dužine. Često se pojavljuju u praksi, imaju nehomogenu strukturu, sve suprotno od onog gore...

Vrste slogova prema ponavljanju vrednosti;

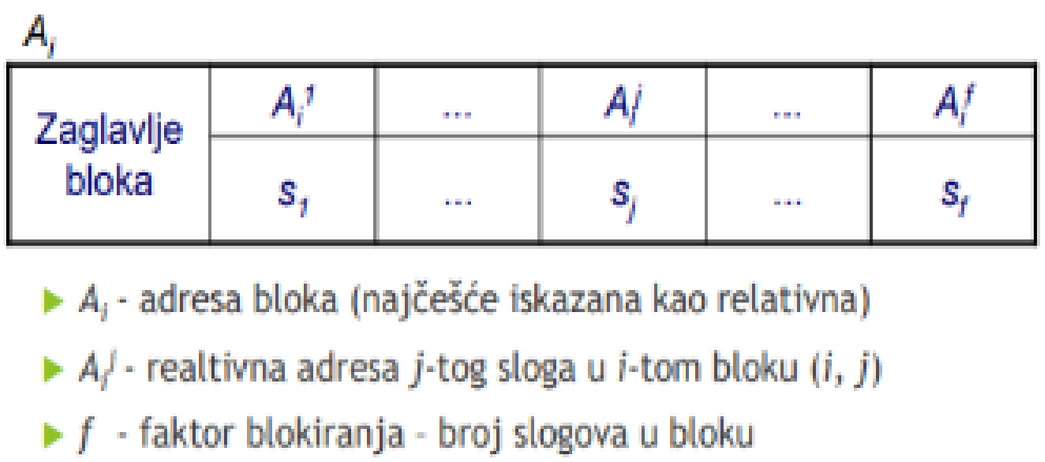
1. slogovi sa ponavljajućim vredonstima- višestruko pojavljivanje vredenosti atributa u jednom slogu i moraju biti slogovi varijabilne dužine.
2. slogovi bez ponavljajućih vrednosti- moguće je uvek projektovati tip sloga bez ponavljajućih vrednosti

Vrste adresa lokacija:

1. apsolutna adresa- strukturirana prema adresnom prostoru jedinice diska. Ne koristi se u organizaciji datoteka, stvara zavisnost od fizičkih karakteristika uređaja i ne zahteva transformaciju
2. relativna adresa- predstavlja redni broj lokacije. Vrlo često se koristi u organizaciji datoteka, obezbeđuje nezavisnost od fizičkih karakteristika i zahteva jednu ili više transformacija do apsolutne adrese.
3. simbolička adresa- vrednost ključa. Često se koristi i zahteva transformaciju u relativnu adresu.

Struktura datoteke kao niza blokova

Blok predstavlja niz slogova i ima konstantan kapacitet. Uobičajeno, jedan blok predstavlja niz od 2 n (n ⩾ 0) fizičkih blokova, ali se može desitit i da bude jednak ili manji od kapacitetea fizičkog bloka.



Vrste blokova:

1. Blokovi sa slogovima promenljive dužine- više slogova može biti smešteno u jedan blok i dozvoljeno je i da veličina jednog sloga premaši kapacitet bloka.
2. Blokovi sa slogovima konstantne dužine- homogena struktura bloka i datoteke, svaki blok datoteke sadrži uvek isti broj slogova
   $B = \lceil (N + x) / f \rceil$
   Gde je f – faktor blokiranja datoteke, B – ukupan broj blokova datoteke, N – ukupan broj slogova u LSP datoteke , x – broj dodatno upotrebljenih specijalnih slogova

Proračun potrebnog kapaciteta datoteke je moguć u slučaju primene blokova sa slogovima konstantne dužine

$f = \lfloor (Kb - Kz) / Ks \rfloor$ gde je K_s kapacitet sloga, K_b' kapacitet bloka, a K_z kapacitet zaglavlja bloka.

$Kd = BKb + Wd$ gde je K_d kapacitet datoteke, a W_d kapacitet STD za datoteku

Strogo strukturirana datoteka je strogo tipizovana datoteka sa pridruženom semantikom i organizovana je kao struktura nad skupom slogova.

Zaglavlje datoteke je potrebno proširenje osnovne structure datoteke za koji se uvodi specijalni slog na početku datoteke. U tom slogu se nalaze podaci o organizaciji datoeteke i format bloka i sloga datoteke (broj slogova, dužina i format sloga, pozicija polja ključa u slogu...)

Oznake kraja datoteke:

a) uvođenjem specijalnog sloga za oznaku kraja datoteke
b) uvođenjem specijalne oznake kraja u polje pokazivača
c) vođenjem posebne evidencije zauzetosti prostora
d) kraj datoteke je kraj prostora dodeljenog datoteci

Metoda pristupa – paket programa za podršku usluga visokog nivoa: upravljanje strogo strukturiranim datotekama i upravljanje baferima metode pristupa. Takođe pruža podršku različitim načinima memorisanja logičkih veza i adresiranja(u storogo strukturiranim

datotekama), vodi brigu o kategorijama(zaglavlje, početak, kraj datoteke, tekući pokazivač) i podrška je u izgradnji specijalnih pomoćnih struktura za poboljšanje efikasnosti obrade podataka. Kreira, traži, pretražuje i ažurira sadržaj datoteka. Koristi usluge niskog nivoa izabranog nivoa OS. Obezbezbeđuje nezavisnost aplikativnog programa od usluga niskog nivoa OS(obezbeđuje preslikavanje strogo strukturirane datoteke u FSP niza fizičkih blokova i obezbeđuje transformacije relativne adrese sloga ili bloka datoteke u relativnu adresu bajta ili fizičkog bloka).

Upravljanje strogo strukturiranim datotekama

1. Podrška organizacije slogova i polja(kontsntne ili promenljive dužine)
2. Podrška različitih tipova podataka
3. Podrška različitih kodnih rasporeda
4. Konverzije podataka: iz tipa podatka programske promenljive u tip podatka atributa datoteke i obratno ili iz tipa podatka atributa strogo strukturirane datoteke u niz bajtova i obratno.
5. Usluge razmene podataka sa aplikativnim programom:
    a. Na nivou sloga – grupisanje slogova u blokove pri upisu podataka, rastavljanje blokova na slogove pri čitanju podataka, održavanje tekućeg pokazivača kao relativne adrese(redni broj sloga u datoteci)...
    b. Na nivou bloka- razmena sadržaja kompletnih logičkih blokova između aplikativnog program ai datoteke, održavanje tekućeg pokazivača kao relativne adrese(redni broj bloka u datoteci)
6. Usluge pristupa podacima iz aplikativnih programa:
    a. Sekvencijalni pristup- slogovima ili blokovima datoteke. Automatski održavaju vrednost tekućeg pokazivača.
    b. Direktni pristup-slogovima ili blokovima datoteke. Zahtevaju eksplicitno zadavanje vrednosti tekućeg pokazivača
    c. Dinamički (kombinovani) pristup
7. Pozivi rutina metode pristupa- otvaranje i zatvaranje datoteka, učitavanje ispisivanje sadržaja sloga ili bloka, pozicioniranje na slog ili blok datoteke, kreiranje, brisanje datoteke...

Upravljanje baferima metode pristupa- alociranje i dealociranje bafera, vođenje evidencije o sadržaju bafera.

Tri nivoa baferisanja podataka:

nivo sistemskih bafera -upravlja OS

nivo bafera metode pristupa - kojim upravlja okruženje u kojem je implementirana metoda pristupa

nivo lokacija promenljivih u aplikativnom programu -upravlja aplikativni program

Metoda pristupa

Okruženja koja uključuju metode pristupa su:

1. Operativni sistem- Uglavnom su to bil stariji operativni sistemi na mejnfrejm računarima. Tada nije postojala jasna podela između niskih i visokih nivoa usluga OS. Ovakvi sistemi su podržavali upravljanje blokovima i baferima za metode pristupa.
2. Progranski jezik sa pridruženim paketima funkcija- svaki savremeni programski jezik pruža određene usluge metode pristupa. Eksplicitno koriste usluge niskog nivoa izabranog OS i najčešće pružaju samo usluge na nivou sloga datoteke(upravljanje blokovima i baferima sakriveno od aplikativnog programa).
3. Sistem za upravljanje bazama podataka- Svaki SUBP obezbeđuje usluge metode pristupa, eksplicitno koristi usluge niskog nivoa izabranog OS(mada je moguće, u specifičnim situacijama, da SUBP "zaobiđe" usluge niskog nivoa OS). Podržava upravljanje blokovima i baferima metode pristupa.

