

# **Baze podataka 1**



# Sadržaj

Osnovni pojmovi .....	1
Realni sistem i informacioni sistem .....	1
Entitet i poveznik .....	2
Obeležje, domen i podatak .....	2
Tip entiteta i pojava tipa entiteta .....	4
Tip poveznika i pojava tipa poveznika.....	6
Strukture podataka .....	8
Logička struktura obeležja (LSO).....	8
Logička struktura podataka (LSP) .....	9
Fizička struktura podataka (FSP) .....	11
Koncepcija baze podataka.....	12
Motivacija .....	12
Klasična organizacija datoteka.....	12
Baze podataka i SUBP .....	13
Šema baze podataka .....	13
Podšema / Eksterna šema .....	14
Pogled.....	15
Sistemi baza podataka .....	16
Modeli podataka.....	17
Pojam modela podataka .....	17
Strukturalna komponenta modela podataka .....	18
Integritetna komponenta modela podataka .....	18
Operacijska komponenta modela podataka.....	19
Modeli podataka.....	20
Model podataka tipova entiteta i poveznika.....	22
Osnovni pojmovi.....	22
Strukturalna komponenta .....	22
ER dijagrami .....	24
Integritetna komponenta.....	25
Kardinalitet tipa poveznika .....	27
Integritet tipa poveznika.....	29
N-arni tip poveznika ( $n > 2$ ).....	30
Gerund i agregacija .....	30

Id-zavisnost, IS-A hijerarhija i kategorizacija .....	32
Završne napomene .....	36
Koncepcija relacionog modela podataka .....	37
Mrežni i hijerarhijski model podataka .....	37
Relacioni model podataka .....	37
12 principa relacionih modela podataka .....	39
Osnove relacionog modela podataka .....	42
Strukturalna komponenta I.....	42
Operacijska komponenta .....	44
Strukturalna komponenta II .....	47
Integritetna komponenta.....	49
Osnovne projektantske pretpostavke.....	57
Fizičke strukture podataka i eksterni memorijski uređaji .....	58
Datoteka .....	58
Jedinice magnetnih diskova .....	58
Sprežni (U/I) podsistem .....	61
Performanse obrade podataka .....	64
Organizacija datoteke i OS.....	64
Datotečki sistem operativnog sistema .....	65
Usluge OS u organizaciji datoteka.....	65
Upravljanje memorijskim prostorom .....	66
Uparavljanje katalogom .....	66
Upravljanje fizičkom razmenom podataka .....	66
Obezbeđenje veze programa i datoteke.....	67
Sistemski pozivi .....	69
Metode pristupa.....	73

# Osnovni pojmovi

## Realni sistem i informacioni sistem

### Realni sistem - informacioni sistem

**Realni svet** - nešto što nas okružuje, gde egzistiramo, vidimo da postoji.

**Realni sistem** mora da poseduje:

- cilj poslovanja - sistem mora da ima deklarisane ciljeve, ako nema ciljeva nema sistema.
- resursi (činioci) poslovanja - raspoloživa dobra koja koristimo da bismo ostvarili cilj
- poslovni procesi - kolekcija aktivnosti koje se sprovode nad resursima da bismo postigli postavljeni zadatak ili cilj
- okruženje - postoji jasna demarkaciona linija između onog što pripada i ne pripada sistemu. Ono što ne pripada sistemu predstavlja njegovo okruženje i pripada realnom svetu.

**Informacioni sistem** je model realnog sistema (procesa i resursa). On je komponenta koja modeluje procese, resurse i pomaže ljudima u radu, tj. upravljanju realnim sistemom.

Cilj izgradnje informacionog sistema je pružanje informacija, neophodnih za funkcionisanje i upravljanje realnim sistemom.

Mesto informacionog sistema u realnom sistemu: Informacioni sistem je infrastrukturna komponenta realnog sistema, namenjena da podrži upravljački sistem realnog sistema.

Zadaci informacionih sistema su:

- obuhvat (akvizicija) podataka - npr. razni senzori, kamere, operator unosi podatke prateći proces poslovanja.
- skladištenje podataka - važna jer podatke najčešće ne koristimo u momentu kada nastaju
- prenos podataka - važna jer se podaci najčešće ne upotrebljavaju tamo gde su nastali
- prezentovanje podataka - korisnike interesuje da mogu imati pristup podacima, ne interesuju ih tehnički aspekti prikupljanja, skladištenja, itd.
- obrada podataka - funkcija transformisanja podataka iz jednog u drugi oblik, pomoću algoritama. Podaci nam često nisu potrebni u obliku u kome ih prikupimo.
- automatizacija upravljačkih funkcija u realnom sistemu

Činioci informacionih sistema su:

- računarsko-komunikaciona i softverska infrastruktura - sav sistemski softver, neophodan da bi naša arhitektura funkcionisala.
- baza ("skladište") podataka
- aplikacije (softverski paketi za rad sa podacima) - softverski proizvodi koji omogućuju praćenje procesa poslovanja i ponašanja našeg resursa u sistemu.
- projektna i korisnička dokumentacija - korisnička dokumentacija predstavlja uputstva za korisnike kako da upotrebljavaju sistem, dok projektna dokumentacija iskazuje sva tehnička rešenja na kojima je sistem napravljen. Važna da bismo taj sistem mogli da održavamo i da mu obezbedimo dug životni vek.
- krajnji korisnici - za njih pravimo informacioni sistem. Od njih dolaze zahtevi o funkcionisanju sistema.
- tim za obezbeđenje eksploatacije i održavanja - mora biti dobro postavljen

## Entitet i poveznik

### Entitet i klasa entiteta

**Entitet (realni entitet)** predstavlja jedinicu posmatranja. U realnom sistemu predstavlja činilac (resurs) poslovanja. Realni entiteti su sigurno realni, tj. egzistiraju u realnom sistemu. Entitet je "nešto" što se može jednoznačno identifikovati.

**Klasa realnih entiteta** je skup "sličnih" entiteta, tj. skup entiteta koji poseduje zajedničko svojstvo. Formalni prikaz je  $E = \{e_i \mid P(e_i)\}$ . Klasama realnog entiteta već opisujemo nešto na višem nivou apstrakcije. Klasa entiteta je apstrakcija, ona je samo pravilo kako posmatramo entitete na određeni način prema nekim osobinama.

### Poveznik i klasa poveznika

Entiteti realnog sistema se nalaze u međusobnim odnosima (vezama).

**Poveznik (veza)** reprezentuje odnos dva ili više realnih entiteta ili prethodno uspostavljenih poveznika.

**Klasa poveznika** je skup veza između klase realnih entiteta ili prethodno identifikovanih klase poveznika, tj. skup poveznika koji poseduje isto svojstvo. Formalni prikaz je  $S = \{e_1, \dots, e_m \mid P(e_1, \dots, e_m)\}$ , gde je  $e_i (i \in \{1, \dots, m\})$  jedan realni entitet ili prethodno uspostavljeni poveznik.

## Obeležje, domen i podatak

### Obeležje (atribut)

**Osobina (predikat)** jeste formalizam koji nam pomaže da razlikujemo da li je neki entitet/poveznik pripadnik klase entiteta/poveznika.

$P(e_i)$ ,  $P(e_1, \dots, e_m)$  - predikat (svojstvo) klase entiteta/poveznika. On iskazuje osobine klase  $E$ , tj. klase  $S$ .

Reći ćemo da predikati nose pojam obeležja.

**Obeležje (atribut)** je osobina klase realnih entiteta ili poveznika, a proističe iz semantike predikata  $P(e_i)$

Oznake:  $A, B, X, W, \dots, BRI, Datum\_Prispreća, JMBG, Prz, Ime$

Prema mogućnosti dekomponovanja na celine nižeg reda, obeležja možemo podeliti u tri vrste:

- elementarno - ne dekomponuje se, reprezentuje atomičnu (elementarnu) vrednost.  
Primer: *Grad, Ulica, Broj, Stan*
- složeno - može se dekomponovati na druga obeležja, reprezentuje složenu vrednost  
Primer: *ADRESA = (Grad, Ulica, Broj, Stan)*
- skupovno - reprezentuje skup vrednosti istog tipa

## Domen

**Domen** je specifikacija skupa mogućih vrednosti nekog obeležja, sa definisanim dozvoljenim relacijama i operacijama nad datim skupom. Domen reprezentuje skup mogućih vrednosti obeležja.

Vrste domena, prema načinu nastanka, su:

- predefinisani (primitivni) - "a priori" definisani
- korisnički definisani (izvedeni) - definisani korišćenjem postojećih domena, primenom unapred utvrđenih pravila.

## Domen obeležja

Svakom obeležju mora biti pridružen domen. Specificira se skup mogućih vrednosti obeležja.

$\text{Dom}(A)$ ,  $(A : D)$  je oznaka za pridruženi domen obeležju, dok je  $\text{dom}(A)$  oznaka za skup mogućih vrednosti obeležja.

