

# 1. OSNOVNI POJMOVI

## REALNI SISTEMI I INFORMACIONI SISTEMI

Realni sistem (RS) – sistem (strukturirani skup objekata), kao deo realnog sveta ima:

- cilj
- resurse (podaci, vreme, ljudi)
- procese (aktivnosti)
- strukturu
- okruženje (svi imaju npr. banku)

Sistemi:

1. realni
2. apstraktni – opisujemo ih apstraktnim mehanizmima, putem matematičkih struktura jer je tako lakše upravljati sistemom

## INFORMACIONI SISTEM (IS)

- model RS
- pao sistem – **softver** ne radi
- obezbeđuje informacije za upravljanje RS
- infrastrukturna komponenta

## **ZADACI IS**

- prikupljanje pod.
- skladištenje pod.
- prenos pod.
- prezentovanje pod.
- obrada pod.
- automatizacija upravljačkih funkcija u RS (greške – izbegavamo ljudski faktor)

## **ČINIOCI IS**

- računarsko-komunikaciona i softverska infrastruktura
- BP
- app. za rad sa pod.
- dokumentacija – projektna i korisnička
- krajnji korisnici
- tim za obezbeđenje eksploatacije i održavanja

**REALNI ENTITET** – 1 činilac poslovanja u RS, resurs, pojedinačno (konkretni student – Nađa)

**KLASA REALNIH ENTITETA** – skup entiteta koji poseduju zajedničko svojstvo, imaju iste osobine (studenti)

$$E = \{e_i \mid P(e_i)\}$$

**POVEZNIK (VEZA)** - odnos 2 ili više E ili prethodno uspostavljenih P

**KLASA POVEZNIKA** – skup veza između klasa E ili prethodno identifikovanih klasa P

$S = \{e_1, \dots, e_m \mid P(e_1, \dots, e_m)\}$ ,  $e_i$  – entiteti, P predikat – zajedničko svojstvo klase

### **OBELEŽJE (ATRIBUT)**

- nazivi su logični
- JMBG, Ime, Prz
- osobine klase E, P
- vrste: elementarno, složeno, skupovno

### **DOMEN**

- specifikacija skupa mogućih vrednosti obeležja (definisane su relacije i operacije nad skupom)
- vrste – predefinisani, korisnički definisani
- domen obeležja – svako obeležje mora imati domen

specifikacija domena:

$DOCENA ::= \{d \in N \mid d \geq 5 \& d \leq 10\}$

$Dom(A), (A : D)$  – obeležju A je pridružen domen D

$Dom(Ocena) = DOCENA$  ili  $(Ocena : DOCENA)$

$dom(A)$  – skup mogućih vrednosti obeležja

$dom(Ocena) = \{5, \dots, 10\}$

### **PODATAK – (ENTITET, OBELEŽJE, VREME, VREDNOST)**

- npr. (student, ocena, ako nije navedeno onda trenutak kada se manipuliše podatkom, 10 – iz  $dom(A)$ )

- uređena četvorka
- činjenica iz RS
- 10 nije podatak bez KONTEKSTA
- smisaona (semantička) komponenta (ENTITET, OBELEŽJE, VREME)

### **TIP ENTITETA – TE**

- model klase realnih entiteta u IS, biramo relevantna obeležja

1.  $Q = \{A_1, \dots, A_n\}$  – skup obeležja,  $Q \subseteq$  skupa obeležja klase realnih E
2. N – naziv

- pr. Radnik({Mbr, Ime, Prz, Zan})

### **POJAVA TIPIA ENTITETA – PTE**

- model 1 realnog E u IS
  - konkretne vrednosti koje se dodeljuju obeležjima koje ima TE
- PTE:  $p(N) = \{(A_1, a_1), \dots, (A_n, a_n)\}$  – obeležje  $A_i$  dobija vrednost  $a_i$  iz  $dom(A)$

- ako se uvede redosled u niz obeležja ( $A_1, \dots, A_n$ ) tada je PTE uređena **torka** ( $a_1, \dots, a_n$ )
- 1 red tabele, reprezentuje 1 realan E: (1040, Nađa, Đordan, Programer)

**IDENTIFIKATOR TE** - skup obeležja – jednoznačno identificuje svaku PTE ili torku

- identifikator PTE - bilo koja vrednost identifikatora TE koja označava max 1 PTE

- vrste:

- **eksterni** – skup obeležja nije  $\subseteq$  skupa obeležja TE (nešto dodatno, npr. 1. klupa 1. mesto)
- **interni** –  $\subseteq$  skupa obeležja TE (biramo one koji će jedinstveno da identifikuju svaku PTE)

**KLJUČ TE** - minimalni interni identifikator TE

- **minimalni skup obeležja sa svojstvom jedinstvene identifikacije svake PTE/torke**

- **minimalni skup** – ne postoji  $\subseteq$  od skupa ključeva koji može jedinstveno da identificuje PTE
- **jedinstvena identifikacija** – ne postoje 2 PTE s istom vrednošću ključa i sva obeležja koja čine ključ moraju imati zadatu vrednost (nisu NULL – nedostajuća vrednost)

$$X \subseteq Q, Q = \{A_1, \dots, A_n\}$$

- pr. IN11/2021 unutar realnog sistema FTN, min skup obeležja ima 3 obeležja, najčešće je 1
- svaki TE poseduje **bar 1** ključ

**STRUKTURA TE N(Q, C)**

- N – naziv TE
- Q = { $A_1, \dots, A_n$ } – skup obeležja TE
- C – skup ograničenja TE
- K = { $K_1, \dots, K_n$ }  $\subseteq$  C, K  $\neq \emptyset$ , uvek ima makar 1 ključ (primarni)

1. Radnik({Mbr, Ime, Prz, JMBG}, {Mbr, JMBG}) – 2 **ekvivalentna** ključa - biramo samo 1 **PRIMARNI KLJUČ** (Mbr).
2. Radnik({Mbr, Ime, Prz, JMBG}, {Mbr + JMBG}) – ključ iz više obeležja

**TIP POVEZNIKA – TP** – povezuje dva ili više TE, ili prethodno definisanih TP

- model veza između pojava povezanih TE ili TP, odnosno između realnih entiteta ili veza

**STRUKTURA TP – N(N<sub>1</sub>, N<sub>2</sub>, ..., N<sub>m</sub>, Q, C)**

- N – naziv TP
- N<sub>i</sub> ( $i \in \{1, \dots, m\}$ ) – povezani tip – TE ili prethodno def. TP; rekurzija – povezuje E iste klase

- $Q = \{B_1, \dots, B_n\}$  - skup obeležja TP, može biti  $\emptyset$
- $C$  - skup ograničenja TP
- $K = \{K_1, \dots, K_k\} \subseteq C$  - skup ključeva TP ( $K \neq \emptyset$ ), ključ TP neće činiti obeležja TP

## **POJAVA TIPOVEZNika - PTP**

- $N(N_1, N_2, \dots, N_m, \{B_1, \dots, B_k\}, C)$
- reprezentuje 1 poveznik u RS
- predstavlja skup podataka:  $p(N) = (p_1, \dots, p_m)(N) = \{(B_1, b_1), \dots, (B_k, b_k)\}$ 
  - $b_i \in \text{dom}(B_i)$
  - skup svih pojava  $p(N)$  mora zadovoljavati skup ograničenja  $C$

**IDENTIFIKATOR TP** - niz  $(N_1, N_2, \dots, N_m)$ , jednoznačno identificuje bilo koje PTP

- identifikator PTP - bilo koja vrednost identifikatora TP -  $(p_1, \dots, p_m)$  označava max 1 PTP

## **KLJUČ TP**

- skup obeležja  $X$  izведен na osnovu ključeva povezanih tipova  $(N_1, N_2, \dots, N_m)$
- $X \subseteq K_1 \cup \dots \cup K_m$ , gde  $(\forall i \in \{1, \dots, m\})(K_i \text{ je 1 izabrani ključ povezanog tipa } N_i)$ ; unija ključeva povezanih tipova ili njen podskup
- $X = \{A_1, \dots, A_n\}$ , takav da
  - (1) ne postoje 2 PTP  $N$  s istom  $x$ -vrednošću (za  $X$ ) - svojstvo jednoznačne identifikacije
  - (2) ne postoji  $X' \subset X$ , za koji važi (1) - svojstvo minimalnosti

**STRUKTURE PODATAKA** - orijentisani graf  $G(V, \rho)$

$V$  – skup čvorova, podaci i zna se semantika

$\rho$  – skup grana,  $\rho \subseteq V \times V$  – binarna relacija, veze između podataka, zna se semantika

- vrste prema **nivou apstrakcije pridružene semantike**: logičke strukture obeležja, logičke strukture podataka, fizičke strukture podataka

## **LOGIČKE STRUKTURE OBELEŽJA - LSO**

- struktura nad **skupom TE, TP i njihovih obeležja**
- model dela realnog sistema  $M = (STE, RTE)$ , STE – skup tipova, RTE – relacija
- pristupi organizaciji:
  - (A) – skup čvorova =  $i$  TE i TP

(B) – skup čvorova = **TE**, skup grana = TP (povezuje samo TE – **mana**)

- nivo detaljnosti vizuelnog prikaza LSO: **globalni** prikaz – nivo TE i TP; **detaljni** prikaz – nivo obeležja

- moramo znati **semantiku**

## **LOGIČKA STRUKTURA PODATAKA – LSP**

- definiše se nad LSO
- u čvorovima i granama su **konkretni** podaci
- linearno uređena struktura pod.

**ŠEMA LSP** – LSO koja daje **KONTEKST** LSP-u

\***LSO**: Student (BRI, Ime, Prz) daje kontekst – **TE**, **TIP SLOGA** – lin. struk. skupa obeležja datog TE

\***LSP**: 1078, Nikola, Stojičić – **PTE**; torka, **SLOG** – lin. struk. nad skupom podataka 1 E

## **DATOTEKA**

- struktura pod. nad skupom slogova
- **nad skupom pojava**(PTE/PTP) **1 TE**
- LSO – TE nad kojom je definisana PTE

## **BAZA PODATAKA**

LSO **nad skupom pojava**(PTE/PTP) **više TIPOVA E i TIPOVA P** (množina)

**ŠEMA BP** – LSO koja daje kontekst BP

reprezentacija:

1. graf
2. tabele
  - veza više-vše -> 3. tabelu
  - veza 1-1 -> direktna propagacija ključa

**FSP** – smeštanje na memorijski medijum

## 2. KONCEPCIJA BAZA PODATAKA

### KLASIČNA ORGANIZACIJA DATOTEKA

- najstarije rešenje – papir
- naprednije rešenje – **sistem datoteka** – svaka grupa procesa (aplikacija) ima svoj skup datoteka

### **OSNOVNI NEDOSTACI:**

1. nepovezanost aplikacija – ručno prepisivanja istih ili sličnih pod.
2. redundantnost pod. – višestruko memorisanja istih ili sličnih pod.
3. čvrsta povezanost programa i pod. – u kodu se pisao i način fizičke smeštenosti na disk i način pristupa podacima
4. konkurentni pristup više korisnika

### RELACIONE BP

- rešenje – sve app. koriste **istu datoteku**
  - nema više redundantnosti, svi vide promene
  - problem – višekorisnički režim rada -> POLICAJAC (softver) – **SUBP** (sistem za upravljanje BP)
  - **transakcijska** obrada – **sve ili ništa**, ako pukne u nekom trenutku mora da se vrati na početno stanje (SUBP vodi računa o tome)
  - višekorisnički konkurentni pristup
  - automatizacija
- BP – na logičkom nivou su skup tabele, koje su povezane
- relacione BP su i dalje primarne u poslovanju

### NoSql BP

- baze koje nisu relacione
- problem: rade brzo, ali ne garantuju tačnost

### **Big Data:**

- problem je brzina odziva, zbog brzine smo se vratili na datoteke
- velika kol. pod.

- multimedijalni sadržaj
- iz gomile pod. -> generišemo info -> znanje
- Tipovi pod. prema strukturi:
  1. strukturirani – svi imaju isti format, u relacionim bazama
  2. polustrukturirani – mogu ali ne moraju da učestvuju svi elementi u strukturi (JSON, HTML)
  3. nestrukturirani – sa stanovišta čuvanja, nema definisan format (stranica sajta, mejl, post)

## **BP i SUBP**

SUBP – okružuje BP

- softver koji omogućuje da pod. vidimo na najvišem logičkom nivou kao tabele
- zaštita
- višekorisnički režim rada
- distribuirane BP – lokacijski udaljene, a SUBP daje uvid kao da su svi na 1 mestu
- obezbeđuje zadavanje šeme BP, preko šeme vidimo i radimo sa podacima

## **ŠEMA BP**

- program koji koristi usluge SUBP-a
- ne vodi računa o povezanosti programa i pod.
- SUBP -> zadatak da preslikava LSP (podaci) <--> FSP (kako su smešteni)
- rad u timu

case alati:

- softver za crtanje šeme i kontroliše nas
- konceptualnu šemu prevodi na relationalnu, a SUBP relationalni model kako se smešta na disk
- Sql skripta – jednim klikom se generiše šema BP napisana Sql jezikom
- uloge: projektant i administrator (mučenik, brine o performansama, šemi, SUBP-u)
- SUBP – rečnik podataka (data dictionary, repository – admin. prava) – čuva šemu BP u svoju bazu
- Sql developer – java projekat – zakači se za bazu i čita parče baze samog SUBP-a (sistemske tabele)
- web app. – sa Sql upitim se obraća direktno SUBP-u
- korisnički pod. – naša BP gde mi unosimo pod.