Parametri organizacije datoteka

Organizacija podataka je projekat logičke strukture obeležja (LSO) i pojekat i implementacija FSP(ovde spada: izbor načina dodele lokacija slogovima, izbor načina memorisanja logičkih veza između slogova u LSP, projektovanje osnovnih struktura podataka, projektovanje pomoćnih struktura podataka, proračun i rezervisanje potrebnog prostora na eksternim memorijskim uređajima, smeštanje slogova sa vezama na eksterne memorijske uređaje...) sa ciljevima da se obezbede zadovoljenje korisničkih zahteva i uslovi za efikasnu obradu podataka. Rezultat organizovanja podataka su sistem baze podataka ili sistem datoteka.

Organizacija datoteke se svodi na projektovanje LSO i izbor vrste organizacije datoteke koja zavisi od vrednosti parametara(način dodele lokacija slogovima i način memorisanja logičkih veza između slogova u LSP)

Način dodele lokacija slogovima(DLS)- postoje 2 pristupa:

1. Dodavanje na kraj fajla(fizički)- To znači da se novi slog uvek upisuje na sam kraj fajla, odmah posle poslednjeg. Ako fajl raste, sistem povećava njegovu veličinu, ali: Poslednji blok fajla sme da ostane delimično popunjen i Svi ostali blokovi moraju biti u potpunosti popunjeni. Ako fajl ima statičku alokaciju (unapred rezervisan), a sve je puno, novi se više ne mogu upisati.
2. Dodavanje na prvo slobodno mesto (linearno)- Faj lima svoj Spisak slobodnih blokova i kada se upisuju novi podaci, fajl traži prvi Slobodan blok u toj strukturi i tu smešta slog. Alokacija je statička- fajl se ne povećava.

3. Direktno, na osnovu ključa(heširanje)- Novi slog dobija svoje mestp u fajlu na osnovu vrednosti ključa. Prostor fajla se alocira statički i ovaj princip se radi samo kada sus vi slogovi iste dužine. Moguće je transformisati vrednosti ključa pomoću hash-a(matematička formula) ili tabelarno.

Način memorisanja logičkih veza(MLV):

1. Fizičkim pozicioniranjem- Logički susedni slogvoi smešteni su jedan pored drugog u fajlu.
2. Pomoću pokazivača (relativne adrese)- Slog sadrži pokazivač (reference) na sledeći slog. Dakle, pored podataka ima i polje koje pokazuje gde se nalazi sledeći. To se radi na dva načina:
    a. Polje pokazivača je ugrađeno u sam slog- svaki slog ima adresu sledećeg
    b. Polje pokazivača je smešteno u pomoćnu strukturu- fajl ima pomoćnu tabelu u kome se pamte parovi identifikator sloga-pokazivač. Cesto se organizuje kao stablo.
3. Logičke veze se uopšte ne pamte- radi se pomoću posebnih programa.

Vrste organizacije datoteka

Osnovne organizacije

FSP nad skupom slogova oragnizovana je u jednoj memorijskoj zoni (često bude da je to jedna datoteka operativnog sistema). U osnovne organizacije datoteka spadaju serijska(pile, heap), sekvencijalna, spregnuta, rasuta sa jedinstvenim memorijskim prostorom.

Složene organizacije

Dobijaju se kombinovanjen osnovnih organizacija, FSP uključuju barem dve memorijske zone. Ovde spadaju rasute-hash- sa zonom prekoračenja(primarna zona je osnovna struktura- osnovna rasuta organizacija, a zona prekoračenja je nastavak osnovne strukture i to je spregnuta ili serijska organizacija), statičke indeksne(indeks-sekvencijalne. Primarna zona je osnovna struktura- sekvencijalna organizacija, a zona prekoračenja je nastavak osnovne strukture i to je spregnuta organizacija. zona indeksa je pomoćna struktura, isto spregnuta organizacija u obliku n-arnog stablja traženja.) i dinamičke indeksne (organizacije sa B- stablom. Primarna zona je osnovna struktura- serijska ili spregnuta organizacija, a zona indeksa je pomoćna struktura, isto spregnuta organizacija u obliku jedne od varijanti B-stabla).

Navedene vrste organizacije se u praksi pojavljuju kao fizičke organizacije datoteka(u sistemima datoteka) ili fizičke organizacije tabela(u sistemima baza podataka. Svaka tabela BP može biti distribuirana u više datoteka podataka kojima upravlja SUBP i u jednoj datoteci podataka kojom upravlja SUBP može biti smešteno više tabela BP).

Opšte procedure nad datotekama

Formiranje datoteke

Postupak kreiranja fajla i smeštanja slogova u njega. Fajl se formira u skladu s organizacijom koju smo odlučili da primeninmo i na osnovu podataka koje već imamo(u nekom drugom fajlu ili strukturi) ili ih korisnik direktno zadaje. Postoje dve vrste datoteka:

1. Datoteke formirane u posebnom postupku. Nastaju sve odjednom, u jednoj operaciji, a podaci se učitavaju iz drugih fajlova ili ih direktno unosi korisnik. Takvi fajlovi su najčešće: sekvencijalni, spregnuti, statički rasuti, statički indeksni.
2. Datoteke formirane u redovnom postupku ažuriranja. Nastaju postepenim dodavanjem novih slogova. takvi fajlovi su serijski, indeksni i dinamički rasuti.

Pristupanje datoteci

Postupak pozicioniranja na željenu lokaciju. Postoje tri vrste pristupa:

1. Sekvencijalni pristup gde se automatski održava relativna adresas tekućeg pokazivaca, a sama operacija se odnosi na neposredno susednu lokaciju u odnosu na lokaciju na kojoj je obavljena prethodna operacija.
2. Direktni pristup gde se ekslicitno zadaje relativna adresa tekućeg pokazivača koji ukazuje na lokaciju nad kojom će se realizovati neka operacija.
3. Dinamički pristup koji je kombinacija prethodna dva.

Traženje u datoteci

AT: dom(K) → Ind × A × S

dom(K)-vrednost ključa, Ind-indeks bloka, A-datoteka, S-slog

Algoritam je u stanju da generiše i vrati u program indikaciju uspešnosti traženja Ind = {true, false}, relativnu adresu mesta zaustavljanja traženja i sadržaj sloga na mestu zaustavljanja traženja.

Specifični algoritmi traženja na izlazu će generisati samo vrednosti onih parametara koji su stvarno neophodni aplikativnom program.

Neke od mogućih primena algoritma su: utvrđivanje da li traženi slog postoji u datoteci ili ne, da bi se utvrdila adresa na kojoj se traženi slog nalazi ili da bi se preneo sadržaj traženog sloga u aplikativni program.

Algoritam:

1. Generiše se početna adresa (prva lokacija) gde da traži.
2. Postavlja se zastavica „postoji potreba za nastavkom“ na DA.
3. Ulazi u petlju sve dok postoji potreba da nastavimo:
Pročitamo slog s tekuće adrese.
Ako ključ tog sloga poklapa traženi,
traženje je uspešno.
Postavljamo „postoji potreba za nastavkom“ na NE.

Ako ključ ne poklapa, ali postoje uslovi da nastavimo (npr. postoji sledeća adresa), računamo sledeću adresu i nastavljamo.
Ako nema uslova za nastavak, traženje je neuspešno i prekidamo petlju.

Metode traženja s obziro na frstu postupka mogu biti:

1. linearno traženje- moguće u sekvencijalnim, serijskim i rasutim organizacijama
2. binarno traženje- samo u sekvencijalnim organizacijama
3. traženje praćenjem pokazivača- samo i spregnutim organizacijama
4. traženje metodom transformacije argumenta u adresu- h: dom(K) → A, moguće u rasutim organizacijama

Metode traženja s obzirom na predistoriju traženja

1. traženje slučajno odabranog sloga(tso): izbor početne adrese traženja je nasumično od strane algoritma. Moguće je u svim organizacijama datoteke.
2. traženje logički narednog sloga (tln): početna adresa traženja predstavlja adresu na kojoj je zaustavljeno prethodno traženje (moguće je ako je prethodno postojalo barem jedno traženje). Svaka naredna adresa traženja može biti samo adresa logički narednog sloga. Moguće u organizacijama u kojima se vode podaci o logički narednom slogu.

Pretraživanje u datoteci

dom(LogUslov) → P(S) ili P(A)

ulaz (dom) je logički uslov a rezultat može da bude ili P(S)- parcijalnni skup slogova koji zadovoljavaju uslov ili P(A)- parcijalni skup adresa slogova koji zadovoljavaju uslov. Pretraživanje je uspešno ako skup slogova koji zadovoljava uslov nije prazan. Algorithm zahteva definisanje sintakse (kako se uslov piše) i predstavlja logički izraz. Za izraz se mogu koristiti logički izrazi, relacioni izrazi, tipski izrazi(aritmetički, alfanumerički ili datumski operatori) ili atributi sloga, konstante ili funkcije primenjene na atribute. Postoje i specijalni tipovi uslova,npr.: konjuktivni gde sve mora da važi ili disjunktivni gde može bilo šta da važi.

Sekundarni ključ je niz obeležja structure po kojem se vrši pretraživanje.

Ažuriranje datoteke

Postupak dovođenja LSP datoteke u sklad sa izmenjenim stanjem klase entiteta u realnom sistemu. Osnovne operacije su:

1) upis novog sloga u datoteku- zahteva prethodno neuspešno traženje
2) modifikacija vrednosti neprimarnih atributa sloga- zahteva prethodno uspešno traženje i zabranjuje se modifikacija primarnog ključa

3) brisanje postojećeg sloga iz datoteke- zahteva prethodno uspešno traženje. Vrste brisanja su:
   a) logičko brisanje sloga iz datoteke- svodi se na izmenu vrednosti polja statusa sloga(iz aktuelnog u neaktuelno). Neaktuelni slog i dalje zauzima lokaciju u memorijskom prostoru, a lokacije neaktuelnih slogova oslobađaju se reorganizacijom.
   b) fizičko brisanje sloga iz datoteke- dovodi do izmene sadržaja bloka u kojem se nalazio izbrisani slog.

Obrada datoteka

Algoritamski iskazani niz operacija nad LSP jedne ili više datoteka sa ciljem svrsishodne transformacije podataka datoteka. Operacije se mogu koristiti za pristup slogovima, traženje i oretraživanje, ažuriranje, generisanje novih podataka.

Podela datoteka prema vrstama primenjivih operacija u obradi:

- ulazna datoteka- vrši se samo čitanje
- izlazna datoteka- samo se zapisuju novi slogovi u obradi
- ulazno-izlazna datoteka- i čitanje i ažuriranje slogova

Podela prema ulozi u traženju slogova

- vodeća datoteka-generiše argumente traženja ili pretraživanja slogova tokom obrade. Makar jedna ulazna datoteka mora biti vodeća.
- obrađivana datoteka- isključivo se vrše traženja na osnovu generisanih argumenata
- vodeća i obrađivana datoteka- datoteka sa obe uloge

Vrste obrade prema načinima traženja slogova u obrađivanoj datoteci:

- direktna obrada- u svakom narednom koraku obrade zahteva se traženje slučajno odabranog sloga (tso)
- sekvencijalna obrada- u svakom narednom koraku obrade zahteva se traženje logički narednog sloga (tln) i/ili sekvencijalni pristup fizički susednoj lokaciji.

Reorganizacija datoteke

Ponovno formiranje datoteke u cilju dovođenja u sklad FSP sa novim stanjem LSP. Npr: nagomilavanje neaktuelnih, logički izbrisanih slogova koji zauzimaju lokacije u FSP. Ovakvu reorganizaciju traže sekvencijalna, spregnuta, statička rasuta, statička indeksna organizacija. Nije potrebna reorganizacija kod serijske, indeksne sa B stablom i dinamički rasute organizacije. Kod poslednje dve se reorganizacija sprovodi dinamički i lokalizovana je.

Performanse obrade datoteke

Mere podobnosti datoteke sa zadatom organizacijom da participira u obradi kao vodeća ili obrađivana ili u redoslednoj ili direktnoj obradi. Idealna organizacija datoteke zahteva tačno

onoliko lokacija koliko sadrži slogova(faktor popunjenosti 100%), zahteva najviše jedan pristup za tso i tln, zahteva najviše jedan pristup za pretraživanje po bilo kom zadatom uslovu, zahteva jedan pristup za bilo koju operaciju ažuriranja, nikada ne zahteva reorganizaciju.

Izbor vrste organizacije datoteke predstavlja kompromisno rešenje zbog toga što je nemoguće da jedna vrsta organizacije zadovolji sve navedene zahteve. Favorizuju se željene mere performansi, u odnosu na zauzeće memorijskog prostora.

Ukupno vreme traženja slogova zavisi od: broja i vremena pristupa blokovima na jedinici diska(odlučuje organizacija fajla is am uslov pretraživanja), veličine fajla , vreme prenosa bloka s diska u radnu memoriju(zavisi od karakteristika diska), broja i vremena poređenja ključa(ako imamo više podataka, više puta ćemo morati da uporedimo.)

7. Serijska i sekvencijalna organiyacija datoteke

Serijska organizacija

Slogovi su smešteni jedan za drugim. Fizička struktura ne sadrži informacije o vezama između slogova logičke strukture datoteke. Ne postoji veza između vrednosti ključa sloga i adrese lokacije u koju je smešten. Redosled memorisanja slogova najčešće prema hronološkom redosledu njihovog nastanka. Oznaka za kraj datoteke je *.

Formiranje serijske datoteke

Nastaje u postupku obuhvata podataka, kada se podaci unose u sistem. Slogovi se formiraju prenosom podataka iz različitih izvora(dokumenata, uređaja), upisuju se jedan za drugim u fajl bez praznina i svaki sledeći slog se dodaje na kraj. Rezultat ovog procesa je neblokirana (bez blokova) ili blokirana (podeljena u blokove) serijska datoteka. Unos podataka obavlja čovek ili specijalizovani softver koristeći program (UI) koji omogućava unos, kontrolu i formatiranje opodataka i ulazne/izlazne uređaje.

Program za obuhvat podataka ima svoj korisnički interfejs koji definiše lazout, omogućava navigaciju(miš, tastatura), kontroliše sadržaj polja(naslov, tip polja, način vizuelizacije, maksimalni broj znakova itd.) i omogućava operacije nad podacima(unos, modifikacija, brisanje...).

Prikupljanje podataka se može obavljati u realnom vremenu ili u odloženom režimu (naknadno) i ovaj proces uključuje i verifikaciju pošto se obično događa kada operater nije bio prisutan. Verifikacija je proces suštinske provere da li su uneti ispravni podaci. Može biti ručna(drugi operater unosi iste podatke) i automatizovana (pomoću softvera).

Traženje sloga u serijskoj datoteci

Ne postoji funkcionalna veza između vrednosti ključa i adrese lokacije sloga i traženje logički narednog je ista procedura kao da se traži logički odabran. (Ovo znači da ako si pronašao jedan slog (npr. slog za Marka), i želiš da pronađeš sledeći slog u datoteci (npr. slog za Milicu), računar i

dalje nema direktnu putanju do tog sledećeg sloga. Moraće da nastavi da pregleda slogove jedan po jedan, baš kao da traži bilo koji slučajni slog.).

Uspešno traženje, ukupan broj pristupa Ru : $1 \leqslant Ru \leqslant B$

Neuspešno traženje, ukupan broj pristupa Rn : Rn = B B – ukupan broj blokova serijske datoteke

- ukupan broj blokova datoteke: $B = \left\lceil \frac{N+1}{f} \right\rceil$
  - N - broj slogova
  - f - faktor blokiranja
  - + 1 - zbog specijalnog sloga sa oznakom kraja datoteke

Obrada serijske datoteke

1. Direktna obrada- pristupanje svakom pojedinačnom slogu (podatku) u datoteci nezavisno od njegovog mesta ili redosleda.
2. Redosledna obrada-obrađujete slogove jedan za drugim, po hronološkom redosledu kako su i zabeleženi u datoteci.

Program koji vrši redoslednu obradu serijske datoteke učitava sukcesivne slogove vodeće datoteke(program čita podatke jedan po jedan iz glavne datoteke). Svaki naredni slog vodeće datoteke sadrži logički narednu vrednost ključa obrađivane serijske datoteke(ako je glavna datoteka sortirana (npr. po ID-u kupca), onda će svaki sledeći pročitani slog imati sledeći ID kupca). Te vrednosti ključa se koriste kao argumenti za traženje u serijskoj datoteci metodom linearnog traženja. Metoda pretrage je "linearna", što znači da se mora krenuti od početka te druge datoteke i pregledati je slog po slog dok se ne nađe podatak koji se traži.