**PODŠEMA / EKSTERNA ŠEMA** – modeli na nivou apstrakcije obeležja

- **šema** - nezavisni programi od podataka pa ne brinemo o fizičkom smeštanju
- izmene u šemi, svi koriste podatke na 1 mestu, ali koriste samo neki deo podataka pa promena BP se ne reflektuju na app. jer app. ne uzima taj deo koji se menja

## **PODŠEMA**

- **LSO nad šemom**
- prilagođava se app, jer uzima **parče šeme BP** koje pokriva grupu procesa neke app.
- izdvajamo **iz šeme neke pod. koje odgovaraju app.**

SUBP – transformacija podšeme i šeme, bilo bi dobro da amortizuje vezu šeme i podšeme

1. logička nezavisnost programa od pod. – promene šeme ne izazivaju promene podšeme i programa
2. fizička nezavisnost programa od pod. – promene FSP ne izazivaju promene šeme, podšeme i programa

- pr. Student\_Fakultet {BRI, IME, PRZ, BPI, NAF}

## **POGLED** – modeli na nivou **apstrakcije pod.**

- **globalni pogled** – **pojava nad šemom BP** – nad svim podacima, u trenutnom stanju
- **pogled** – pojava nad **podšemom** – konkretni podaci, nad samo nekim

## **SISTEMI BP**

1. u užem smislu – 1 BP 1 instanca BP
  2. u širem smislu – više BP
- OLTP – sistem: pod. se arhiviraju posle 1 god. (FTN, banke,...)
  - NoSql – nestrukturirani podaci, velika brzina
  - ostvarivost ciljeva BP zavisi od:
    1. projektanta BP – problemi ako nije dobro isprojektovana, viškovi pod. su uvek loši u BP
    2. metode, tehnike, koncepti projektovanja
    3. SUBP – alati i cena

### 3. MODEL PODATAKA

#### MODEL PODATAKA – MP

- mehanizam uz pomoć kojeg preslikavamo RS u model
- **jezik** (matematička apstrakcija) kojim se modeluju šeme BP

#### **MP (S, I, O)**

- S – strukturalna komponenta
  - I – integritetna komp.
  - O – operacijska komp.
- nivoi apstrakcije:
1. nivo **intenzije** – LSO, konteksna, nivo **tipa**
  2. nivo **ekstenzije** – LSP, konkretizacija, nivo **pojave** tipa
- **meta** pod. – pod. o podacima, čuvaju se u rečniku pod.
- meta level 0 – LSP, FSP, BP (konkretna PTE)
  - meta level 1 – LSO, šema BP (daje kontekst, konkretni TE)
  - meta level 2 – opisuju se koncepti, MP (naziv, skup obeležja...)
  - meta level 3 – tipovi, pojave, MOF (jezik koji ima koncepte jezika za modelovanje)

#### STRUKTURALNA KOMPONENTA MP

- **KONCEPT** – apstraktna predstava 1 klase pojmove; služi za modeliranje LSO – šeme BP

- **primitivni koncept** – ne definiše se, uvodi se, predstavlja 1 klasu RS (npr. **vrednosti**, **domen**, **obeležje**,...)

sadrži:

1. skup **primitivnih** koncepata – skup osobina, pravila kako ih primeniti i semantike
2. skup **formalnih pravila** – **složeni koncepti** izgrđeni od primitivnih ili već predefinisanih složenih
3. skup **predefinisanih složenih** koncepata – osobine, pravila, **semantika** npr. **TP**

pr. TP:

- formira se korišćenjem drugih koncepata (TE, skup obeležja, naziv tipa,...)
- pravila:  $N(N_1, \dots, N_m, Q, C)$
- semantika: TP modeluje veze između klasa realnih entiteta ili prethodno uspostavljenih poveznika

## **INTEGRITETNA KOMPONENTA**

-modelovanje ograničenja

1. skup **tipova ograničenja**:
  - skup osobina (kako formalno da specificiramo, interpretacija – da li zadovoljava ili ne)
  - skup pravila za korišćenje
  - semantika
2. skup **formalnih pravila za izvođenje zaključaka** o važenju ograničenja
3. skup formalnih pravila za **kreiranje novih tipova ograničenja**

- ograničenja proverava SUBP

- ako se odnose na pod. ograničenja treba da budu uz njih ili user interface (jeftiniji npr.\*obavezno polje)

- pr. ograničenje domena, ključa, kardinalitet TP

## **OPERACIJSKA KOMPONENTA MP**

1. skup tipova operacija:

- skup **osobina** (kako formalno da specificiramo, izvršenja nad pod. – Sql)
- skup **pravila za korišćenje**
- **semantika**

npr. INSERT, DELETE, UPDATE,...

definiše:

1. **upitni** jezik - Query Language (QL) - iskazivanje upita (select) nad BP
2. jezik za **manipulisanje podacima** - Data Manipulation Language (**DML**) – ažuriranje BP
3. jezik za **definiciju podataka** - Data Definition Language (**DDL**) – kreiranje i modifikacija specifikacija

specifikacija operacije sadrži komponente:

1. **aktivnost** – akciju
2. **selekcija** – uslov nad kojim torkama

operacijska komponenta:

1. **proceduralna**
2. **deklarativna** – šta hoćemo, ne kako (Sql)

## 4. MODEL PODATAKA TIPOVA I ENTITETA I POVEZNIKA – ER MODEL

### OSNOVNI POJMOVI

- obeležje i domen
- TE, PTE
- TP, PTP

### STRUKTURALNA KOMPONENTA

- *primitivni* koncepti:

- **vrednost** – const
- **domen** – korisnički definisan (u teoriji nasleđuje prethodno kreiran domen ali SUBP nema nasleđivanje)
- **obeležje**

- *izvedeni* koncepti:

- podatak
- TE, PTE
- TP, PTP

## ER DIJAGRAMI

- služe za projektovanje **šeme BP**
- nema pod. o načinu implementacije
- semantički bogat, i kada se obogaćivao nastao je EER
- dijagrame svi mogu da **razumeju** pa se zato zadržao, razumevanje kako RS funkcioniše
- nivoi detaljnosti prikaza: **globalni** - nazivi tipova; **detaljni** nivo - nazivi obeležja
- **ER model - jezik za modelovanje pod. klase E ili P**

## **INTEGRITETNA KOMPONENTA**

Tipovi ograničenja u ER modelu podataka:

- ograničenje domena
- ograničenje vrednosti obeležja
- ograničenje pojave tipa
- kardinalitet tipa poveznika
- ograničenje ključa (integritet tipa) - za TE i TP

## OGRANIČENJE DOMENA

### SPECIFIKACIJA DOMENA

- struktura: **D(id(D), Predef)**
  - D - naziv domena
  - **id(D)** - ograničenje domena
  - Predef - predefinisana vrednost domena, umesto null, nije obavezna

### OGRANIČENJA DOMENA id(D)

- primenom pravila nasleđivanja se definiše ograničenje domena
- ograničenje nasleđenog domena je struktura: **id(D) = (Tip, Dužina, Uslov)**
  - Tip - tip podatka, obavezno (oznaka primitivnog domena ili prethodno definisanog kor. domena)
  - Uslov - logički uslov koji svaka vrednost domena mora da zadovolji
- pr. DOCENA((Number, 2,  $d \geq 5 \wedge d \leq 10$ ),  $\Delta$ )  $\rightarrow$  DPOZOCENA((DOCENA,  $\Delta$ ,  $d \geq 6$ ), 6)
- null** - nula (**nedostajuća**) vrednost -  $\omega$ , vrednost obeležja nedostaje - nije zadata
  1. **nepoznata** ali postojeća vrednost obeležja
  2. **nepostojeća** vrednost obeležja
  3. **neinformativna** vrednost obeležja

## OGRANIČENJE VREDNOSTI OBELEŽJA

- specifikacija obeležja, obeležje  $A \in Q$ , datog tipa N

- struktura: (**id(N, A)**, **Predef**)

- **id(N, A)** – ograničenje vrednosti obeležja
- Predef – ako se ne navede važeći je *Predef* odgovarajućeg *domena*

- **id(N, A) = (Domen, Null)** – oba su *obavezna*

- Domen – naziv domena pridruženom obeležju
- Null – da li obeležje sme ili ne sme da ima null vrednost

## **OGRANIČENJE POJAVE TIPOVIMA**

ograničenja na moguće vrednosti pod. unutar iste pojave TE ili TP

- **ograničenje važi i za TE i TP**

N: **id(N) = ({id(N, A) | A ∈ Q'}, Uslov)** – skup ograničenja vr. obeležja sa pridodatim logičkim uslovom

- N = TE:  $Q' = Q$
- N = TP:  $Q' = Q \cup K_p$ ,  $K_p$  skup obeležja primarnog ključa TP, a Q može biti  $\emptyset$
- Uslov – odnosi se na obeležja, i na 1 pojavu

## **KARDINALITET TIPOVA POVEZNIKA**

(a, b)

- $a \in \{0, 1\}$  – min kardinalitet
- $b \in \{1, N\}, N \geq 2$  – max kardinalitet ( $M : N, N : 1, 1 : 1$ )

- ograničava u koliko PTP može učestvovati 1, bilo koja POJAVA POVEZANOG TIPOA (TE ili prethodno def TP)

- definiše se za svaki povezani tip

**čitanje dijagrama:**

- (0, 1) – ne mora nijedan, ili max 1
- (1, 1) – tačno 1
- (0, N) – ne mora nijedan, a može i više
- (1, N) – bar 1, a može i više

- **rekurzivni** TP

- isti povezani tip sa obe strane
- povezuje entitete iste klase
- koristi se za neku hijerarhiju, delove organizacije

## **OGRANIČENJE KLJUČA**

- Integritet TE – ograničenje ključa

- Integritet TP – niz naziva povezanih tipova, ili njegov neprazan podniz; ograničenje ključa

3 opšte grupe **max kardinaliteta** koji utiču na **formiranje ključeva TP**:

- **M : N** - (Radnik, Projekat); Kp = Mbr+Spr -> **uniramo** ključeve povezanih tipova
- **N : 1** - (Radnik); Kp = Mbr -> uzima se ključ sa strane gde je **1**; podniz povezanih tipova
- **1 : 1** - (Radnik) i (PolisaOsiguranja); K1 = MBR i K2 = BrPol -> **biramo** ključ
- M : N i rekurzija - (Deo, Deo); Kp = DelD+DelDkom -> **preimenujemo** 1 obeležje jer moraju biti različiti

## **GERUND**

- glagolska imenica
- TE dobijen transformacijom TP
- TP koji predstavlja **povezani tip u nekom 2. TP**
- uloga i **TE i TP**
- simbol u simbolu
- TP  $N(N_1, N_2, \dots, N_m, \{B_1, \dots, B_k\}, C)$  i neka je neki  **$N_i$  TP**, tada je  $N_i$  gerund jer se ponaša kao **TE u odnosu na N**
- služi za modelovanje **kombinacija** koje ulaze u kombinaciju (kod 2 kombinacije taj 1 isti TE kod obe mora u implementaciji da se reši da bi se radilo o istom TE)

## **AGREGACIJA**

- objedinjuje složenije ER strukture (odvajamo deo dijagrama) – pa se ponaša **kao 1 TE**
- najjednostavniji primer je gerund

**N-armi TP** - može da povezuje više od 2 druga tipa, i određuje se kardinalitet za svaki povezani tip

## **SLABI TE**

- TE čije su **pojave zavisne od pojava nekog 2. TE**

- vrste zavisnosti slabih TE:

### **1. egzistencijalna**

- između pojava 2 TE
- min kardinalitet TP **(1, )**
- **vremensko prethodenje** – mora da postoji pojava pre neke 2. pojave
- pr. regularni TE: Radno\_mesto; slabi TE: Radnik, jer je egzistencijalno zavisan od TE Radno\_mesto (ako se ukine radno mesto, radnik gubi posao)
- **regularni TE** – TE koji **nije** u egzistencijalnoj zavisnosti

### **2. identifikaciona**

- podrazumeva egzistencijalnu

- i min i max kardinalitet TP prema slabom TE (1, 1)
- uvodi klasifikaciju TP: neidentifikacioni TP i identifikacioni TP
- štedimo je i korisitimo samo kada ima potrebe

## IDENTIFIKACIONI TP

geometrijski:

- bilo koja pojava id-zavisnog TE se identificuje vrednostima njegovih identifikacionih obeležja (Student: Brl, GodU) i vrednosti ključa nadređenog TE

- Identifikator id-zavisnog TE  $N_i$  (Student) (N, X)

N - naziv nadređenog TE (StudijskiProgram)

X - skup identifikacionih obeležja TE  $N_i \rightarrow \{Brl, GodU\}$

- Ključ id-zavisnog TE  $N_i$   $K_i = X \cup K$ , K - ključ nadređenog TE {IdStudPro}  $\rightarrow K_i = Brl + GodU + IdStudPro$

**SLAB TIP** – ima identifikaciona obeležja, nema svoj ključ jer se formira od ključa nadređenog tipa i od svojih id. obeležja

## IS-A HIJERARHIJA

- specifični TP i uvodi:

- superklasu – 1 klasa entiteta (entiteti sa zajedničkim osobinama, obeležjima), ima više podskupova sa specifičnim osobinama
- potklasu – set specifičnih osobina (atributi ili neke specifične veze) i ona je  $\subset$  početnog skupa

- semantički model podataka:

- specijalizacija – pojava superklase se specijalizuje na pojavu potklase
- generalizacija – pojava potklase se generalizuje na pojave superklase

- pojmovi superklase i potklase se uvode:

- da bi model RS bio semantički bogatiji
- da bi se izbegle nula vrednosti (ako bismo sve stavili u 1 TE i tada bismo imali puno null vrednosti)
- da bi se izbeglo definisanje TP koji nema mnogo smisla

- min kardinalitet (a, )

- 0 – parcijalna
- 1 – totalna

- max kardinalitet ( , b)

- 1 – nepresečna
- N – presečna

bitne karakteristike:

- nasleđivanje osobina superklase
- svaka **potklasa je identifikaciono zavisna od superklase** (npr. mora postojati Radnik gore, pa tek onda Projektant)
- **razlika od id-zavisnosti** je što možemo da pravimo **više potklasa** – bar 2, a ključ svake potklase je primarni ključ superklase – **nasleđivanje ključeva, ne unira se**
- **potklase** mogu imati svoje **sopstvene ključeve**
- potklasa može imati ulogu superklase u drugoj IS-A hijerarhiji
- nad 1 tipom može se napraviti više različitih IS-A hijerarhija, koristeći različite kriterijume