U režimu direktne obrade- Kada se radi direktna obrada, program i dalje čita slogove jedan po jedan iz glavne datoteke. Ali ovde, ključevi (ID-ovi, imena, itd.) u tim slogovima nisu nužno sortirani ili u nekom logičkom nizu. Oni su "slučajno odabrani" u smislu da program traži specifične, pojedinačne slogove, a ne čitav niz. Traženje mora biti linearno.

Ažuriranje serijske datoteke

Upsi novog sloga se vrši na prvu slobodnu lokaciju na kraju datoteke i mora mu prethoditi jedno neuspešno traženje.

Brisanju postojećeg sloga mora prethoditi jedno uspešno traženje i najčešće se odvija samo logičko pošto bi fizičko zahtevalo veliki broj pristupa.

Serijska organizacija je pogodna za male fajlove koji mogu da stanu u radnu memoriju ili ako imamo puno pristupa podacima. Ovakva struktura je veoma jednostavna, lako se formira, upisuje i čita, ako se kombinuje sa indeksnim strukturama, postaje pogodna za direktnu obradu. To je osnovni fizički sloj relacionih baza podataka, ali i polazna osnova za kasnije kreiranje složenijih organizacionih formi.

Sekvencijalna organizacija datoteke-fizički sevencijalna organizacija

Slogovi su smešteni sukcesivno jedan za drugim, a logički susedni slogovi se smeštaju u fizički susedne lokacije(slog sa najmanjom vrednošću ključa smešta se u prvu lokaciju). Veza između memorisanih vrednosti ključa k(S) i adresa lokacija nije ugrađena u strukturu datoteke i ne predstavlja bilo kakvu matematičku funkciju.

Formiranje sekvencijalne datoteke se izvodi sertiranjem serijske datoteke po rastućoj ili opadajućoj vrednosti ključa.

Traženje sloga u sekvencijalnoj datoteci

Kod traženja slučajno odabranog sloga moguće je primeniti linearno ili binarno traženje. Ovo ima smisla ako je cela datoteka smeštena u OM.

Traženje logički naredni sloga se vrši linearnom metodom traženja, gde se, počevši od prvog, fizički susedni blokovi učitavaju u OM. U centralnoj jedinici se vrši upoređivanje argumenata traženja i vrednosti ključa sukcesivnih slogova dok se traženi slog ne pronađe, argument traženja ne postane manji od vrednosti ključa sloga ili ne dođe do kraja datoteke. Traženje novog, logički narednog sloga, započinje od sloga na kojem se prethodno traženje zaustavilo.

Broj pristupa pri uspešnom i pri neuspešnom traženju $0 \leqslant R \leqslant B$ – i gde je i redni broj tekućeg bloka u odnosu na početak.

Obrada sekvencijalne datoteke

1. Direktna obrada-ima smisla ako je sekvencijalna datoteka mala, tako da se može smestiti u operativnu memoriju

2. Vodeća datoteka u direktnoj i redoslednoj obradi- sukcesivno učitavanje fizički susednih slogova, počevši od prvog pa do poslednjeg

3. Redosledna obrada- iterativan proces gde vodeća datoteka generiše logički naredne vrednosti ključa za traženje u obrađivanoj sekvencijalnoj datoteci. U svakom koraku procesa, program traži sledeći relevantni slog. Vrši se metodom linearnig traženja- da bi se našao taj sledeći slog u sekvencijalnoj datoteci, program mora da krene od početka (ili od mesta gde je stao) i da pregleda slog po slog dok ga ne pronađe. Nema direktnog skoka.

Ažuriranje sekvencijalne datoteke

Upis novog sloga- pomeranje udesno svih slogova sa vrednostima ključa većim od vrednosti ključa novog sloga.

Brisanje postojećeg sloga- pomeranje za jednu lokaciju ulevo svih slogova sa većom vrednošću ključa.

Modifikacija sadržaja sloga- prethodno pronalaženje sloga – uspešno traženje

Upis i brisanje predstavljaju problem zbog pomeranja.

Ažuriranje u režimu direktne obrade(menjaju se pojedinačni slogovi):

> Pomeranje polovine od ukupnog broja slogova za jednu lokaciju udesno (pri upisu) ili ulevo (pri brisanju) sloga. Primenjuje se kada je kompletna datoteka smeštena u OM.

Ažuriranje u režimu redosledne obrade:

Ne može se direktno menjati postojeća serijska datoteka, već se mora kreirati potpuno nova na osnovu postojeće. Ovo se radi kad se cela datoteka ne može smestiti u operativnu memoriju.

Ds - **Obrađivana, ulazna (stara) sekvencijalna datoteka:** Ovo je originalna serijska datoteka koju želimo da ažuriramo.

Dn - **Obrađena, izlazna (nova) sekvencijalna datoteka:** Ovo je potpuno nova serijska datoteka koja će biti rezultat ažuriranja. U nju se upisuju svi promenjeni podaci.

Dp - **Vodeća datoteka promena, serijska, ulazna:** Ovo je datoteka koja sadrži *samo promene* (dodavanja, brisanja, izmene) koje želimo da primenimo na Ds. I ova datoteka mora biti serijska i sortirana.

Dg - **Datoteka grešaka, izlazna:** Ako se tokom procesa ažuriranja pojave neke greške, te greške se beleže u ovu datoteku.

-----pola toga sam preskocila prosto ne mogu

Prednosti sekvencijalne organizacije su :

- najpogodnija fizička organizacija za redoslednu obradu
- ekonomično korišćenje memorijskog prostora
- mogućnost korišćenja i magnetne trake i magnetnog diska, kao medijuma

Mane:

- nepogodnost za direktnu obradu
- potreba sortiranja pri formiranju
- relativno dugotrajan postupak ažuriranja

8. Rasuta organizacija datoteke

Često se naziva i direktnom jer se slogovima pristupa direknto na osnovu poznavanja adrese memorijske lokacije u kojoj su smešteni. Adresa lokacije se dobija transformacijom vrednosti identifikatora sloga u adresu.

h: dom(K) → A K-domen identifikatora, A-skup adresa lokacija memorijskog prostora datoteke

Identifikator je skup obeležja čije vrednosti jednoznačno određuju slogove datoteke- interni ili eksterni. Vrste transformacija vrednoti identifikatora u adresu su:

1. deterministička- funkcija h je injektivna. Svakoj vrednosti identifikatora odgovara jedna adresa i svakoj adresi odgovara najviše jedna vrednost identifikatora.
2. probabilistička- svakoj vrednosti identifikatora odgovara jedna adresa i jednoj adresi može odgovarati više rezultata transformacije.

Fizička struktura podataka ne sadrži informaciju o vezama između slogova logičke strukture datoteke. U dve fizički susedne lokacije mogu a ne moraju biti memorisani logički susedni slogovi. Slogovi su, na slučajan način, rasuti po memorijskom prostoru datoteke

Baket – blok kod rasutih datoteka

Vrste rasutih datoteka s obzirom na način alokacije memorijskog prostora mogu biti:

1. Statičke-veličina adresnog prostora određuje se i kompletno alocira unapred
2. Dinamičke-veličina dodeljenog adresnog prostora menja se tokom ažuriranja, saglasno potrebama

Statičkoj rasutoj datoteci, u postupku formiranja, dodeljuje se Q = bB lokacija (b je factor baketiranja). Slogovi se upisuju saglasno hronološkom redosledu nastanka. Upisu sloga prethodi neuspešno traženje, na osnovu obavljene transformacije identifikatora u adresu, a slog se smešta u baket sa izračunatom adresom

Direktna i relativna organizacija datoteke

Postoji vrlo jednostavan i direktan način da se eksterni identifikator (npr. ID broj) pretvori u adresu (lokaciju) gde se slog nalazi na disku. Funkcija h samo preslikava vrednost iz skupa A (identifikatora) u skup A (adresa). Vrednost identifikatora je, u isto vreme, i adresa baketa. Direktna organizacija se ne oslanja na sadržaj samog sloga da bi pronašla njegovu adresu, niti na složene veze između slogova. Oslanja se isključivo na vrednost identifikatora.

Vrste direktnih datoteka, s obzirom na vrstu upotrebljavanih adresa:

- Vrednosti identifikatora su mašinske adrese baketa: Identifikator je direktno fizička adresa na disku.
- Vrednosti identifikatora su relativne adrese baketa: Identifikator ukazuje na poziciju unutar datoteke, ali ta pozicija nije nužno fizička adresa. fizičku adresu na disku.

Direktna datoteka sa mašinskim adresama koristi adrese baketa na disku, oblika(u,c,t,s) ii ma samo istorijski značaj zbog brojnih nedostataka (čvrsta povezanost programa sa fizičkim karakteristikama ili odsustvo logičke veze između vrednosti identifikatora i sadržaja sloga)

Direktna datoteka sa relativnim adresama – slogovi su smešteni u memoriji prema rednom broju. Memorija se deli na Q prostora (od 1 do Q), redni broj sloga predstavlja eksterni identifikator zbog čega nema više jake povezanosti između slogova i fizičkih kakrakteristika diska. Glavna mana je nedostatak veze između identifikatora i sadržaja sloga.

Relativna metoda pristupa

Deo sistema za upravljanje podacima koji obavlja transformaciju relativne adrese u mašinsku i ostale operacije. Obavlja se na nivou baketa (kada program zatraži podatak, sistem mu daje ceo blok podataka, a ne pojedinačni slog.) Prihvata samo vrednost faktora blokiranja jednaku 1. "Faktor blokiranja" je broj slogova u jednom bloku. Ako je faktor blokiranja 1, to znači da jedan blok sadrži samo jedan slog. 1 blok=1 slog Aktivnosti blokiranja i rastavljanja blokova na slogove moraju se realizovati u okviru samog programa. Pošto sistem ne radi to automatski, ako programer želi da ima više slogova u jednom bloku, on mora sam da napiše kod koji će se baviti pakovanjem slogova u blokove i raspakivanjem kada ih čita.

Formiranje relativne datoteke se odvija na osnovu vodeće serijske ili neke druge vrste datoteke. Slogovi se sukcesivno učitavaju i automatski se pridružuje vrednost identifikatora slogu. Potom se smešta slog u lokaciju sa pridruženom relativnom adresom.

Traženje sloga u relativnoj datoteci se izvodi zadavanjem relativne adrese lokacije metodi pristupa.

Ažuriranje relativne datoteke se vrši u režimu direktne obrade. Upis se vrši tako što se novom slogu pridružuje vrednost identifikatora, a potom se slog upisuje u datoteku ako postoji slobodna lokacija u bloku. Brisanje se realizuje kao logičko tako što se nakon provere vrednosti ključa izmeni sadržaj statusnog polja sloga.

Ovaj princip je pogodan za traženje slučajno odabranog sloga i za traženje logički narednog sloga. Uvođenje relativne adrese lokacije kao identifikatora rešava problem čvrste povezanosti slogova datoteke sa karakteristikama memorijskog uređaja. Takođe je moguća primena relativne metode pristupa koja je osnov za izgradnju spregnute datoteke (relativna adresa lokacije logički narednog sloga smešta se u polje pokazivača tekućeg sloga), osnov za izgradnju rasutih datoteka sa probabilističkom transformacijom ključa u adresu (probabilistička transformacija brine o vezi između vrednosti identifikatora i relativne adrese) i osnov za izgradnju indeksnih datoteka.

Statička rasuta organizacija datoteke

Ovo je napredniji način organizacije podataka koji omogućava veoma brz pristup slogovima, slično direktnoj organizaciji, ali sa jednom bitnom razlikom: umesto da se ključ direktno

preslikava u adresu, koristi se posebna funkcija (heš funkcija) koja "raspršuje" ključeve po adresnom prostoru. Ključ može uzimati jednu od v^p ili v^p -1 vrednosti(Ako imate p pozicija i svaka pozicija može imati v različitih vrednosti (npr. v=10 za cifre 0-9), onda je ukupan broj mogućih ključeva 10p. Ako je ključ npr. petocifreni broj, imate 105 mogućih ključeva.).

Formula na dnu slajda: $v^p >> Q \geq N$ gde je v^p (ukupan broj mogućih ključeva) je mnogo veći od Q (ukupan broj raspoloživih lokacija/mesta u datoteci), koje je veće ili jednako sa N (stvarni broj slogova koje trenutno imamo u datoteci).

Metoda probablističke transformacije

Uvode se kako bi se prevazišli nedostaci do kojih dovodi deterministička transformacija vrednosti ključa u adresu. Cilj ovih metoda je da uzmu neki ključ (npr. JMBG, ime proizvoda) i da ga "transformišu" u **adresu** (lokaciju) gde bi se slog sa tim ključem trebao nalaziti u datoteci. Zato se koristi "probabilistička transformacija", često zvana **heš funkcija**.

1. Pretvaranje nenumeričke u numeričku vrednost ključa $k(S) \in \{ 0, 1, \ldots, v^p - 1 \}$
2. Pretvaranje numeričke vrednosti ključa k(S) u pseudoslučajan broj T(k(S))- Ovaj numerički ključ se zatim provlači kroz neku matematičku funkciju (heš funkciju) koja ga pretvara u drugi, **pseudoslučajan broj T**. Cilj je dobiti broj T koji je unutar željenog opsedga za adresu. $n = \lceil \log_v B \rceil$ gde je n dozvoljeni broj cifara.
3. Pseudoslučajan broj T (dobijen u koraku 2) možda još uvek nije direktna adresa baketa. Ovaj korak ga "skalira" ili "modulira" tako da padne unutar opsega od 1 do B (ukupan broj baketa u datoteci).
4. Pretvaranje relativne u mašinsku adresu.

ključ u broj $\rightarrow$ broj u "raspršen" broj $\rightarrow$"raspršen" broj u relativnu adresu $\rightarrow$ relativna adresa u fizičku

Tri često upetrebljivane metode su

1. Metoda ostatka pri deljenju $A = 1 + k(S) \pmod B$

   Ključ sloga se deli fiksnim brojem m (kod izbora m najpogodnije bi bilo da je to prost broj ili neparan broj sa velikim prostim činiocima). Adresa sloga = ostatak pri toj podeli (ključ % m).To omogućava ravnomernu (približno slučajnu) raspodelu slogova po memoriji. Slogovi sa sukcesivnim vrednostima ključa iz paketa dobijaju adrese fizički susednih baketa.

2. Metoda centralnih cifara kvadrata ključa

   Ključ sloga se prvo kvadrira, a zatim se uzimaju centralne cifre tog kvadrata (sredina broja). Uzima se onoliko centralnih cifara kvadrata vrednosti ključa koliko pozicija treba da sadrži relativna adresa(formira se T). Te centralne cifre predstavljaju adresu sloga u fajlu.

3. Metoda preklapanja

Ključ sloga se podeli na više delova (recimo na jednake grupe cifara),a zatim se ti delovi saberu (direktno ili pomoću OR, XOR...). Dobijeni zbir (ili kombinacija) predstavlja adresu sloga u fajlu. Cifre ključa premeštaju se kao pri savijanju, tj. preklapanju hartije. Pogodna za primenu kada je broj pozicija vrednosti ključa p mnogo veći od broja pozicija relativne adrese n. Preklapanje se izvodi po osama koje zdesna u levo određuje broj pozicija n relativne adrese.

Karakteristika probabilističke transformacije

Sinonimi-- slogovi koji transformacijom dobiju iste relativne adrese.

Matični baket je baket čija relativna adresa predstavlja rezultat transformacije, slogovi se uvek smeštaju u matični baket dok se ne popuni. **Primarni slog** je slog koji je smešten u matični baket. **Prekoračilac** je slog koji ne može bitn smešten u matični baket usled njegove popunjenosti.

Verovatnoća pojave sinonima zavisi od raspodele vrednosti ključa unutar opsega dozvoljenih vrednosti, odabrane metode transformacije i faktora popunjenosti memorijskog prostora. Broj prekoračilaca će biti manji što su slogovi ravnomernije raspoređeni po baketima, što je faktor popunjenosti manji i što je faktor baketiranja veći. U praksi se bira $q \leqslant 0{,}8$ (faktor popunjenosti) i $b \leqslant 10$ (factor baketiranja).

Prekoračioci se mogu rešiti na tri načina: smeštaj svih prekoračilaca unutar jedinstvenog adresnog prostora, smeštaj svih prekoračilaca u posebnu zonu adresnog prostora(zona prekoračenja) i kombinacija prethodna dva.