## KATEGORIZACIJA

TP kategorizacije uvodi pojam **kategorije** – predstavlja posebnu vrstu tipa (TE, ili TP – gerunda)

- **1 TE se povezuje s više kategorija** (bar 2)
- **ekskluzivni TP** – **TE koji se kategorije** može biti **samo 1 kategorije** pa je **max** kardinalitet prema kategorijama **1 ( , 1 )**
- **nema id-zavisnosti**, kategorije su **nezavisne** 1 od 2., mogu imati različite ključeve (kao 3 posebne tabele se mogu predstaviti)
- može, a ne mora postojati skup klasifikacionih obeležja kategorije

**IS-A ne može da se zameni kategorizacijom** – kategorizacija **ne modeluje 1 klasu TE i svaki TE je nezavisan TE**

**tip** kategorizacije prema kardinalitetu:

- **(0, 1)** – **parcijalna**
- **(1, 1)** – **totalna**

pr: član kluba mora biti ili pravno ili fizičko lice, a lica mogu ostvariti 0 ili više članstva kluba

## 5. RELACIONI MODEL

### MOTIVACIJA I KONCEPCIJA RELACIONOG MODELA

- nezavisnost programa od podataka
- **šema relacije: N(R, C)**
  - R – skup obeležja
  - C – skup ograničenja,  $K \subseteq C$  – neprazan skup ključeva
- **šema releacije (zaglavljene) LSO**, daje kontekst **relaciji (sadržaj) LSP**

## **STRUKTURALNA JEDNOSTAVNOST**

- relacija – zasnovana na matematici, skup torki

selekcija podataka:

- kod ranijih MP – putem **fizičkih** (relativnih ili absolutnih) adresa: pozicioniranje indikatorom aktuelnosti ili odnos između podataka
- kod rel. MP – **asocijativno adresiranje -> simbolička** adresa – podaci se nalaze na osnovu naziva relacije, obeležja i vr. ključa a SUBP transformiše simboličke u relativne adrese

povezivanje podataka:

- kod ranijih MP – fizičke adrese u funkciji **pokazivača** – fizičko pozicioniranje logički susednih podataka
- kod relacionog MP – simboličkih adresa – **prenetih vrednosti ključa**
  1. preko nove tabele (više više) – prostiranje ključa
  2. ista tabela (max kardinalitet je 1) – prostiranje ključa (strani ključ, ograničenja referencijalnog integriteta)
- **strani ključ**
  - propagirani ključ
  - skup vrednosti stranog ključa je podskup skupa vrednosti primarnog ključa
  - sme da bude null

## **DEKLARATIVNI JEZIK**

2 alata za upitni jezik:

1. **relaciona algebra** (teorija skupova i skupovnih operacija)
  - skupovni operatori: unija, presek, razlika
  - specijalizovani skupovni operatori: spoj (join), projekcija, selekcija, ....
2. **relacioni račun** (predikatski račun I reda)

**SQL** – Structured Query Language

- zasnovan na relacionom računu nad torkama
- deklarativan – ne kako, nego samo šta da radi
- rad sa skupovima podataka (torki)

## **STRUKTURALNA KOMPONENTA**

1. *primitivni* koncepti nivoa **intenzije**:

- **obeležje** – reprezentuje osobinu klase E ili P u RS
- **domen** – specifikacija skupa mogućih vrednosti koje neka obeležja mogu da dobiju (svakom obeležju obavezno se pridružuje tačno 1 domen)

primitivni koncept nivoa **ekstenzije**:

- **vrednost**

## 2. složeni koncepti - kombinovanje primitivnih korišćenjem definisanih pravila

Nivo intenzije (LSO)	Nivo ekstenzije (LSP)
Domen	Vrednost
Obeležje	Podatak
Skup obeležja	Torka
Sema relacije	Relacija
Sema BP	BP

### Torka

- reprezentuje 1 PE ili PP
- svakom **obeležju**, iz skupa obeležja, **dodeljuje se konkretna vrednost** (iz skupa mogućih vrednosti definisanog domenom)
- $U = \{A_1, \dots, A_n\}$   $U = \{\text{MBR, IME, POL, SPR, NAP}\}$
- $\text{DOM} = \cup (\text{dom}(A_i))$  - skup svih mogućih vrednosti
- $t : U \rightarrow \text{DOM}$ ,  $(\forall A_i \in U)(t(A_i) \in \text{dom}(A_i))$
- $t_1 = \{(\text{MBR, 101}), (\text{IME, Ana}), (\text{POL, Ž}), (\text{SPR, 1100}), (\text{NAP, Univerzitetski IS})\}$

### Restrikcija (“skraćenje”) torke t

- na skup obeležja  $X \subseteq U$
- oznaka:  $t[X]$
- svakom obeležju iz skupa X pridružuje se ona vrednost koju je **imala polazna torka t**
- $X \subseteq U$ ,  $t : U \rightarrow \text{DOM}$ ,  $t[X] : X \rightarrow \text{DOM}$ ,  $(\forall A \in X)(t[X](A) = t(A))$
- $X = \text{MBR+IME}$   $t_1[X] = \{(\text{MBR, 101}), (\text{IME, Ana})\}$

### Relacija

- konačan **skup torki**
- reprezentuje skup realnih entiteta ili poveznika

$r(U) \subseteq \{t \mid t : U \rightarrow \text{DOM}\}, |r| \in \mathbb{N}_0$  skup svih mogućih torki nad skupom obeležja U – Tuple(U)

- $r_1(U) = \{t_1, t_2\}$
- u relaciji se ne mogu pojaviti 2 identične torke (to je ista torka, samo 2 puta prikazana)
- relaciju predstavlja **tabela**
- poređak kolona i torki je nebitan

## Šema relacije: N(R, O)

- N - naziv
- R - skup obeležja
- O - skup ograničenja šeme

- **Pojava** nad šemom relacije (R, O) je bilo koja **relacija r(R)** koja **zadovoljava sva ograničenja iz skupa O**

## Relaciona šema baze podataka (S, I)

- **S - skup šema relacija**  $S = \{(R_i, O_i) \mid i \in \{1, \dots, n\}\}$
- **I - skup međurelacionih ograničenja**

## Relaciona BP

- **skup relacija**
- **1 pojava nad zadatom relacionom šemom** BP (S, I)
- s:  $S \rightarrow \{r_i \mid i \in \{1, \dots, n\}\}, (\forall i)s(R_i, O_i) = r_i$ 
  - svakoj šemi relacije iz skupa S odgovara 1 njena pojava
  - **svaka relacija mora da zadovolji O**
  - **skup relacija s** mora da zadovoljava sva **međurelaciona ograničenja iz skupa I**

## BP

- reprezentuje 1 stanje RS
- ažurira se, jer promene RS treba da prate promene podataka u BP
- Nivo intenzije :  $\{((R_1, O_1), \dots, (R_n, O_n)), I\}$  - šema BP: staticka (sporo promenljiva)
- Nivo ekstenzije:  $\{r_1(R_1), \dots, r_n(R_n)\}$  - relaciona BP: dinamička (stalno promenljiva)

## Konzistentno stanje BP

$RBP = \{r_i \mid i \in \{1, \dots, n\}\}$  nad **šemom (S, I)** nalazi se u

- **formalno konzistentnom stanju**
  - o  $(\forall r_i \in RBP)(r_i \text{ zadovoljava } \text{sva ograničenja} \text{ odgovarajuće šeme } (R_i, O_i))$
  - o **RBP** zadovoljava sva **međurelaciona ograničenja** iskazana putem I
- **suštinski konzistentnom stanju**
  - o formalno konzistentnom stanju
  - o predstavlja venu sliku stanja RS

**SUBP** može da kontroliše formalnu konzistentnost

## INTEGRITETNA KOMPONENTA

- definisana putem tipova ograničenja

- karakteristike tipa ograničenja:

- formalizam za zapisivanje (definicija) - MP
- pravilo za interpretaciju (validaciju) - pravila
- oblast definisanosti - LSO nad kojom se definiše ograničenje
- oblast interpretacije - LSP nad kojom se ograničenje interpretira

### Karakteristike tipa ograničenja

- **kritične operacije** - skup operacija nad BP koje mogu da **naruše ograničenja** datog tipa (INSERT, UPDATE, DELETE)
- skup **akcija** se definiše za kritične operacije kojima se obezbeđuje konzistentnost BP
- pasivna akcija (SUBP u pozadini stalno proverava) i aktivna akcija
- ograničenje treba biti uz podatke, tu gde se nalaze

**LSO** nad kojim se definiše ograničenje:

- **vanrelaciono** – van konteksta šeme relacije (ogr. Domena)
- **jednoretaciono** – nad 1 šemom relacije (ogr. Ključa)
- **višeretaciono** – nad bar 2 šeme (ogr. Referencijalnog integriteta)

**LSP** nad kojim se interpretira ograničenje:

- **vrednosti** – nad 1 vrednošću obeležja
- **torke** – nad 1 torkom
- **relaciono** – nad skupom torki
- **meduretaciono** – nad bar 2 relacije (bilo koja relacija, rezultat spoja, pogled, bilo šta)

### SPECIFIKACIJA DOMENA: **D(id(D), Predef)**

- D – naziv domena
- id(D) – ograničenje domena
- Predef – predefinisana vrednost domena, umesto null, nije obavezna

### OGRANIČENJA DOMENA **id(D)**

- **id(D) = (Tip, Duzina, Uslov)**
- Tip – tip podatka, obavezno (oznaka primitivnog domena ili prethodno definisanog domena)
- Uslov – logički uslov koji svaka vrednost domena mora da zadovolji
- pr. DOCENA((Number, 2,  $d \geq 5 \wedge d \leq 10$ ),  $\Delta$ )  $\rightarrow$  DPOZOCENA((DOCENA,  $\Delta$ ,  $d \geq 6$ ), 6)
- **null** – nula (**nedostajuća**) vrednost –  $\omega$ , vrednost obeležja nedostaje – nije zadata
  1. nepoznata ali postojeća vrednost obeležja
  2. nepostojeća vrednost obeležja
  3. neinformativna vrednost obeležja

LSO: **vanrelaciono**

LSP: nad **vrednošću**

## SPECIFIKACIJA OBELEŽJA ŠEME RELACIJE

- $A \in R$ ,  $N(R, O)$
- zadaje se za svako obeležje šeme relacije: (**id(N, A)**, **Predef**)
  - $\text{id}(N, A)$  - ograničenje vrednosti obeležja
  - Predef - predefinisana vrednost obeležja
- radi se u svakoj šemi relacije - npr. primarni ključ ne sme null, a strani sme null

## OGRANIČENJE VREDNOSTI OBELEŽJA

**id(N, A) = (Domen, Null)** – oba su obavezna

- Domen – naziv domena pridruženom obeležju
- Null – da li obeležje sme ili ne sme da ima null vrednost
- LSO: **jednorelacioni**
- LSP: **vrednosti**

## OGRANIČENJE TORKE

- za šemu relacije  $N(R, O)$       **id(N) = id(R) = (\{id(N, A) | A \in R\}, Uslov)**
  - **Uslov** – svaka torka mora da zadovolji, tu su obeležja samo iz te šeme relacije
  - LSO: **jednorelacioni**
  - LSP: nad **torkom**
  - predstavlja **skup ograničenja vrednosti obeležja, kojem je pridodat logički uslov**

## KLJUČ ŠEME RELACIJE

- **minimalni podskup skupa obeležja šeme relacije**, na osnovu kojeg se **jedinstveno može identifikovati** svaka **torka relacije nad datom šemom**
- X je ključ ako:

1.  $(\forall u, v \in r(R))(u[X] = v[X] \Rightarrow u = v)$
2.  $(\forall Y \subset X)(\neg 1.)$

- LSO: **jednorelaciono**
- LSP: nad **realcijom**

## OGRANIČENJE KLJUČA

- šema relacije  $N(R, K)$ ; ključ  $X \in K$ ,  $X \subseteq R$
- $\text{Key}(N, X)$

za sva obeležja ključa nula vrednosti su zabranjene:  $(\forall K_i \in K)(\forall A \in K_i)(\text{Null}(N, A) = \perp)$

- svaka šema relacije mora posedovati najmanje jedan ključ ( $K \neq \emptyset$ ) - tačno 1 primarni ključ koji se bira iz skupa ekvivalentnih ključeva i koristi se kao asocijativna adresa za povezivanje podataka
- kritične operacije: INSERT, UPDATE – SUBP redom ide i proverava vrednosti ključa

## OGRANIČENJE JEDINSTVENOSTI

- vrednosti obeležja šeme relacije  $N(R, O)$

### Unique(N, X)

- $X$  – nad skupom obeležja (znači više obeležja ne mora samo 1),  $X \subseteq R$
- može imati null, a ako je ne-null onda kombinacija vrednosti obeležja iz  $X$  mora biti jedinstvena u relaciji nad  $N(R, O)$ ,

$$(\forall u, v \in r(R))((\forall A \in X)(u[A] \neq \omega \wedge v[A] \neq \omega) \Rightarrow (u[X] = v[X] \Rightarrow u = v))$$

- skup svih ograničenja jedinstvenosti:  $Uniq = \{\text{Unique}(N, X) \mid X \subseteq R\}$
- nad 1 šemom relacije može više unique, onliko koliko treba
- LSO: **jednorelaciono**
- LSP: **relaciono** (skup torki)
- kritične operacije: INSERT, UPDATE
- aktivnosti: da se zabrani

Skup svih ograničenja šeme relacije:  $N(R, K \cup Uniq \cup \{id(R)\})$  – ogr. ključeva, jedinstvenosti i torke