Formiranje rasute datoteke

1) inicijalno alociranje prazne datoteke- unapred se rezerviše mesto koje će fajl da koristi. Izbor prepoznavanja slobodnih lokacija unutar baketa se izvršava putem statusnog polja, upisom specijalnih znakova ili vođenjem indeksa slobodnih lokacija.
2) upisivanjem slogova u rasutu datoteku- radi se prema opštem postupku, sekvencijalnim učitavanjem(ako se koristi vodeća datoteka) ili direktnim upisom kada znamo adresu.

A. Formiranje u jednom prolazu- slogovi se upisuju u hronološkom redosledu nastanka
B. Formiranje u dva prolaza- slogovi se učitavaju iz vodeće datoteke i upsiuju u rasutu. U prvom prolazu se upisuju samo oni slogovi koji će biti smešteni u matične bakete. U drugom prolazu se upisuju preostali slogovi. Ovaj princip ima smisla kod datoteka sa jedinstvenim adresnim prostorom.

Traženje sloga u rasutoj datoteci

Traženje logički narednog = slučajno odabranog sloga

Vrši se metodom transformacije argumenta u adresu baketa. Ako se slog ne nađe u matičnom baketu, a matični baket ima prekoračilaca, traženje se nastavlja.

Vrste statičkih rasutih organizacija

- rasute datoteke s jedinstvenim adresnim prostorom(sa fiksnim korakom il isa slučajno odabranim korakom)
- rasute datoteke sa zonom prekoračenja(sa serijskom ili spregnutom zonom prekoračenja)

Rasuta s linearnim traženjem prekoračilaca s fiksnim korakom $k = 1$

Ukoliko je matični baket popunjen, slog se smešta u prvu narednu slobodnu lokaciju.

Traženje slučajno odabranog sloga-prekoračioca se vrši linearnom metodom. Upis je na prvu slobodnu lokaciju iza matičnog baketa. Za brosanje postojećeg sloga, ako je logičko potrebna su tri statusna sloga- aktuelan, naektuelan i slobodna lokacija. Za fizičko se, ako postoje prekoračioci, svi prekoračioci pomeraju za jednu poziciju prema matičnom baketu. Glavni nedostaci ovog princima su: efekat nagomilavanja prekoračilaca, neefikasno traženje i neefikasno neuspešno traženje.

Rasuta s linearnim traženjem prekoračilaca s fiksnim korakom $k > 1$

k i B moraju biti uzajamno prosti brojevi kako bi se obezbedio, siguran obilazak svih mogućih baketa.

Ovaj način odlaže nagomilavanje prekoračilaca.

Rasuta s linearnim traženjem prekoračilaca sa slučajno odabranim korakom $k \geq 1$

k se određuje na slučajan način i predstavlja rezultat druge probabilističke transformacije, primenjene na vrednost identifikatora – ključa sloga. k i B moraju biti uzajamno prosti brojevi- razlog isti kao i gore. Ovaj način odlaže nagomilavanje prekoračilaca.

Rasute s otvorenim načinom adresiranja su pogodne za upotrebu u slučaju manje popunjenosti, q $\leqslant$ 0,7

Rasuta sa sprezanjem u primarnoj zoni

Ukoliko je matični baket popunjen, slog se smešta u prvu slobodnu lokaciju iz lanca slobodnih lokacija. Postoji specijalan(nulti) baket koji pokazuje gde lanac počinje. Svi sinonimi (slogovi sa istom adresom) se povezuju u lanac. Matični baket sadrži pokazivač na prvi u lancu, a svaki naredni sadrži pokazivač na sledeći.

Traženje slučajno odabranog sloga se vrši transformacijama vrednosti ključa i pristupanje matičnom baketu.

Upis novog sloga je na prvu praznu lokaciju iz lanca baketa sa slobodnim lokacijama.

Brisanje postojećeg sloga je fizičko oslobađanje lokacije, uz eventualno vraćanje baketa u lanac baketa sa slobodnim lokacijama.

Glavna svrha ovog principa je izbegavanje efekata neefikasnosti pri traženju zato što se traži samo u baketima koji pripadaju istom skuou sinonima. Negativno kod ovoga je komplikovana fizička struktura.

Rasuta sa sprezanjem u zoni prekoračenja

Uvodi se spregnuta datoteka-zona prekoračenja. Ukoliko je matični baket popunjen, slog se smešta u prvu slobodnu lokaciju iz lanca slobodnih lokacija u zoni prekoračenja. Postoji specijalan(nulti) baket koji pokazuje gde lanac počinje. Vrši se sprezanje svih prekoračilaca. Tipičan faktor blokiranja f = 1(mala je verovatnoća da se dva prekoračioca iz istog lanca sinonima nađu u susednim lokacijama).

Traženje slučajno odabranog sloga vrsi se transformacijom vrednosti ključa u adresu i pristupanje matičnom baketu. Praćenje lanca prekoračilaca, započinjući od pokazivača na početak lanca u matičnom baketu.

Upis novog sloga je u matični baket ako ima mesta, tj. u lanac prekoračilaca ako nema.

Brisanje postojećeg sloga je fizičko oslobađanje lokacije, uklanjanjem sloga iz matičnog baketa uz, eventualno, prebacivanje prvog prekoračioca u matični baket.

Glavna svrha ovog principa je izbegavanje efekata neefikasnosti pri traženju i uklanjane efekta nagomilavanja prekoračilaca.

Rasuta sa serijskom zonom prekoračenja

Uvođenje zone prekoračenja – serijska datoteka. Ukoliko je matični baket popunjen, slog se smešta u prvu slobodnu lokaciju u serijskoj zoni prekoračenja.

Ocenu traženja sloga u rasutoj datoteci stvarno nmg...

Obrada rasute datoteke sa probabilističkom transformacijom

Nisu pogodne kao osnovne vodeće datoteke, mada mogu da se koriste kao: obrađivane ili vodeće, ali samo u režimu direktne obrade. Nisu pogodne kao vodeće u režimu redosledne obrade, jer fizička struktura nema informaciju o logičkom redosledu. Mogu se obrađivati i u režimu redosledne i u režimu direktne obrade. Performanse redosledne i direktne obrade rasute datoteke su iste.

Korisno je u svim mrežnim SUBP. Prednost: mali očekivani broj pristupa pri traženju slučajno odabranog sloga. Mana: potreba da se unapred odredi veličina datoteke i broj pristupa pri traženju može biti nepredvidivo velik.

Dinamička rasuta organizacija datoteke

Vrste rasutih dinamičkih datoteka:

- rasute datotekeke koje se mogu širiti

- dinamičke rasute datoteke
- linearne dinamičke rasute datoteke

Sve ove vrste imaju zajedničke osobine: metoda transformacije h koja ne zavisi od veličine prostora dodeljenog datoteci i ne menja se zbog upisa novih slogova, rezultat primene h na vrednost ključa i dužina d i broj baketa B se povećavaju i smanjuju dinamički.

Dinamička rasuta datoteka je sastavljena iz dva dela

1. adresar-sadrži niz pokazivača dužine 2^d sa adresama baketa i druga polja
2. zona podataka sa baketima-svaki baket sadrži zaglavlje i bar jedan slog. Zaglavlje baketa sadrži: polje d' ($0 \leqslant d' \leqslant d$) sa lokalnom dužinom vrednosti transformacije i polje m sa brojem aktuelnih slogova u baketu. d' govori koliko istih bitova najveće težine vrednosti transformacije vt moraju imati svi slogovi u baketu

Adresar i zona podataka se realizuju u dve posebne datoteke

Adresar

Najčešće se realizuje kao linearna struktura koja sadrži jednodimenzionalni niz od 2^d ($\geqslant B$) pokazivača ka baketima u zoni podataka, gde je B broj aktuelnih baketa u zoni podataka, promenljivu d(broj bitova koji se trenutno koriste za indeksiranje niza pokazivača) i faktor baketiranja b. $d = \lceil \log_2 B \rceil$. Zahteva mali kapacitet memorijskog prostora(ceo adresar se može smestiti u OM). Indeksi niza pokazivača se ponekad predstavljaju kao kompletno binarno stablo u čijim listovima se nalaze pokazivači ka baketima.

Veza između adresara i zone podataka

Adresar radi kao neka vrsta **mape**. On se koristi da, kada znamo transformacionu vrednost sloga (njegov “ključ”) odemo u adresar i nađemo **pokazivač na baket**. Dakle, **adresar** glumi **vezu ili most** između ključa (podatka) i njegove fizičke lokacije u **zoni podataka**. Zone podataka su smeštene u baketima.
Adresar zna: u koji baket da te uputi na osnovu transformacionih bitova. Ako više indeksa pokazuje na isti baket, to znači da sve te vrednosti dele iste gornje bitove. Ako se baket podeli, povećava se d, pa se adresar prilagodi da sve više indeksa pokazuje sve više baketa.

Generisanje vrednosti transformacije

Ako bismo jednostavno uzeli sam ključ kao indeks, sve slične ključeve (recimo 1, 2, 3...) dobili bismo u istom ili susednom baketu. To povećava **sudare** (više slogova u istom baketu). Da to sprečimo, primenjujemo **netrivijalnu transformaciju:** Ključ k prolazi kroz funkciju h(k) koja ga "razbacuje".Da sve bude još nasumičnije, ponekad se i **invertuje redosled bitova** u rezultatu. Metoda transformacije (funkcija h) se ** bira slučajno ili pseudoslučajno**, ali da daje

ravnomernu raspodelu pošto dinamičke rasute datoteke ne poseduju mehanizam za smeštaj prekoračilaca.

Formiranje dinamičke rasute datoteke

Formiranje dinamičke rasute datoteke u režimu direktne obrade vrši se tako što se ključevi ulaznih slogova prvo transformišu, pa se od tog rezultata uzima d bitova najveće težine i oni se koriste kao indeks u jednodimenzionalnom nizu pokazivača (adresaru). Preko tog indeksa dolazi se do adrese baketa, koji se zatim učitava u operativnu memoriju. Ako u baketu već postoji slog sa istim primarnim ključem, upis se prekida, ali ako takav slog ne postoji, on se upisuje u baket. Dakle, svaki upis u baket prethodi neuspešnom pretraživanju, da se ne bi uneli duplikati.

Postoje tri slučaja upisa:

1. Prost upis- izvršava se u baketu sa m < b slogova gde se novi slog upisuje u prvi slobodnu lokaciju u baketu, a broj zauzetih lokacija m se povećava za jedan
2. Upis koji dovodi do deljenja baketa i udvostručavanja dužine niza pokazivača u adresaru
3. Upis koji dovodi do deljenja baketa

Traženje sloga u dinamičkoj strukturi

Traženje logički narednog i slučajno odabranog sloga vrši se korišćenjem istog algoritma.

Koraci algoritma za traženje slučajno odabranog sloga:

- vrednost ključa k se podvrgava transformaciji h i rezultat te transformacije se pretvara u vrednost transformacije dužine d_{max} bita
- u adresaru se pronalazi adresa baketa, u kojem bi traženi slog trebalo da bude
  Sve se radi u OM, disku se pristupa jedino pri čitanju baketa.

Ažuriranje dinamičke rasute datoteke

Upis novog sloga- u najnepovoljnijem slučaju upis zahteva

- dva pristupa datoteci, ako upis ne dovodi do prepunjenja baketa (jedan pristup za neuspešno traženje i drugi za upis baketa)
- tri pristupa datoteci, ako upis dovodi do prepunjenja baketa (jedan pristup za neuspešno traženje i drugi i treći pristup za upis polaznog i jednog novog baketa u datoteku)

Brisanje postojećih slogova

Ažuriranje dinamičke rasute datoteke vrši se u tri slučaja brisanja.

Prvi slučaj je prosto brisanje, kada u baketu ima m > 1 slogova i kada je ukupan broj njegovih slogova i slogova njegovog prijatelja veći od b; u tom slučaju, posle uspešnog traženja, svi

slogovi u baketu koji su smešteni posle sloga koji se briše pomeraju se za jednu poziciju u levo, a parametar m se smanjuje za jedan. Ako se briše poslednji slog u baketu, m se jednostavno smanjuje za jedan, ali posle toga se učitava prijatelj baketa da bi se proverio ukupan broj slogova u ta dva baketa.

Drugi slučaj brisanja jeste spajanje susednih baketa, bez uticaja na veličinu adresara; to se dešava kada posle brisanja sloga ukupan broj slogova u dva baketa prijatelja nije veći od b. Tada se baketi spajaju, u spojenom baketu lokalna vrednost transformacije postaje d' = d' - 1, parametar m u baketu dobijenom spajanjem postaje m <= b, a svi elementi niza pokazivača koji su pokazivali na bakete prijatelje pre spajanja sada pokazuju na baket dobijen spajanjem.

Treći slučaj brisanja u dinamički rasutoj organizaciji datoteke jeste spajanje susednih baketa sa smanjenjem dužine niza pokazivača na pola. To se dešava kada na svaki baket u adresaru pokazuju najmanje po dva pokazivača. Tada se u adresaru vrednost transformacije d umanjuje za jedan, pa se samim tim dužina niza pokazivača smanjuje na pola. To znači da se svaka dva pokazivača, čiji se indeksi razlikuju samo u najmanje značajnom bitu, spajaju u jedan, pa na taj način dolazi do objedinjavanja baketa i smanjenja veličine adresara.

Dinamičke (kao i statičke) rasute datoteke su nepogodne za korišćenje u ulozi osnovne vodeće datoteke,ali se mogu koristiti kao obrađivane i vodeće u režimu direktne obrade. Ne mogu se koristiti kao vodeće u režimu redosledne obrade pošto im fizička struktura ne sadrži podatke o logičkoj strukturi podataka. Mogu se obrađivati i u režimu redosledne i u režimu direktne obrade.

Ne proizvodi slogove prekoračioce, datoteka se širi i skuplja u zavisnosti od broja aktuelnih slogova, broj pristupa pri traženju ne zavisi od veličine datoteke.

9. Indeks-sekvencijalna organizacija datoteke

Indeksne datoteke karakteriše postojanje posebne pomoćne strukture podataka – indeksa, koji je najčešće realizovan u posebnoj datoteci u obliku stabla traženja. Indeks sadrži parove u formi (vrednost ključa, relativna adresa sloga ili bloka), i koristi se za brz pristup kako slučajno odabranom slogu, tako i logički narednom slogu. Sâm podatak (slog) se nalazi u posebnoj datoteci koja predstavlja primarnu zonu. Vrste indeksnih datoteka:

1. statičke- istorijski prve, statička alokacija i statički indeks
2. dinamičke- dinamička alokacija i dinamički indeks koji se ažurira paralelno sa ažuriranjem zone podataka.

Statička indeks-sekvencijalna datoteka

1. Primarna zona- sadrži slogove koji su uređeni u skladu sa rastućim vrednostima ključa i grupisani u blokove, pri čemu je poželjno da faktor blokiranja bude što veći radi efikasnijeg korišćenja memorije. Ova zona se kreira u fazi formiranja statičke indeks-sekvencijalne datoteke i nikada se naknadno ne ažurira. Cilj primarne zone je da omogući korišćenje prednosti sekvencijalne organizacije prilikom redosledne obrade podataka, ali da se istovremeno izbegnu loše performanse koje bi nastale ako bi sekvencijalno organizovana datoteka morala da se često ažurira.

2. Zona indeksa- puno stablo traženja, spregnuta struktura reda n ($n \geqslant 2$), visine h gde je čvor stabla = blok i sadrži od 1 do n elemenata. Elementi u čvoru uređeni su saglasno rastućim vrednostima ključa, pri čemu je čvor sekvencijalno organizovana struktura.
   Zona indeksa u indeks-sekvencijalnoj organizaciji sadrži retko popunjeni indeks koji se organizuje u obliku stabla, pri čemu se reprezentativne vrednosti ključa iz svakog bloka primarne zone propagiraju kroz nivoe stabla. Ključevi koji se nalaze u stablu predstavljaju najmanje ili najveće vrednosti ključeva iz svakog bloka primarne zone, u zavisnosti od implementacije. Listovi stabla sadrže po jednu takvu reprezentativnu vrednost ključa iz svakog bloka, dok čvorovi na višim nivoima hijerarhije sadrže po jednu vrednost ključa iz svakog njima direktno podređenog čvora.
   Vrste zone indeksa:
   - zona indeksa s propagacijom najvećih vrednosti ključa iz svakog bloka- u slučaju poslednjeg bloka, propagira se ne aktuelna najveća vrednost ključa, već najveća dozvoljena vrednost ključa (chat je dao dobar primer pogledaj)
   - zona indeksa s propagacijom najmanjih vrednosti ključa iz svakog bloka- u slučaju prvog bloka, propagira se ne aktuelna najmanja vrednost ključa, već najmanja dozvoljena vrednost ključa
3. Zona prekoračenja- dodatna komponenta indeks-sekvencijalne organizacije datoteke i služi za smeštanje slogova koji ne mogu da se upišu u primarnu zonu. Svaki blok primarne zone može imati sopstvenu zonu prekoračenja. Prilikom pretrage, koristi se stablo koje mapira vrednosti ključa na adrese blokova, i ukoliko ključ slogova k(S) pripada granici između dva bloka (tj. veći je od ključa prethodnog a manji ili jednak sledećem), traženje se usmerava na odgovarajući blok ili njegovu zonu prekoračenja. Ako blok u primarnoj zoni **nije kompletan** (m < > n), novi slog se smešta pomeranjem postojećih slogova u bloku. Kada je blok **potpuno popunjen** (m = n), novi slog se mora upisati tako što se jedan od postojećih slogova izmešta u zonu prekoračenja (konkretno onaj čija je vrednost ključa veća od novog), a novi se smešta na odgovarajuću poziciju. Ako je novi ključ veći od maksimalnog u bloku, slog se direktno dodaje u zonu prekoračenja. Mogu se svrstati u lance spregnutih prekoračilaca gde svaka primarna zona ima najviše jedan. Mogu bit direktno i indirektno povezani sa listom stabla traženja.