## ZAVISNOST SADRŽAVANJA

- **2 šeme** relacije  $N_i(R_i, O_i)$  i  $N_j(R_j, O_j)$

**uredimo** kao nizove, zna se pozicija:

- $X \subseteq R_i$ :  $X = (A_1, \dots, A_n)$ ,  $(\forall l \in \{1, \dots, n\})(A_l \in R_i)$
- $Y \subseteq R_j$ :  $Y = (B_1, \dots, B_n)$ ,  $(\forall l \in \{1, \dots, n\})(B_l \in R_j)$

$X, Y$  – isti br. elemenata, 1 ili više obeležja imaju

- $(\forall l \in \{1, \dots, n\})(\text{dom}(A_l) \subseteq \text{dom}(B_l))$  – obeležja moraju biti **domenski kompatibilni**

$$N_i[X] \subseteq N_j[Y]$$

ako za  $r(R_i, O_i)$  i  $s(R_j, O_j)$  zadovoljeno  $(\forall u \in r)(\exists v \in s)(\forall l \in \{1, \dots, n\})(u[A_l] = \omega \vee u[A_l] = v[B_l])$

- ili je **vrednost obeležja null** – zadovoljava
- ili za **vrednost obeležja A u torki u mora da postoji vrednost obeležja B u torki v**
- **skup vrednosti za A je  $\subseteq$  skupa vrednosti za B**

- LSO: **višerelaciono**

- LSP: **nad 2 relacije nad šemama**

## OGRANIČENJE REFERENCIJALNOG INTEGRITETA

- specifičan slučaj zavisnosti sadržavanja

- $N_i[X] \subseteq N_j[Y]$ , kada je  $Y$  **KLJUČ šeme relacije**  $N_j(R_j, K_j)$
- $N_i$  - **referencirajuća** šema relacije
- $N_j$  - **referencirana** šema relacije

kritične operacije (levo=RADPROJ, desno=RADNIK):

- INSERT levo (stranog ključa ako ne postoji primarni ključ desno)
- UPDATE levo i desno
- DELETE desno (brisanje primarnog ključa koga referencira strani ključ možda)

## FUNKCIONALNA ZAVISNOST (FZ)

$f: X \rightarrow Y$ ,  $X$  i  $Y$  skupovi obeležja

- $X$  i  $Y$  podskupovi  $U$  – univerzalni skup **obeležja** (sva obeležja u RS)
- **ako je poznata  $X$  vrednost, poznata je i  $Y$  vrednost**, obrnuto ne važi
- svakoj  $X$  vrednosti odgovara samo 1  $Y$  vrednost
- **$X$  funkcionalno određuje  $Y$**

relacija  $r$  zadovoljava FZ  $X \rightarrow Y$  ako važi  $(\forall u, v \in r)(u[X] = v[X] \Rightarrow u[Y] = v[Y])$

LSP: relacija  $r(N)$  ili  $r(U)$

### Trivijalna FZ

- ona koja je zadovoljena u bilo kojoj relaciji
- FZ  $X \rightarrow Y$ , za koju važi  $Y \subseteq X$
- npr. MBR  $\rightarrow$  MBR, MBR  $\rightarrow \emptyset$ , AB  $\rightarrow$  A

## LOGIČKA POSLEDICA

$F \models f$  FZ  $f$  je logička posledica od skupa FZ  $F$

- ako svaka relacija  $r$  koja zadovoljava  $F$  zadovoljava i  $f$  –  $(\forall r \in \text{SAT}(U))(r \models F \Rightarrow r \models f)$
- $F_1 \models F_2$  – skup FZ  $F_2$  je logička posledica od skupa FZ  $F_1$  ako  $(\forall f \in F_2)(F_1 \models f)$
- implikacioni problem: utvrditi da li važi  $F \models f$
- **ekvivalentnost** skupova FZ:  $F_1 \equiv F_2$  ako  $F_1 \models F_2 \wedge F_2 \models F_1$

## ZATVARAČ SKUPA FZ

$F^+$  – skup koji sadrži sve logičke posledice od  $F$

- $F^+ = \{f \mid F \models f\}$
- važi za svaki  $F$  da  $F \subseteq F^+$
- $F_1 \models F_2$  akko  $F_2^+ \subseteq F_1^+$
- ekvivalentnost skupova:  $F_1 \equiv F_2$  akko  $F_1^+ = F_2^+$

## ARMSTRONGOVA PRAVILA IZVOĐENJA

- refleksivnost:  $Y \subseteq X \mid - X \rightarrow Y$
- proširenje:  $X \rightarrow Y, W \subseteq V \mid - X \rightarrow YW$
- pseudotranzitivnost:  $X \rightarrow Y, YV \rightarrow Z \mid - X \rightarrow VZ$
- 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:  $X \rightarrow Y, Y \rightarrow Z \mid - X \rightarrow Z$

## Nepotpuna FZ

- $X \rightarrow Y \in F$  je nepotpuna ako sadrži logički suvišno obeležje na levoj strani:  $(\exists X' \subset X)(X' \rightarrow \in F^+)$
- redukujemo

## Tranzitivna FZ

- $X \rightarrow Z$  tranzitivna ako važi  $X \rightarrow Y \in F^+$  i  $Y \rightarrow Z \in F^+$ , a ne važi da je  $Y \rightarrow X \in F^+$
- suvišno – izbacimo

## Ključ šeme relacije i FZ

$X$  je ključ šeme relacije  $(R, F)$ , ako važi

1. iz  $F$  sledi  $X \rightarrow R$  ( $X \rightarrow R \in F^+$ ) –  **$X$  funkcionalno određuje čitav skup obeležja**
2.  $X$  je **minimalni skup** obeležja s osobinom 1.  $\neg(\exists X' \subset X)(X' \rightarrow R \in F^+)$

## ZATVARAČ skupa obeležja – skup svih obeležja koja funkcionalno zavise od $X$

$XF^+ = \{A \in U \mid X \rightarrow A \in F^+\}$ ,  $A$  funkcionalno zavisi od  $X$

## IZRAČUNAVANJE ZATVARAČA

$U = \{A, B, C, D, E, F\}$ ,  $F = \{AB \rightarrow AC, CD \rightarrow E, A \rightarrow B, AE \rightarrow F\}$

1.  $(AD)^+ = AD$  (1. korak samo ta obeležja)
2.  $(AD)^+ = ADB$  ( $A \rightarrow B$ ) – iz  $F$  uzimamo FZ gde je sa leve strane  $A, D$ , ili  $AD$  i uzimamo ono što je desno
3.  $(AD)^+ = ADBC$  ( $AB \rightarrow AC$ )
4.  $(AD)^+ = ADBCE$  ( $CD \rightarrow E$ )
5.  $(AD)^+ = ADBCEF$  ( $AE \rightarrow F$ )
6.  $(AD)^+ = ADBCEF$  (nema više) –  $AD$  funkc. određuje sva obeležja pa je kandidat za ključ

1.  $(BD)^+ = BD$
2.  $(BD)^+ = BDEG$  ( $D \rightarrow EG$ )
3.  $(BD)^+ = BDEGC$  ( $BE \rightarrow C$ )

4.  $(BD)^+ = BDEGCA$       ( $C \rightarrow A$ )
5.  $(BD)^+ = BDEGCA$

## ALGORITAM ZA TRAŽENJE 1 KLJUČA

$U = \{A, B, C, D, E\}$ ,  $F = \{AB \rightarrow CDE, E \rightarrow A, CD \rightarrow B\}$

1.  $(ABCDE)^+ = R$                             /E - krećemo od čitavog skupa obeležja
2.  $(ABCD)^+ = ABCDE = R$       ( $AB \rightarrow CDE$ )      /D - odbacujemo obeležja
3.  $(ABC)^+ = ABCDE = R$       ( $AB \rightarrow CDE$ )      /C
4.  $(AB)^+ = ABCDE = R$       ( $AB \rightarrow CDE$ )      /B - kraj kada više ne možemo da redukujemo
5.  $(A)^+ = A \neq R$
6.  $(B)^+ = B \neq R$

## ALTERNATIVNI KLJUČEVNI

- ako je sa desne strane A, B, ili AB menjamo sa onim sa leve

- $K = \{AB, EB, ACD, ECD\}$
- $E \rightarrow A$
- $CD \rightarrow B$

$U = \{A, B, C, D, E, F, G, H\}$ ,  $F = \{AB \rightarrow CE, C \rightarrow B, ED \rightarrow F, F \rightarrow G\}$

1.  $(ABCDEFGH)^+ = R$       ( $F \rightarrow G$ )      /G
2.  $(ABCDEFH)^+ = R$                             ( $ED \rightarrow F$ )      /F
3.  $(ABCDEH)^+ = R$                             ( $C \rightarrow B$ )      /B
4.  $(ACDEH)^+ = R$                             /E
5.  $(ACDH)^+ = R$                             /C
6.  $(ADH)^+ \neq R$                             /A
7.  $(CDH)^+ \neq R$                             /D
8.  $(ACH)^+ \neq R$

- $K = \{ACDH, ABDH\}$
- $AB \rightarrow CE$
- $AB \rightarrow C$

## OPERACIJSKA KOMPONENTA

- Jezik za manipulaciju podacima – (DML) ažuriranje relacija: INSERT, UPDATE, DELETE
- Jezik za definiciju podataka – (DDL) operacije za upravljanje šemom BP: CREATE, DELETE, ALTER
- Upitni jezik u RMP – operacije za izražavanje upita: SELECT
- Upitni jezik sačinjavaju:
  - operatori za izražavanje upita

- pravila za formiranje operanada upita – izraza
- pravila za primenu tih operatora
- Osnovne skupovne operacije nad relacijama: Unija, Presek, Razlika

- Vrste teoretskih upitnih jezika:

- relaciona algebra (teorija skupova)
- relacioni račun (predikatski račun I reda)

### **SELEKCIJA** - torki iz relacije

$$\sigma_F(r(R)) = \{t \in r \mid F(t)\} \quad F - \text{logička formula (u Sql-u WHERE)}$$

$$\sigma_F(\text{radnik}), F := \text{PLT}>5000$$

### **PROJEKCIJA (RESTRIKCIJA) RELACIJE** - projektovanje relacije na podskup skupa obeležja

$$X \subseteq R \quad \pi_X(r(R)) = \{t[X] \mid t \in r(R)\} \quad X - \text{na taj skup se projektuje}$$

$$\pi_{\text{MBR+IME}}(\sigma_F(\text{radnik})), F := \text{sef} = 10$$

### **PRIRODNI SPOJ RELACIJA** - spajanje torki različitih relacija na osnovu istih vrednosti zajedničkih obeležja $r(R)$ i $s(S)$ , $r(R) \bowtie s(S) = \{t \in \text{Tuple}(RS) \mid t[R] \in r \wedge t[S] \in s\}$

- t – skup torki definisan nad unijom obeležja
- $r(R), s(S)$  – ako nemaju zajednička obeležja, onda restrikcija
- ( $\bowtie$  = mašnica simbol)

$$\pi_{\text{NAP}}(\sigma_F(\text{radnik} \bowtie \text{radproj} \bowtie \text{projekat})), F := \text{plt}>2500$$

### **DEKARTOV PROIZVOD RELACIJA** - spajanje svakih sa svakim i nemaju zajedniča obeležja

$$R \cap S = \emptyset \quad r(R) \times s(S) = \{t \in \text{Tuple}(RS) \mid t[R] \in r \wedge t[S] \in s\}$$

- ❖ kada radimo INNER JOIN – br. torki će biti koliko ih ima u tabeli radnik
- ❖ kada radimo OUTER JOIN – br. torki će biti koliko ima u radproj + sve torke iz radnika koji nisu u radproj

## 6. METODE PRISTUPA I ORGANIZACIJA DATOTEKA

### OSNOVNA STRUKTURA DATOTEKE

Datoteka = struktura slogova – definisana nad **1 tipom sloga**

- LSO - daje šemu za slog
- LSP – vrednost obeležja sloga
- FSP – uključuje podatke iz LSP i podatke o organizaciji FSP na eksternom memorijskom uređaju, svaki slog predstavlja niz polja sa vrednostima atributa

**format sloga** – pravila za strukturiranje i interpretaciju sadržaja sloga ([kako da čitamo sadržaj sloga](#))

Opšta struktura sloga datoteke kao FSP:

$k_i(S)$	$p_i(S)$	$s(S)$	$u_i(S)$	$f_i(S)$
----------	----------	--------	----------	----------

- $k_i(S)$  – polja vrednosti atributa [primarnog ključa](#), obavezno polje
- $p_i(S)$  – ostalih atributa
- $s(S)$  – 1 polje [statusa sloga](#) – indikator aktuelnosti sloga u LSP
- $u_i(S)$  – pokazivači na logički narednog
- $f_i(S)$  – [kontrolna polja](#) kod slogova varijabilne dužine

Vrste polja u sloganima:

1. polja konstantne dužine – ne memoriše se granica polja
2. polja promenljive dužine:
  - memoriše se granica polja
  - koristi se [kontrolno polje](#)  $f_i(S)$
  - **tehnike:**
    - navodi se [aktuelna dužina](#) polja [ispred](#) sadržaja polja
    - [specijalna oznaka kraja](#) polja [nakon](#) sadržaja polja

Vrste sloganova prema dužini:

1. sloganovi konstantne dužine – sva polja su konstantne dužine
2. sloganovi promenljive dužine – postoji bar 1 polje promenljive dužine u slogu
  - **tehnike:**
    - navođenjem aktuelne dužine sloga u [kontrolnom polju](#), ispred sadržaja sloga
    - navođenjem specijalne oznake kraja sloga u [kontrolnom polju](#), nakon sadržaja sloga

- o uvođenjem posebne indeksne strukture sa rednim brojevima bajtova koji ukazuju na početke slogova

Vrste slogova prema ponavljanju vrednosti:

1. slogovi s ponavljajućim grupama – višestruko pojavljivanje vrednosti atributa u 1 slogu
2. slogovi bez ponavljajućih grupa

Vrste adresa lokacija:

1. **apsolutna (mašinska)** – strukturirana prema adresnom prostoru jedinice diska
2. **relativna** – predstavlja redni broj lokacije (rbr. bajta u bloku), OS transformiše relativnu u maš.
3. **simbolička (asocijativna)** – vrednost ključa, MP transformiše simboličku u relativnu

## **STRUKTURA DATOTEKE KAO NIZ BLOKOVA**

Blok (logički blok)

- organizaciona jedinica podataka
- predstavlja niz slogova
- Odnos blok – fizički blok: uglavnom 1 blok predstavlja niz od  $2^n$  ( $n \geq 0$ ) fizičkih blokova

### **FORMAT BLOKA**

<b>ZAGLAVLJE BLOKA</b>	<b>A<sub>i</sub><sup>1</sup></b>	...	<b>A<sub>i</sub><sup>j</sup></b>	...	<b>A<sub>i</sub><sup>f</sup></b>
	<b>s<sub>1</sub></b>	...	<b>s<sub>j</sub></b>	...	<b>s<sub>f</sub></b>

- $A_i$  - adresa bloka (najčešće relativna)
- $A_{i,j}$  - relativna adresa j-tog sloga u i-tom bloku ( $i, j$ )
- **f - faktor blokiranja** – max broj slogova koji može da stane u blok
- zaglavlje bloka – neobavezno, dodatne podatke npr. različita polja pokazivača, br. slogova u bloku, indeks na početke slogova

Vrste blokova:

1. blokovi sa slogovima promenljive dužine – više slogova može biti smešteno u 1 blok, dozvoljeno je i da veličina 1 sloga premaši kapacitet bloka pa se radi ulančavanje blokova 1 sloga
2. blokovi sa slogovima konstantne dužine – svaki blok datoteke sadrži uvek isti broj slogova

Struktura datoteke kao niza blokova:

1. linearna struktura blokova datoteke - svaki blok datoteke obuhvata niz slogova datoteke
2. strogo strukturirana datoteka - svi su istog tipa

zaglavlje datoteke - slog na početku datoteke, tu stavljamo ono što mislimo da nam treba (br. slogova, dužina i format sloga, pozicija polja ključa u slogu)

Oznaka kraja datoteke:

1. uvođenje specijalnog sloga
2. uvođenje specijalne oznake kraja u polje pokazivača
3. vođenjem posebne evidencije zauzetosti prostora
4. kraj datoteke je kraj prostora dodeljenog datoteci - nema označavanje

## **METODA PRISTUPA (MP)**

Paket programa (rutina) za podršku usluga visokog nivoa

Usluga razmene podataka sa aplikativnim programom:

- upravljanje strogo strukturiranim datotekama (organizacija, memorisanje polja, slogova i blokova)
- upravljanje baferima MP
- podrška različitim vrstama organizacije datoteka - podrška različitim načinima memorisanja logičkih veza i adresiranja (pristupa podacima)
- vođenje brige o kategorijama - zaglavlje, početak i kraj datoteke, tekući pokazivač, indikator aktuelnosti
- podrška izgradnji specijalnih pomoćnih struktura za poboljšanje efikasnosti obrade podataka
- podrška opštih postupaka upravljanja sadržajem datoteka - kreiranje, traženje, pretraživanje, ažuriranje i reorganizacija
- koristi ili uključuje usluge niskog nivoa izabranog OS
- obezbeđuje nezavisnost aplikativnog programa od usluga niskog nivoa OS

Upravljanje strogo strukturiranim datotekama:

- podrška organizacije slogova i polja
- podrška različitih tipova podataka
- podrška različitih kodnih rasporeda
- konverzije podataka

Usluge razmene podataka sa aplikativnim programom na nivou:

1. **sloga**: grupisanje slogova u blokove pri upisu podataka, rastavljanje bloka na slogove pri čitanju podataka, održavanje tekućeg pokazivača kao relativne adrese sloga, redni broj sloga u datoteci i njegova transformacija
2. **bloka**: razmena sadržaja kompletnih logičkih blokova između aplikativnog programa i datoteke, održavanje tekućeg pokazivača kao relativne adrese bloka

Usluge pristupa podacima (slogovima ili blokovima) iz aplikativnih programa:

1. **sekvencijalni** pristup - automatski održavaju vrednost tekućeg pokazivača
2. **direktni** pristup - eksplicitno zadavanje vrednosti tekućeg pokazivača
3. **dinamički** (kombinovani)

3 nivoa "baferisanja" podataka datoteke u OM:

1. nivo **sistemskih bafera** - njima upravlja OS
2. nivo **bafera MP** - njima upravlja okruženje u kojem je implementirana MP
3. nivo **lokacija promenljivih u aplikativnom programu** - njima upravlja aplikativni program

Okruženja koja uključuju metode pristupa:

- **OS** - podržava upravljanje blokovima i baferima MP
- programski jezik sa pridruženim paketima (**bibliotekama**) funkcija – usluge nivoa sloga datoteke
- **SUBP** - upravljanje blokovima i baferima MP

## **PARAMETRI ORGANIZACIJE DATOTEKE**

Organizacija podataka – projektovanje LSO i FSP kako bi obezbedile korisničke zahteve i efikasnu obradu podataka = rezultat su sistem BP, sistem datoteka

Kako dodeliti lokaciju slogovima i memorisati logičke veze, koje strukture koristiti, smeštanje u memoriju...

Organizacija datoteke – projektovanje LSO se svodi na projektovanje tipa entiteta N(Q, C), tj. tipa sloga

## **IZBOR VRSTE ORGANIZACIJE DATOTEKE ZAVISI OD:**

1. **načina dodelje lokacija novim slogovima:**
  - o piše se na kraj datoteke, na poslednju lokaciju
  - o spregnuta lista
  - o transformacija ključa u adresu – hash ili tabelarno
2. **načina memorisanja logičkih veza između slogova u LSP**
  - o logički susedan = fizički susedan

- o pokazivači na logički susedne – tabelu sa vrednošću ključa i adresom logički susednog ili svaki slog ima dodat pokazivač
- o nema memorisanja o logički susednim

## VRSTE ORGANIZACIJE DATOTEKE

1. **osnovne** organizacije – FSP ima **1 memorisku zonu** (serijska, sekvencijalna, spregnuta, rasuta)
2. **složene** organizacije – kombinovanjem osnovnih organizacija, FSP ima **bar 2 memoriske zone** (rasute sa zonom prekoračenja, indeks-sekvencijalne, B-stabla)

SEKVENCIJALNA	PISE SE NA KRAJ	LOG SUSEDNI = FIZ SUSEDNI
SERIJSKA	PISE SE NA KRAJ	-
SPREGNUTA	SPREGNUTA LISTA	POKAZIVACI
RASUTA	TRANSFORMACIJA KLJUCA	-

## OPŠTE PROCEDURE NAD DATOTEKAMA

### FORMIRANJE DATOTEKE - postupak kreiranja FSP datoteke

2 vrste datoteka:

1. formiraju se u **posebnom** postupku – npr. sortiranje
2. formiraju se u **redovnom** postupku ažuriranja (upisa novih slogova)
  - ad hoc – sa unosom novih slogova
  - na osnovu podataka koji već postoje u drugoj datoteci

### PRISTUPANJE DATOTECI – postupak pozicioniranja na željenu lokaciju sloga/bloka datoteke

- o **sekvencijalni** pristup – automatsko održavanje rel. adrese **tekućeg pokazivača**, logički sledeći
- o **direktni** pristup – zadajemo **direktnu adresu** i skačemo na nju
- o **dinamički** – direktno zadamo adresu bloka i sekvencijalno čitamo slogove u bloku

\***sekvencijalna organizacija** – zavisi od 2 parametra (uzima se poslednja lokacija i log sus. = fiz sus.), a **sekvencijalan pristup** – pristupanje podacima na osnovu tekućeg pokazivača, automatski se ažurira

### TRAŽENJE U DATOTECI

- $\text{dom}(K) \rightarrow \text{Ind} \times A \times S$ 
  - argument traženja – vr. ključa iz domena,  $a \in \text{dom}(K)$
  - $\text{Ind}$  = uspešnost traženja, true = našao (delete, update), false = nije našao (insert)

- A = realtivna adresa mesta zaustavljanja traženja
- S = sadržaj sloga ako nađe ili ako ne nađe specijalna vrednost

prema vrsti postupka traženja:

- o linearo – redom
- o binarno – soritrano,<,>
- o pokazivačem
- o hash funkcijom

vrste traženja s obzirom na predistoriju traženja:

- traženje **slučajno odabranog sloga**
- traženje **logički narednog sloga** – početna adresa traženja je adresu na kojoj je **zaustavljeno** prethodno traženje, mora se znati podatak o **logički narednom slogu**

## PRETRAŽIVANJE U DATOTECI

- dom(**LogUslov**) → P(S) ili P(A)

- argument pretraživanja je logički uslov
- P(S) – **skup slogova** koji zadovoljavaju uslov, uspešno je ako nije prazan skup
- P(A) – skup adresa koje zadovoljavaju uslov
- sekundarni ključ – niz obeležja strukture po kojem se vrši pretraživanje

## AŽURIRANJE DATOTEKE

postupak dovođenja LSP datoteke u sklad sa izmenjenim stanjem klase entiteta u RS

1. **upis novog** sloga u datoteku – 1 neuspešno traženje, može iziskivati premeštanje slogova
2. **modifikacija** vrednosti **atributa** sloga – 1 uspešno traženje, sem atributa koji čine primarni ključ
3. **brisanje** postojećeg sloga iz datoteke – 1 uspešno traženje, može iziskivati premeštanje slogova
  - o **logičko brisanje - polja statusa sloga**: aktuelan ->neaktuelan slog, ali i dalje zauzima memoriju, posle se vrši reorganizacija pa se oslobođa memorija
  - o **fizičko brisanje** – izmena sadržaja bloka u kojem se nalazio izbrisani slog, oslobođa se memorija odmah, može izazvati pomeranje drugih slogova iz 1 u 2. lokacije

## OBRADA DATOTEKE - svrshodna transformacija

Uloge datoteka u obradi:

- podela prema vrstama primenjenih operacija u obradi:

- **ulazna** – čitanje
- **izlazna** – pisanje
- **ulazno-izlazna** – oba

- podela prema *ulozi u traženjima slogova*:

- **vodeća** – generiše argumente traženja ili pretraživanja slogova tokom obrade, bar 1 ulazna datoteka u obradi mora biti vodeća
- **obrađivana** – u njoj se vrše traženja/pretraživanja, na osnovu generisanih argumenata
- **vodeća i obrađivana** – vodeća za neku drugu obrađivanu; obrađivana u odnosu na neku vodeću

- vrste obrade, prema načinima traženja slogova u obrađivanoj datoteci:

- **direktna** obrada – u svakom sledećem koraku obrade vrši se **traženje slučajno odabranog** sloga
- **redosledna** (sekvencijalna) obrada – u svakom narednom koraku obrade zahteva se **traženje logički narednog** sloga i/ili sekvensijalni pristup fizički susednoj lokaciji

## REORGANIZACIJA DATOTEKE

- ponovno formiranje datoteke u cilju dovođenja u **sklad FSP sa novim stanjem LSP** jer operacije ažuriranja vrše izmene u LSP koje FSP nekada ne prati na odgovarajući način i to dovodi do nagomilavanja neaktuelnih, logički izbrisanih slogova koji zauzimaju lokacije u FSP

## 7. SERIJSKA I SEKVENCIJALNA ORGANIZACIJA DATOTEKA

### SERIJSKA

- beleži slogove, slog iza sloga 1 za 2., novi slog se **piše na kraj, hronološki**
- **fizička struktura nema memorisanja veza o logičkim vezama** (ne zna se ko je logički susedan)
- **nema veze između vr. ključa i adrese lokacije**

Ako je datoteka velika i smeštena je na ekstreni memorijski uređaj, a mi treba podatke da smestimo u RAM, problem je cena čitanja i prenosa tih podataka u RAM. Onda moramo podatke **deo po deo** da prenosimo tako što pristupamo 1 slogu i onda zgrabimo i ostale ali imamo tehnološka ograničenja kao što je faktor blokiranja f

- blokiranje – blok od f slogova (f = faktor blokiranja koji može biti različit)
- $B = \frac{N+1}{f}$  blokova ima = ( br. slogova + \*(kraj datoteke)) / f
- logički blok – gradivna jedinica od  $2^n$  fizičkih blokova, a kraj bloka je npr. \* (specijalni znak)
- fizički blok – zavisi od OS, a kraj bloka je sam kraj fizičke strukture podataka
- $A_i$  – relativne adrese u odnosu na početak datoteke

### **FORMIRANJE** – nema poseban postupak

- seriska datoteka se generiše najčešće u postupku obuhvata podataka
- **obuhvat podataka** – inicijalno punjenje datoteke kada dolaze podaci sa papira, dokumenata ili sa uređaja koji prikuplja i beleži podatke pa se svaki novi slog piše na kraj datoteke, jedan za drugim
- podatke upisuje čovek uz pomoć format programa ili postoji softver koji ih piše
- **format program** – korisnički interfejs koji je prilagođen u zavisnosti od namene:
  - raspored polja
  - pomeranje kursora – tab, enter
  - dozvoljene operacije – ručni unos, modifikacija, brisanje
  - opis i formatiranje polja – tip, izgled
  - specijalne kontrole sadržaja polja – ograničenja (npr. obavezno polje)
- obuhvat podataka – vreme obavljanja:
  - u realnom vremenu
  - u odloženom režimu – već postojeći podaci
  - verifikacija – postupak suštinske provere ispravnosti unetih podataka

### **TRAŽENJE SLOGA**

- nema razlike između logički narednog i slučajno odabranog
- počinjemo od početka datoteke i idemo redom
- uspešno traženje, ukupan broj pristupa  $R_u : 1 \leq R_u \leq B \rightarrow$  **najbolji slučaj u 1. bloku, najgori poslednji blok**
- neuspešno traženje, ukupan broj pristupa  $R_n : R_n = B \rightarrow$  **moramo do poslednjeg bloka** doći
- očekivani (srednji) broj pristupa bloku za uspešno traženje (verovatnoća):
  - $p_i = \frac{f}{N}, i = 1, \dots, B-1 \rightarrow$  u blokovima do poslednjeg