Indeks-sekvencijalna metoda pristupa

Obezbeđuje sekvencijalni, direktni i dinamički način pristupa indeks-sekvencijalnoj datoteci.

Formiranje IS datoteke- program redosledno učitava slogove ulazne sekvencijalne datoteke i smešta blokove u primarnu zonu IS datoteke. Već formirana sekvencijalna datoteka proglašava

se primarnom zonom IS datoteke. Podrazumeva dva osnovna koraka: formiranje zone indeksa i formiranje zone prekoračenja.

Zona indeksa se gradi iterativno, po nivoima stabla traženja, od dna ka vrhu i s leva na desno. Najpre se formiraju svi listovi – čvorovi najnižeg nivoa (nivoa h), zatim čvorovi nivoa h-1, pa sve do čvorova nivoa 1. U svaki čvor na i-tom nivou hijerarhije (gde je i = h−1, h−2, ..., 1), upisuju se najveće (ili alternativno najmanje) vrednosti ključeva iz n sukcesivnih čvorova sa nivoa ispod. Kod propagacije najvećih vrednosti, u poslednji element krajnjeg desnog čvora upisuje se maksimalna dozvoljena vrednost ključa, dok se kod propagacije najmanjih vrednosti u prvi element krajnjeg levog čvora upisuje minimalna dozvoljena vrednost ključa.

Nakon toga sledi **formiranje zone prekoračenja**, koja se inicijalno alocira kao prazna i čiji se svi blokovi povezuju u lanac slobodnih blokova. Pokazivač na početak tog lanca beleži se u zaglavlju zone prekoračenja, čime je spremna za kasniji upis slogova koji ne mogu da budu smešteni u primarnu zonu.

Traženje logički narednog sloga se vrši kombinovanom primenom principa linearnog traženja i metode praćenjem pokazivača. Počinje u prvom bloku primarne zone i svako naredno traženje se nastavlja od tekućeg sloga datoteke u bloku primarne zone(linearna metoda). Po dolasku do poslednjeg sloga bloka traženje se nastavlja u lancu prekoračilaca, ako postoji, i to metodom praćenja pokazivača

Direktno povezivanje prekoračilaca- pristupa se blokovima primarne zone, prekoračiocima i listovima stabla traženja. Broj pristupa R i pri uspešnom i pri neuspešnom traženju jednog logički narednog sloga(ima formula)

Indirektno povezivanje prekoračilaca- nastavak traženja direktno u zoni prekoračenja

Traženje slučajno odabranog sloga - vrši se praćenjem pokazivača u stablu pristupa, počevši od korena ka listu stabla traženja. Tokom pretrage uvažava se da li je stablo organizovano propagacijom maksimalnih ili minimalnih vrednosti ključa iz svakog bloka. Po dolasku do odgovarajućeg elementa u listu, donosi se odluka da li se traženje nastavlja direktno u bloku primarne zone ili praćenjem lanca prekoračilaca u zoni prekoračenja – u slučaju **direktnog povezivanja prekoračilaca**. U slučaju **indirektnog povezivanja**, pokazivač iz lista vodi do odgovarajućeg bloka u primarnoj zoni, odakle se po potrebi dalje nastavlja pretraga kroz lanac prekoračilaca.

Obrada IS datoteke omogućava efikasno izvođenje i redosledne i direktne obrade, što je čini pogodnom za korišćenje u ulozi vodeće datoteke u oba režima. Redosledna obrada nad vodećom datotekom sa Nv slogova odvija se naizmeničnim pristupanjem blokovima primarne zone i pripadajućim lancima prekoračilaca. Pristup se započinje od adrese prvog bloka primarne zone, koja je smeštena u zaglavlju datoteke.

Obrada IS datoteke pri redoslednoj obradi podrazumeva izračunavanje ukupnog broja pristupa R_{uk}, koji zavisi od načina povezivanja prekoračilaca. U slučaju **direktnog povezivanja**, ukupni broj pristupa jednak je zbiru broja blokova primarne zone B, broja blokova u zoni prekoračenja Z, i približnog broja prekoračilaca po bloku ⌈B/n⌉, pa važi formula:

$R_{uk}=B+Z+⌈Bn⌉$

U slučaju **indirektnog povezivanja**, broj dodatnih pristupa za prekoračilace se ne računa posebno, pa je:

$R_{uk}=B+ Z$

Očekivani broj pristupa pri pronalaženju jednog sloga izražava se kao srednja vrednost ukupnih pristupa po broju slogova Nv. Za direktno povezivanje to je:

$R=B+Z+⌈Bn⌉/Nv$

a za indirektno povezivanje:

$R=B+Z/Nv$

Redosledna obrada je **nešto efikasnija** kod datoteka sa **indirektnim povezivanjem prekoračilaca**, mada je kod uobičajenih vrednosti reda stabla nn ta razlika u efikasnosti **neznatna**.

Direktna obrada IS datoteke vrši se pomoću vodeće datoteke sa ukupno Nv slogova, koji se sastoje od slogova u primarnoj zoni i slogova u zoni prekoračenja. Očekivani broj pristupa pri traženju slogova zavisi od verovatnoće uspešnog i neuspešnog traženja, kao i od načina povezivanja prekoračilaca. U poređenju sa indirektnim povezivanjem, direktna obrada može biti nešto efikasnija, ali je ta razlika obično mala.

Ažuriranje IS datoteke se vrši u režimu direktne obrade.

Upis novog sloga- Ako se neuspešno traženje zaustavilo u bloku primarne zone vrši se pomeranje slogova sa većom vrednošću ključa od vrednosti ključa novog sloga za jednu lokaciju ka kraju bloka.a slog sa do tada najvećom vrednošću ključa u bloku upisuje se u zonu prekoračenja. Ako se neuspešno traženje zaustavilo na nekom od prekoračilaca, novi slog se upisuje u prvu slobodnu lokaciju i uvezuje se sa ostalim prekoračiocima.

Brisanje sloga- logičko(češće) i fizičko brisanje- važi sve od ranije

Reorganizacija IS datoteke neophodna je zbog degradacije performansi koje nastaju vremenom usled čestih upisa slogova u zonu prekoračenja i logičkog brisanja slogova. Reorganizacija se sprovodi periodično kako bi se uklonile negativne posledice ažuriranja. Postupak uključuje ponovno formiranje primarne zone (redoslednim čitanjem slogova iz postojeće primarne i prekoračene zone), generisanje novog stabla traženja i formiranje nove zone prekoračenja.

Interval između dve reorganizacije može biti **fiksan** (npr. jednom mesečno) ili **dinamički određen** na osnovu stepena popunjenosti zone prekoračenja (npr. kada popunjenost pređe 80%). Korišćenjem **distribuiranog slobodnog prostora** unutar blokova (npr. popunjavanje blokova do 60%–80%) unapred se ublažava potreba za reorganizacijom, jer se ostavlja mesta za nove slogove. Ovo produžava vremenski interval između reorganizacija.

IS datoteka se može ažurirati i u sekvencijalnom režimu, pri čemu se stablo traženja i zona prekoračenja formiraju iz početka, kao rezultat potpunog prenosa sadržaja i regeneracije strukture.