- $p_B = \frac{N - f(B - 1)}{N}$  -> u poslednjem bloku
- $R_u = \sum_{i=1}^B i * p_i = \sum_{i=1}^{B-1} i \cdot \frac{f}{N} + B \cdot \frac{N - f(B - 1)}{N} = \frac{(B-1)(B-1+1)}{2} \cdot \frac{f}{N} + B \cdot \frac{N - f(B - 1)}{N} = \dots = \frac{B}{N} \left( N - \frac{f(B-1)}{2} \right)$

- uspešno traženje, ukupan broj upoređivanja arg. traženja i vr. ključa  $U_u : 1 \leq U_u \leq N$
- očekivani (srednji) broj:  $U_u = \sum_1^N i * p_i = \frac{N(N+1)}{2} \cdot \frac{1}{N} = \frac{N+1}{2}$
- neuspešno traženje:  $U_n = N$

## OBRADA

- direktna obrada – kao vodeća često ima redosled biranja (order by kako bi se postigla logička narednost)
- redosledna obrada – učitava slogove vodeće datoteke koji generišu argumente za traženje u serijskoj obrađivanju

vodeća:  $N_v = N_v^u + N_v^n$  broj slogova koji iniciraju uspešno i neuspešno traženje

ukupan prosečan broj traženja:  $R_{uk} = N_v^u R_u + N_v^n R_n$   $R_u, R_n$  – srednji br. pristupa uspešnih/neuspešnih

## AŽURIRANJE

- **upis novog sloga** – u 1. slobodnu lokaciju na kraju datoteke, a prethodi mu 1 neuspešno traženje

- $R_i = R_n + 1$ , ako f ne deli N+1 -> nije popunjeno poslednji blok, samo 1 upis za kraj datoteke (\*)
- $R_i = R_n + 2$ , ako f deli N+1 -> popunjeni su svi blokovi pa nam trebaju 2 pristupa: 1 za upis novog i 1 za kraj datoteke u novi blok (\*)

- **brisanje sloga** – prethodi mu 1 uspešno traženje, najčešće logičko – izmenom statusa aktualnosti sloga, a fizičko brisanje bi zahtevalo veliki broj pristupa

- **modifikacija** sloga – prethodi mu 1 uspešno traženje

- očekivani broj pristupa brisanje/modifikaciju:  $R_d = R_u + 1$  (+1 za izmenu ili promenu statusa sloga)

## OBLAST PRIMENE I OCENA KARAKTERISTIKE

- koriste se za **prekoračioce** – za slogove koji ne mogu da stanu
- **SUBP** najčešće piše u njih (**relacione BP**)
- **pogodne kao male** datoteke – kada mogu stati cele u OM, zbog **velikog broja pristupa**
- u kombinaciji sa indeksnim strukturama – veoma pogodna za direktnu obradu

## SEKVENCIJALNA

- slogovi su smešteni sukcesivno **1 za 2.**
- **logički susedni** slogovi smeštaju se u **fizički susedne** lokacije
- **nema veze** između **vr. ključa i adrese lokacije**
- relacija **poretka** – rastuće uređenje po vrednostima ključa  $\Rightarrow$  slog sa min vrednošću ključa smešta se u 1. lokaciju
- blokiranje – blok ima f sloganova, f je poželjno da bude što veće
- imamo sekvencijalni način pristupa (C jezik) – treba da napravimo našu **MP** – da ne **pristupamo preko bajtova, nego preko blokova i sloganova**

**FORMIRANJE** – hronološka serijska datoteka se **sortira** rastućim/opadajućim vrednostima ključ –> sekv.

### **TRAŽENJE SLOGA**

1) slučajno odabranog sloga

- linearno/binarno
- ima smisla kada je čitava učitana u operativnu memoriju jer je laka za čitanje kada je sve već poređano, ali ako je velika pitanje je isplativosti zbog čitanja bloka po bloku i još sortiranja

2) logički narednog sloga

linearno – koristeći **tekući pokazivač** krećemo od 1. lokacije i učitavamo fizički susedne blokove u OM i poredimo vr. ključa sa arg. traženja i alg. se zaustavlja ako:

- neuspšeno – nema sloga (**vr. ključa sloga >arg. traženja**) ili je došao do kraja dat. (vr. arg. traženja  $> \text{max vr. ključa sloga}$ )
- uspšeno – ima sloga i našao ga je

– br. pristupa kod uspšnog/neuspšnog traženja:

- $0 \leq R \leq B - i$
- **0** –> kada je blok već u OM, znači da je **prethodno obrađivan taj blok** pa nema pristupa datoteci
- **B - i** –> oduzima se redni broj tekućeg bloka, jer se **traženje nastavlja tamo gde smo se prethodno zaustavili**

– br. poređenja arg. traženja i vr. ključeva sloganova kod uspšnog/neuspšnog traženja:

- $1 \leq U \leq N - i + 1$ , mora makar 1, i gde smo završili prethodno

**OBRADA** – često se koristi kao **vodeća** datoteka u direktnoj i redoslednoj obradi

- 1) direktna – ima smisla ako je sekvencijalna datoteka **mala**, tako da se može smestiti u OM

- $R_{uk} = N_v^u R_u + N_v^n R_n$ ,  $R = \frac{B}{2}$  srednji broj pristupa za svaki slog
- 2) **redosledna**
- **odlična za traženje logički narednog**
  - **vodeća** sekvencijalna generiše logički naredne vrednosti ključa za traženje u obradivanoj, serijskoj datoteci

## AŽURIRANJE

- 1) upis novog sloga
  - 1 **neuspešno** traženje
  - treba da se upiše u lokaciju onog ko mu je **logički naredni**
  - a ostali se **pomeraju udesno** koji imaju vr. ključa > od vr. ključa novog
- 2) brisanje postojećeg sloga
  - 1 **uspešno** traženje
  - **pomeranje za 1 lokaciju** uлево svih slogova sa > vr. ključa obrisanog
- 3) modifikacija sadržaja sloga
  - **uspešno** traženje

**LOŠE PERFORMANSE** – ne koristimo sekvencijalnu ako imamo puno **ažuriranja** (redosledna obrada)

može se poboljšati: u datoteci promena **prikupljamo** sve promene i kada se one nakupe i kada smatramo da je isplativo ponovo kreiramo našu novu datoteku (**reorganizacija**)

- **stara datoteka** (obradjivana i sekvencijalna) – trenutna
- **datoteka promene** (vodeća) – prvo je to **serijska** u koju hronološki upisujemo promene pa je sortiramo i dobijemo **sekvencijalnu** dat. promene (kako bismo imali 1 **pristup** bloku vodeće i obradivane jer su obe sekvencijalne)
- generišemo **novu izlaznu datoteku** na osnovu dat. promene, a sve greške pišemo u **izlaznu dat. greške**

\***prednost** je što nema pomeranja, a **mana** jer nije ažurna

format sloga datoteka:

- stara – vr. ključa, ostala obeležja
- promena – vr. ključa, obeležja, polje za **operaciju** (unos, brisanje, modifikacija)
- greške – vr. ključa, obeležja, polje za **komentar o greški**

br. blokova nove dat:  $B_n = (\text{br. slogova stare} + \text{nove} - \text{brisanja} + 1 \text{ za kraj}) / f$

## OBLAST PRIMENE I OCENA KARAKTERISTIKE

- **prednost:** **odlična za redoslednu obradu**
- **nedostatak:** nepogodna za direktnu obradu, košta ažuriranje
- koristi se kao memorijска zona за indeks-sekvencijalne

## 8. RASUTA ORGANIZACIJA DATOTEKE

### RASUTE ORGANIZACIJE DATOTEKA

- dodela lokacije se vrši **transformacijom vr. ključa**
- naziva se i **direktna** jer zadajemo adresu i tako direktno pristupamo - odlična je za direktnu obradu
- **ne momorišu se logičke veze**, pa je jako loša za redoslednu obradu
- **postoji veza između vr. ključa i lokacije**
- blokirana - direktan pristup bloku (**baket**) i u njemu ima max f/b (**faktor baketiranja**) slogova

**IDENTIFIKATOR** – skup obeležja čije vrednosti jednoznačno određuju slogove datoteke

- **interni** – čine ga obeležja koja su *obeležja datog tipa sloga* -> **ključ**
- **eksterni** – ne pripada skupu obeležja datog tipa sloga

### TRANSFORMACIJA

- vrednosti identifikatora u adresu – vrednost id-a se preslikava na skup adresa
  - **h: dom(K) → A**, K – domen identifikatora, A – skup adresa lokacija mem. prostora dat.
1. **deterministička**:
    - 1-1 injektivna
    - svakoj vr. id. odgovara 1 adresa
    - svakoj adresi odgovara max 1 vr. id.
  2. **probabilistička**:
    - svakoj vr. id. odgovara 1 adresa
    - 1 adresi može odgovarati **više rezultata transformacije**
    - metoda za generisanje pseudoslučajnih brojeva

\*problem je jer možemo za **različite vrednosti ključa** da dobijemo **istu adresu**

\*korisitmo probabilističku kako bismo slogove što **ravnomernije** raspodelili po datoteci

\*npr. 1. radnik mbr = 1, 2. radnik mbr = 72, 3. radnik mbr = 150, ako bismo korisitli determinističku transformaciju ključa onda bismo morali da zauzmemos 150 lokacija a na samo 3 lokacije će biti smešteni radnici, pa deterministička nije pogodna jer se zauzima **bespotrebno više memoriskog prostora**

\*kada je broj mogućih vrednosti za ključ >> od potrebnih lokacija (u ovom slučaju 150>3) onda koristimo probabilističku, a u zavisnosti od raspodele vr. ključa imamo različite probabilističke transformacije koje su pogodne

**prekoračilac** – slog koji ne može da stane u matični baket (onaj baket koji vrati transformacija vr. ključa)

Način alokacije memorijskog prostora:

- statički – alocira se unapred, nema proširenja
- dinamički – može da se doalocira

## FORMIRANJE STATIČKE RASUTE

dodeljuje se  $Q = bB$  lokacija

- $b$  – faktor baketiranja,  $B$  – broj baketa
- $Q \geq N$  **treba nam min N lokacija** ( $N$  je br. slogova, procenjujemo)
- $Q$  se **ne može menjati** jer je statička

faktor popunjenoosti:  $q = \frac{N}{Q}$ ,  $0 < q \leq 1$ ,  $N$  – aktuelni br. slogova,  $Q$  – br. zauzetih lokacija

- **ako nemamo prekoračioce imamo 1 pristup baketu**
- ako *dobro procenimo* odložićemo prekoračioce, jer što ih je više to je lošije jer postaje nepredvidivo
- redosled slogova nebitan
- *hronološki* se upisuju
- upisu sloga prethodi 1 neuspešno traženje, na osnovu obavljene transformacije id-a u adresu
- slog se smešta u baket sa izračunatom adresom

## DIREKTNA I RELATIVNA ORGANIZACIJA DATOTEKE

### DIREKTNA ORGANIZACIJA DATOTEKE

- **eksterni** identifikator, **deterministička** transformacija  $h: A \rightarrow A$
- **vr. id. je i adresa baketa**:  $A_i = k_i$
- nema preslikavanje veza između sadržaja slogova i adresa

#### 1. direktna datoteka sa **mašinskim** adresama:

- adresa baketa na disku
- ima dobru brzinu
- mana je jer **zavisi od uređaja** gde se datoteka nalazi
- **odsustvo** logičke veze između vr. identifikatora i sadržaja sloga

#### 2. direktna datoteka sa **relativnim** adresama:

- korišćenje relativnih adresa slogova

- lokacije se numerišu **rednim brojevima** 1 – Q / 0 – Q-1
- oslobođamo se problema čvrste povezanosti sloga sa memorijskim uređajem
- nedostatak je i dalje što nema veze između vr. identifikatora i sadržaja sloga

**RELATIVNA METODA PRISTUPA:** transformiše relativnu u mašinsku adresu i radi na nivou 1 bloka: 1 blok = 1 slog (kao da nema baketiranja)

- mi (programeri) – treba da napravimo MP kako bismo imali blokiranje i rastavljanje blokova na slogove kako bismo efikasnije iskoristili mem. prostor (\*od MP zavisi koja vrsta organizacije dat.)
- SUBP
- Sistem za upravljanje datotekama
- biblioteke – transformacija relativne u mašinsku, usluge na nivou sloga, nema veze između vr. ključa i rel. adrese, ne prave razliku između organizacija, moguć direktni pristup

### FORMIRANJE **RELATIVNE DAOTOTEKE**

- u **posebnom postupku**
- na osnovu **vodeće** serijske/sekvenčne
- učitamo slogove iz vodeće i redom se generišu argumentni traženja za obradivanu
- u toku formiranja treba da povežemo eksterni identifikator sa vrednošću sloga (dodelimo mu odgovarajuću relativnu adresu i tu ga smeštamo)

traženje logički narednog/slučajno odabranog – zadamo rel. adresu i vidimo gde se nalazi slog

### AŽURIRANJE RELATIVNE DATOTEKE

- upis novog sloga – pridružuje mu se eksterni identifikator, prethodno 1 neuspešno traženje
- brisanje – logičko -> izmenimo polje statusa sloga

### OCENA KARAKTERISTIKA RELATIVNE DATOTEKE

- **traženje slučajno odabranog – najefikasnije**
- traženje logički narednog – efektivnije od serijske, lošije od sekvenčne, može eventualno vodeća da odredi logički narednog
- uvek imamo samo 1 pristup
- nema zavisnosti od mem. uređaja jer se uvela relativna adr. lokacije kao identifikatora
- nema i dalje veze između vr. identifikatora i sadržaja sloga
- primena: osnov za izgradnju indeksnih i spregnutih datoteka

### **STATIČKA RASUTA ORGANIZACIJA DATOTEKE**

- **probabilistička** transformacija ključa (interni identifikator)
- za različite vrednosti ključa možemo da dobijemo iste adrese

$v^p \gg Q > N$

- $v^p$  = br. mogućih vr. ključa, imamo v dozvoljenih vrednosti koje može da ima svaka pozicija p
- Q = Bb - broj lokacija (procenujemo), B - broj baketa, b - broj lokacija u baketu
- N - broj aktuelnih slogova u datoteci

npr. moramo 9 ljudi da smestimo u memorijske lokacije na osnovu ključa koji je broj lične karte od 8 cifara -> onda su moguće vrednosti ključa  $10^8$  i ljude smeštamo na neku od 1-9 (0-8) lokacija. Tada koristimo probabilističku transformaciju kako bismo ih ravnomerno rasporedili i jer bi deterministička zauzela bespotrebno mnogo lokacija ali problem sada mogu biti prekoračioci

### CILJEVI:

- što **ravnomernija** raspodela slogova
- **pseudoslučajna** transformacija vr. ključa u adresu
- pravilno dizajniranje potrebnog **adresnog prostora**

## 4 KORAKA PROBABILISTIČKE TRANSFORMACIJE:

1. pretvaranje nenumeričke u numeričku vrednost ključa:  $k(S) \in \{0, \dots, v^p - 1\}$ , ako je već broj onda preskačemo ovaj korak. p - br. cifara za vrednost ključa, v - osnova brojnog sistema ( $v=10$ )
2. pretvaranje  $k(S)$  u pseudoslučajan broj  $T(k(S))$ ,  $T \in \{0, \dots, v^n - 1\}$ 
  - n - dozvoljeni broj cifara relativne adrese  $A \in \{1, \dots, B\}$ ,  $n = \lceil \log_v B \rceil$ ,  $1 \leq n \leq p$
3. T dovodimo u opseg dozvoljenih vrednosti relativne adrese  $A \in \{1, \dots, B\}$ 
  - $A = 1 + \lfloor kT \rfloor$ ,  $k = \frac{B}{v^n}$ , B - br. blokova,  $v^n$  - max br. vrednosti lokacije
4. pretvaranje relativne u mašinsku adresu - to ne radimo jer to rade funkcije iz biblioteke koju smo korisili za MP

## 3 METODE PROBABILISTIČKE TRANSFORMACIJE:

### 1. METODA OSTATKA PRI DELJENJU

- $T = k(S) \text{(mod } m\text{)}$ ,  $k(S)$  - vr. ključa sloga s, m je ceo broj; celobrojni ostatak pri deljenju vr. ključa
- $$k = \frac{B}{m}$$
- preporuke za biranje m - neparan, prost broj, obično se bira da je  $m = B$  (broju blokova)
- pogodna: kada se vr. ključa javljaju u paketima, između su intervali sa neaktivnim vr. ključa jer slogovi sa sukcesivnim vr. ključa iz paketa dobijaju adresu fizički susednih baketa

primer:

- $k(S) = 34$   
 $B = 5, m = 5$
- (ostatak kada se  $34:5$  je 4)  $\Rightarrow T = 4$   
 $A = 1 + 4 = 5$

## 2. METODA CENTRALNIH CIFARA KVADRATA

- polinomijalni zapis:  $(k(S))^2 = \sum_{i=0}^{2p-1} c_i v^i, c^i \in \{0, \dots, v-1\}$  vr. ključa se diže na kvadrat
- krećemo od pozicije  $t = \lfloor p - \frac{n}{2} \rfloor$  i biramo n cifara (max br. cifara relativne adrese)  $c_t, \dots, c_{t+n-1}$
- formiramo T
- relativan adresa  $k = \frac{B}{v^n}$ , centriranje i normiranje T na zadati opseg relativnih adresa

primer:

- $p = 2, v = 10$   
 $B = 20 \rightarrow n = 2$  (br. cifara)  
 $t = 1, k = 0.2$
- $k(S) = 34$   
 $\Rightarrow k(S)^2 = 1156$   
 $\Rightarrow$  od 1. pozicije ( $t=1$ ) uzimem 2 cifre ( $n=2$ )  
 $\Rightarrow T = 15$   
 $\Rightarrow A = 1 + \lfloor 15 * 0.2 \rfloor = 4$

## 3. METODA PREKLAPANJA

- cifre ključa premeštaju se kao pri preklapanju hartije
- preklopljene vrednosti se sabiraju po modulu  $v^n$
- pogodna:  $p >> n$  (br. pozicija vr. ključa mnogo >> od br. pozicija relativne adrese n)
- $\lceil \frac{p}{n} \rceil$  - br. segmenata za preklapanje

primer:

- $p=6$  (vr. ključa ide do 6-ocifrenog broja),  $v=10$   
 $B=20, n=2$  (po 2 broja za preklapanje)  
 $\lceil \frac{p}{n} \rceil = 3, k=0.2$
- $k(S1) = 341201$   
 $\Rightarrow T1 = (01 + 21 + 34) \bmod 10^2 = 56$   
 $\Rightarrow A = 1 + \lfloor 56 * 0.2 \rfloor = 12$

**sinonimi** – slogovi kada se transformacijom vr. ključa dobiju iste relativne adrese

**matični baket** - baket čiju relativnu adresu dobijamo transformacijom ključa

**primarni slog** - slog koji je u svom matičnom baketu

### **PREKORAČILAC:**

- nije primarni
- nema mesta u svom matičnom baketu
- pripada skupu sinonima
- traži se novi postupak za smeštanje
- nepoželjna pojava jer se narušavaju performanse

Verovatnoća pojave sinonima zavisi od:

- raspodele vr. ključa unutar opsega dozvoljenih vrednosti
- odabrane metode transformacije
- faktora popunjenoosti memoriskog prostora:  $q = \frac{N}{Q}$  N – aktuelnih slogova, Q – zauzetih lokacija

Broj prekoračilaca će biti manji:

- što su slogovi **ravnomernije** raspoređeni po baketima
- što je **faktor popunjenoosti manji** (**q malo** – puno slobodnih lokacija, a što je **q bliže 1** to je dobro iskorišćen mem. prostor ali veća verovatnoća za prekoračioce, otprilike je  $q < 0.8$ )
- što je **faktor baketiranja veći** (**b veliko** – raste vreme prenosa bloka, **b malo** – veća verovatnoća za prekoračioce ali je bolja preciznost transformacije, bira se  $b \leq 10$ )

Pri projektovanju rasute datoteke se utvrđuju faktor popunjenoosti q, faktor baketiranja b, metoda transformacije vr. ključa u adresu, postupak za smeštanje prekoračilaca

### **Postupci za smeštaj prekoračilaca:**

- svi u **jedinstven** adresni prostor – datoteka ima samo 1 **primarnu** zonu
- svi u posebnu zonu adresnog prostora – datoteka ima **2 zone**: primarna i **zona prekoračenja** (serijska/spregnuta), pa će biti 2 datoteke fizički ali logički samo 1
- kombinacija – imamo i posebnu zonu prekoračenja, a i u primarnoj zoni mogu da se smeštaju prekoračioci

## **PROJEKTOVANJE I FORMIRANJE RASUTE DATOTEKE**

- implementiramo relativnu MP uz pomoć C-jezika i njegovih biblioteka
1. statička pa alociramo memorijski prostor
    - procena aktuelnog broja slogova, ažuriranja, faktora popunjenoosti
    - na početku je prazna sa formiranim baketima sa b (faktor baketiranja) slogova
    - oznaka da je slog prazan: status polje, specijalan znak, indeksiranje slobodnih lokacija i sprezanje u listu

2. upisujemo transformacijom ključa u adresu
  - formiramo je na osnovu serijske vodeće
  - direktno upisivanje slogova u realnom vremenu

Formiranje u posebnom postupku može biti u:

- 1 prolazu: hronološki, tražimo 1. slobodnu lokaciju i tu upisujemo, ako imamo [zonu prekoračenja](#)
- 2 prolaza: u 1. prolazu upišemo samo one što mogu da stanu u svoj matični blok, a u 2. prolazu ostale kako bi se smanjio br. prekoračilaca

## TRAŽENJE SLOGA

- [logički narednog = slučajno odabranog](#) jer nema memorijskih veza ali postoji veza između sadržaja sloga i adrese
- [idealno ako nema prekoračilaca](#) imamo samo **1** uspešno/neuspešno traženje
- ali pošto imamo ažuriranje – narušavamo performanse, transformacijom ključa odemo u matični baket i ako ga nema tu ali ima prekoračilaca, onda tražimo dalje pa postaje nepredvidivo koliko će traženje trajati jer zavisi od metode za smeštanje prekoračilaca

**METODE ZA SMEŠTANJE PREKORAČILACA:** linearna metoda, ponovna transformacija, metoda praćenja pokazivača

### Rasuta sa linearnim traženjem

#### 1. sa fiksним korakom k, k=1 najčešće

- $A_0 = h(k(S))$  – adresa matičnog baketa gde treba taj slog da se smesti, ako nema mesta za  $k=1$  tražimo 1. sledeću slobodnu lokaciju dok ne dođemo do slobodne lokacije
- traženje – transformacija do matičnog baketa, idemo baket po baket – kao za [serijsku](#) pa se i performanse narušavaju

- zaustavlja se: uspešno traženje (našli smo ga), neuspešno (došli smo do 1. slobodne lokacije, znači da je morao biti pre nje ili smo se vratili ponovo u matični baket na početak jer je datoteka skroz puna)
- upis novog – u 1. slobodnu lokaciju
- brisanje – logičko (status polje postavimo na slobodnu lokaciju), fizičko (brisanje iz matičnog – oslobođimo ga, pa 1. sledeći prekoračilac uskače u svoj matični baket, a svi prekoračioci se pomeraju za 1 mesto, ali se ne pomeraju slogovi koji su u svojim matičnim baketima)
- nedostaci: nagomilavanje prekoračilaca, tako što popune nečiji matični baket i onda oni koji treba da uđu u svoj matični baket ne mogu od njih, imamo neefikasno traženje

## 2. sa fiksnim korakom k, $k>1$ , uglavnom je $k=3$

- ako je matični baket popunjen, ide u narednu slobodnu lokaciju udaljenu za k pozicija
- kako bismo razbili nagomilavanje i dobili ravnomerniju raspodelu

## 3. sa slučajno odabranim korakom k, $k \geq 1$

- ako je matični baket popunjen, ide u narednu slobodnu lokaciju udaljenu za k pozicija
- k se određuje na slučajan način i predstavlja rezultat 2. probabilističke transformacije
- rasute sa otvorenim načinom adresiranja (fiksni korak  $k>1$  i slučajno odabrani korak) su pogodne kada imamo manji faktor popunjenoosti i manje ažuriranja

## Rasuta sa sprezanjem u PRIMARNOJ ZONI

E – zaglavlje datoteke, adresa

$A_i$  – pok. na lanac  
baketa sa slobodnim  
lokacijama

**A<sub>i</sub>:**

u pok. na 1. slog sinonima	b pok. na prethodni	n pok. na naredni	e br. slobodnih lokacija	A <sub>i</sub> <sup>1</sup>			A <sub>i</sub> <sup>2</sup>		
				k(S <sub>1</sub> )	p(S <sub>1</sub> )	u(S <sub>1</sub> ) pok. na sledeći slog	k(S <sub>2</sub> )	p(S <sub>2</sub> )	u(S <sub>2</sub> )

- A<sub>i</sub> – basket koji može da sadrži i prekoračioce i svoje primarne slogove
- u – pokazuje na 1. slog koji pripada matičnom basketu (A<sub>i</sub>) -> A<sub>i</sub><sup>1</sup> slog
- b – pokazuje na *prethodni basket* koji ima slobodne lokacije (pripada lancu basketa sa slobodnim lokacijama)
- n – pokazuje na *naredni basket* koji ima slobodne lokacije (pripada lancu basketa sa slobodnim lokacijama)
- e – br. slobodnih lokacija za trenutni basket (A<sub>i</sub>)
- u(S<sub>1</sub>) – pok. na sledeći slog koji pripada matičnom basketu A<sub>i</sub> (oni su sinonimi svi) -> A<sub>i</sub><sup>2</sup>

- imamo i lanac basketa koji nemaju slobodne lokacije
- imamo **lanac sinonima** i svaki slog ima pokazivač na sledeći sinonim
- ako je matični basket popunjen, slog se smešta u 1. slobodnu lokaciju iz lanca slobodnih lokacija
- dvostruki spregnuti lanac basketa sa slobodnim lokacijama za smeštanje novih slogova
- specijalan (multi) basket s pokazivačem na početak lanca
- za svaki matični basket po 1 lanac sinonima

traženje slučajno odabranog – **praćenje pokazivača**, uspešno (kad ga nađemo), neuspešno (došli **do poslednjeg u lancu sinonima**)

mana – isto je moguće **nagomilavanje**

prednost – **poboljšali smo traženje jer idemo samo kroz sinonime**

upis – 1 neuspešno traženje, i onda na osnovu zaglavlja datoteke iz lanca basketa slobodnih lokacija dođemo do 1. prazne lokacije i tu upišemo i uvezujemo u listu sinonima

brisanje – fizičko: uklanjanje sloga iz lanca sinonima, uz prevezivanje, i ako smo basketu izbrisali poslednji slog vraćamo basket u lanac basketa sa slobodnim lokacijama

## Rasuta sa sprezanjem u ZONI PREKORAČENJA

ako je matični baket popunjen, slog se smešta u 1. slobodnu lokaciju iz lanca slobodnih lokacija **u zoni prekoračenja – jednostruka lista baketa sa slobodnim lokacijama u zoni prekoračenja**, specijalan (multi) baket s pokazivačem na početak lanca

za svaki **matični baket po 1 lanac prekoračilaca**, u zaglavlju matičnog baketa je pokazivač na početak lanca, a svaki slog ima pokazivač na sledeći u lancu

tipičan faktor blokiranja  $f = 1$ , jer se ne očekuje veliki broj prekoračilaca ( $<10\%$ )

formiranje – **u 1 prolazu**

traženje slučajno odabranog sloga:

- transformacija vr. ključa u adresu i pristupanje matičnom baketu
- ako ga nema u matičnom, a nema prekoračioce (neuspjeh)
- ako ima prekoračioce onda postoji pokazivač na lanac baketa i tu se traži metodom praćenja pokazivača 1 po 1: ako smo ga pronašli (uspjeh), došli smo do kraja (neuspjeh)

prednosti – poboljšali smo performanse traženja (uspjeh/neuspjeh) i nema nagomilavanja

mane – **složenija struktura** i procedure

upis:

- 1 neuspjeh traženje
- ako u matičnom baketu ima mesta tu ga pišemo, ako nema upisujemo u zonu prekoračenja jer **u zaglavlju postoji pokazivač na početak lanca na prazne lokacije i na 1. praznu upisujemo** (ako je to 1. upisan slog onda njegovu adresu prebacimo u zaglavlj, ako nije onda samo uvezujemo)

brisanje:

- fizičko
- iz matičnog baketa oslobađa se lokacija, a ako ima prekoračilaca **1. iz lanca prekoračilaca dolazi u matični baket**
- iz zone prekoračenja – **prevezivanje** (ako se 1. briše onda pokazivač treba da pokazuje na njegov sledeći)

### Rasuta sa serijskom zonom prekoračenja

- pogodna za **mali broj prekoračilaca**
- $f = 1$  najčešće
- traženje od 1. do poslednjeg prekoračioca (kroz sve mora da prođe)

\*rasuta je idealna za **direktnu obradu i traženje slučajno odabranog sloga dok god nema prekoračilaca**

\*nepogodna jer je **statička**

## 9. INDEKS-SEKVENCIJALNA ORGANIZACIJA DATOTEKE

### INDEKSNE DATOTEKE

- indeksi = pomoćna struktura podataka – **stablo traženja**
- primarna zona – sa slogovima
- **zona indeksa – posebna zona**, sa parovima (replika **vr. ključa**, relativna **adresa sloga/bloka**)

Vrste indeksnih datoteka:

- **statičke** – nastale 1. kako bi se ubrzala pretraga, ali manja je jer se indeks ne ažurira pa dolazi do **debalansiranja stabla**
- **dinamičke** – datototeka sa B-stablom, ažuriraju se i indeksi

efikasnost – **sekvencijalna** (idealna za redoslednu obradu, traženje logički narednog), **rasuta** (za direktnu obradu, traženje slučajno odabranog), dok indeksna ima **solidne obe obrade i oba traženja**

### INDEKS-SEKVENCIJALNA ORGANIZACIJA

3 memorijske zone:

**primarna zona** – **sekvencijalna**, tu su **slogovi** (sve što važi za sekvencijalnu)

**zona prekoračenja** – spregnuta organizacija (**lanci prekoračilaca**)

**zona indeksa:**

- n-arno stablo (ima max n podstabala)
- **puno stablo traženja** (jednako svako rastojanje listova od korena)
- reda n ( $n \geq 2$ ) i visine h
- **n – faktor blokiranja**, a **1 čvor = 1 blok**, pa je **br. el. u čvoru = n**
- elementi u čvoru = **(vr. ključa, adresa bloka)**, sekvencijalno organizovani (rastući redosled)

**retko popunjeni indeks:**

- ne idu sve vr. ključa iz primarne zone u indeks nego samo **po 1 vr. ključa iz svakog bloka**
- min ili max vr. ključa

elementi **listova** stabla:

- sadrže po 1 vr. ključa iz svakog bloka
- pokazivač na **primarnu zonu na blok**

- ako se propagira **min** vr. ključa iz bloka onda - za 1. bloka se bira **najmanja moguća vr. ključa**
- ako se propagira **max** vr. ključa iz bloka onda - **poslednji** blok ima **najveću moguću vr. ključa**

elementi čvorova na višim nivoima:

- sadrže po 1 vr. ključa iz svakog direktno podređenog čvora
- pokazivač **na granu ispod**

## ZONA PREKORAČENJA

- za svaki blok ima po 1 lista prekoračilaca
- postoji pokazivač na slobodne lokacije u zoni prekoračenja

po što upisu prethodi 1 neuspešno traženje i ono se završava u bloku primarne zone, pa ako je taj **blok pun** onda u zonu prekoračenja ide onaj el. tako da se **očuva SORTIRANOST**

**brisanje:**

- fizičko - ako brišem iz **primarne** zone izbrišem i eventualno pomeranje **ulevo**, ako ima prekoračilaca **vraća se 1 prekoračilac u primarnu**, a ako brišem prekoračilac onda se vrši **prevezivanje**
- logičko - češće

**upis:**

- ako blok u **primarnoj** zoni ima mesta onda tu upišem i eventualno pomeranje **udesno**
- ako je blok pun, upisujem ili u primarnu ili u zonu prekoračenja tačno gde mu je mesto (da se **očuva rastući poredak**), jedan el. **će ići u zonu prekoračenja** i uvek će **poslednji** spregnut el. biti el. sa **najvećom vr. ključa za taj blok**

pokazivač na početak lanca prekoračilaca za svaki blok:

- **direktno** povezivanje – pok. u **listu**
- **indirektno** povezivanje – pok. u **bloku**

## DIREKTNI PRISTUP

**direktna** obrada (traženje **slučajno** odabranog)

idemo kroz stablo pa je bolje kada je pok. u **listu** jer u listu imam **if** koji odlučuje da li da nastavim traženje **u primarnoj zoni ili u prekoračiocima**

menjamo strukturu lista:

- **max vr. ključa u primarnoj zoni i pok. na blok**
- **max vr. ključa u zoni prekoračenja i pok. na nju**

ako su obe vr. ključa iste - nema prekoračilaca (kod inicijalnog formiranja)

## INDIREKTNI PRISTUP

kod **redosledne** obrade (traženje **logički** narednog) – bolje je kada je pok. u bloku kod traženja

## PERFORMANSE

- **problem traženja kroz spregnute liste prekoračilaca**
- traže privremenu **reorganizaciju** – onda nastaje primarna zona bez prekoračilaca
- 1 rešenje može biti da inicijalno bude **20% prazna primarna zona**, da ne idem odmah u prekoračioce, ali zahteva više memorije

# 10. B-STABLA

## B-stablo

- B = balansirano
- **puno** stablo – svaki list je podjednako udaljen od korena
- stablo **traženja** – levo podstablo sa manjim, desno sa većim vr. ključa
- **gusto popunjene indeks** – sve vr. ključeva iz primarne zone se propagiraju u stablo
- **dinamički indeks** – ažurira se, stablo se menja dinamički pri ažuriranju, ne traže reorganizaciju

### rang:

- ograničenje na **max i min broj el. u stablu**
- svaki čvor min **r**, max **2r** (koren može i 1 zbog početnog kreiranja)
- svaki čvor min 50%, max 100% popunjen
- npr. rang = 2 ->svaki čvor ima min 2, max 4 el. (osim u korenu)

$$\text{red} = 2r + 1$$

## FORMAT ČVORA

Zaglavljje bloka	P <sub>0</sub>							...				Neiskorišćeni prostor
		k <sub>1</sub>	A <sub>1</sub>	P <sub>1</sub>	k <sub>2</sub>	A <sub>2</sub>	P <sub>2</sub>		k <sub>m</sub>	A <sub>m</sub>	P <sub>m</sub>	

struktura elementa (sloga):

- k<sub>1</sub> -1. el.
- A<sub>1</sub> - adresa ka primarnom bloku gde je taj el.
- P<sub>1</sub> - adresa na podstablo

ograničenja:

- svi elementi su uređeni rastuće
- $P_0$  pok. na podstablo sadrži sve vr. ključa  $<k_1$
- $P_1$  sadrži sve vr. ključa  $>k_1$  i  $<k_2$
- $P_m$  pokazuje na vr. ključa  $>k_m$

dobro je i za redoslednu i za direktnu obradu:

- redosledna obrada (traženje log. narednog) – dole levo 1. el. je min vr. ključa u stablu, inorder obilazak
- direktna obrada (traženje sl. odabranog) – krećem od korena i idem levo/desno

poluprazno B-stablo – svaki čvor r, koren 1 (50% popunjenoš)

kompletno B-stablo – svaki čvor 2r (100% popunjenoš); manja h stabla je bolja, a to je kod kompletogn stabla – kada broj slogova raste, h ostaje mala jer je u formuli za računanje h stabla logaritam (što znači da ćemo i za mnogo slogova imati samo 2-3 pristupa)

formiranje: OS, programski jezici, SUBP

## PRIMARNA ZONA

- SERIJSKA datoteka (samo je preuzmemo), samo dodajemo na kraj novi, ali je loše traženje koje popravljamo korišćenjem indeksa

## ZONA INDEKSA

sregnuta struktura B-stabla

formiranje zone indeksa – nema poseban postupak, nego se formira kako se kasnije i upisuju novi el, prvo mora 1 neuspešno traženje slučajno odabranog koje se uvek završava u listu

popunjenoš listova:

- delimično popunjen list  $< 2r$
- potpuno popunjen list =  $2r$

**upis:**

1. čvor  $<2r$  dodamo u list i vrši se eventualno pomeranje udesno zbog poretku
2. čvor =  $2r$  balansiramo tako što radimo DELJENJE ČVORA:
  - prvih  $r$  el. do srednjeg ostane isto
  - srednji el. ide u roditeljski čvor i pokazuje na desno podstablo
  - ostalih  $r$  ide desno u novi doalocirani list

ako je i roditeljski isto napunjen isto se radi balansiranje, u najgorem slučaju dobićemo  $h+1$

broj bafera može biti:

- $h$  bafera – da bi ceo pristupni put mogao da stane u OM
- 1 bafer – više učitavanja čvorova i više pristupa istim čvorovima

- za redoslednu obradu je najbolje da imamo h bafera i kompletno stablo, a najgore 1 bafer i poluprazno stablo

## **BRISANJE**

- 1 uspešno traženje koje se može završiti bilo gde
- razlikuje se brisanje iz lista ili iz onog koji nije list

**brisanje iz list >r** samo ga uklonimo i pomerimo **ulevo** ostale, a **brisanje iz lista = r:**

## **TEHNIKA POZAJMLJIVANJA**

- 1 sused sa > r**
- formiramo niz u OM od susednog čvora, nadređenog el, i trenutnog čvor
- $\frac{1}{2}$  ostane levo, srednji el. ide gore, ostatak ide desno

\*kada se el. penje gore ostavi svoj pok. desno u  $P_0$ , a kada se spušta preuzima ga

## **TEHNIKA SPAJANJA**

- susedi = r**
- u OM niz od **r** el. suseda + **1** nadređeni + **r - 1** iz čvora iz kog se briše
- svih 2r el. ostaje u levom** čvoru, a **desni čvor se dealocira**

ako brišemo **el. koji nije u listu** uzmemo 1. sledeći el. veći od njega koji se nalazi u **desnom podstablu dole levo** i sve ide isto – jer smo ga pretvorili u list

prednosti:

- dobra** i redosledna i direktna **obrada**
- ne kvari** je ažuriranje
- nema zone prekoračenja**

mane:

- zbog deljenja čvorova **favorizacija polupraznog stabla** 50% popunjeno (kod sekvencera)
- kod **redosledne** obrade se pristupa gornjim čvorovima

## **B\***

favorizuje **75%** popunjeno

jedina razlika je **PRI UPISU – 1. POKUŠA TEHNIKU PRELIVANJA, ne ide odmah u deljenje čvora**

## **TEHNIKA PRELIVANJA**

- čvor u koji se upisuje = 2r, bar 1 sused <2r**
- susedni čvor + 1 nadređeni + trenutni prepunjeni

- $\frac{1}{2}$  levo, srednji gore, ostatak desno

## B#

- varijanta B\* stabla
- izmena u postupku **UPISA**
- minimalna popunjenošć 66%

## TEHNIKA DISTRIBUIRANOG DELJENJA

- **svi = 2r**, ne ide 1. prelivanje nego se vrši **DELJENJE**
- umesto od 1 čvora da pravim 2 poluprazna (na kraju bi bilo 4 čvora), od 2 čvora pravim 3 čvora
- 1. puni čvor + 2. puni čvor + **alociram novi 3. čvor**
- $2r + 2r + 1$
- $\frac{1}{3}$  ide u 1. čvor, sledeći gore, naredna  $\frac{1}{3}$  ide u 2. čvor, sledeći gore, ostatak u 3. novoalocirani čvor

## B<sup>+</sup>

- u praksi zastupljeno
- **problem kod B-stabla je redosledna obrada jer je išla po svim nivoima**
- B<sup>+</sup> poboljšava performanse traženja logički narednog (redosledna obrada)

strukturalna razlika:

- **u listovima će biti sadržane sve vr. ključa**
- čvorovi listova **dvostruko spregnuti**
- svaki el. koji bi išao u više nivoe stabla ostaje sačuvan i u listu
- svi **neterminalni** čvorovi pokazuju samo na **podstabla**, nemaju više pok. na primarnu **zonu** jer će se on pojaviti u **listu** i tu piše gde je u primarnoj zoni

## 1 IZMENA ALGORITMA ZA UPIS I BRISANJE

- kada delim list:  $\frac{1}{2}$  u **levi** čvor, **srednji** gore, a u **desni** čvor **OPET SREDNJI PONOVLJEN** i ostatak
- kada se list deli, onda su **gore ponovljene najmanje vr. iz svakog lista**
- sada se u redoslednoj obradi ide samo kroz listove, ali cena toga je ponavljanje nekih el.
- traženje sl. odabranog će se isto završavati uvek u **LISTU**

performanse:

- $B^+$  bolje u svim karakteristikama od B-stabla jer je format el. samo vr. ključa i 1 pok, a kod običnog B stabla je vr. ključa i 2 pok.
- napravićemo veći rang što je bolja osnova logaritma – imaćemo manje čvorova i manju h