Primer:

$$\text{DOCENA} := \{d \in \mathbb{N} \mid d \geq 5 \wedge d \leq 10\} \quad \text{specifikacija domena}$$

$$\begin{aligned} \text{Dom(Ocena)} &= \text{DOCENA} && \text{pridruživanje domena obeležju} \\ (\text{Ocena} : \text{DOCENA}) \end{aligned}$$

$$\text{dom(Ocena)} = \{5, 6, 7, 8, 9, 10\} \quad \text{skup mogućih vrednosti domena (posledica specifikacije i pridruživanja)}$$

## Podatak

**Podatak** je uređena četvorka (*Entitet, Obeležje, Vreme, Vrednost*):

- *Entitet* - identifikator (oznaka) entiteta.
- *Obeležje* - oznaka (mnemonik) obeležja.
- *Vreme* - vremenska odrednica.
- *Vrednost* - jedna vrednost iz  $\text{dom}(A)$ .

Podatak je vrednost nekog obeležja za dati konkretni entitet u datom konkretnom vremenskom trenutku.

**Kontekst podatka** je semantička (smisaona) komponentna podatka. On predstavlja trojku (*Entitet, Obeležje, Vreme*).

Ako se eksplisitno navede samo vrednost, a obeležje, entitet ili vreme nije ni implicitno zadato, to nije podatak, jer smisao nije određen.

Vreme, kao komponenta podatka, može se izostaviti ako se:

- uvede konvencija da se podatak, u tom slučaju, odnosi na vremenski trenutak u kojem se tim podatkom manipuliše ili
- identificuje posebno obeležje, čija vrednost predstavlja vremensku odrednicu posmatranog podatka

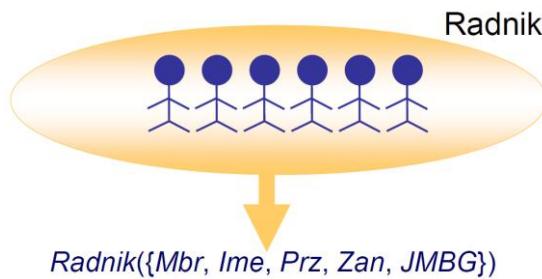
Podatak je činjenica iz realnog sistema. Ne mora svaki podatak biti informacija. Informacija je onaj podatak koji nam na bilo koji način umanjuje neznanje. Promenljiva postaje podatak tek kada dobije vrednost.

## Tip entiteta i pojava tipa entiteta

### Tip entiteta

**Tip entiteta** je model klase realnih entiteta u informacionom sistemu. Gradi se od obeležja bitnih za realizaciju ciljeva informacionog sistema. Tip entiteta poseduje naziv ( $N$ ) i skup obeležja ( $Q = \{A_1, \dots, A_n\}$ ). Skup obeležja tipa entiteta predstavlja podskup skupa obeležja klase realnih entiteta.

Primer:



### Pojava tipa entiteta

**Pojava tipa entiteta** je model jednog realnog entiteta u informacionom sistemu.

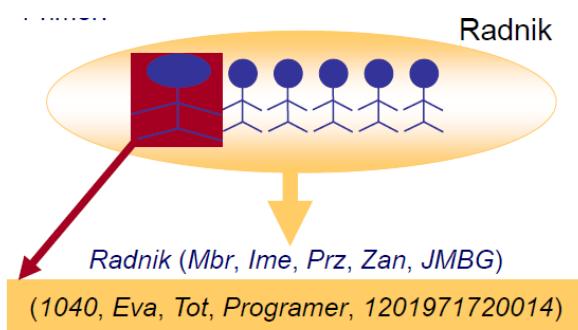
Tip entiteta formalno reprezentuje skup pojava:  $SP(N) = \{p_i \mid P(p_i)\}$

Svaka pojava  $p_i \in SP(N)$  reprezentuje tačno jedan realni entitet  $e_i \in E$ .

Ako je dat tip entiteta s nazivom  $N$  i skupom obeležja  $Q = \{A_1, \dots, A_n\}$ , tada pojava skupa tipa entiteta u zadatom trenutku vremena  $p(N, Vreme)$ , ili samo  $p(N)$  ako se vremenska odrednica ne navodi, predstavlja skup podataka:  $p(N) = \{(A_1, a_1), \dots, (A_n, a_n)\}$ , gde za svako  $A_i \in Q$  važi  $a_i \in dom(A_i)$ .

Ukoliko se u skup atributa uvede redosled  $(A_1, \dots, A_n)$ , tada se pojava  $p(N)$  posmatra kao  $n$ -torka (torka):  $(a_1, \dots, a_n)$ . Uređenje vrednosti podataka u pojavi je diktirano uređenjem obeležja u tipu entiteta.

Primer:



### Identifikator tipa entiteta

**Identifikator tipa entiteta** je skup obeležja, koji ima ulogu da obezbedi način za jedinstveno (nedvosmisleno) označavanje (identifikaciju) bilo koje pojave entiteta.

Bilo koja vrednost identifikatora tipa entiteta označava najviše jednu pojavu tipa entiteta, naziva se **identifikator pojave tipa entiteta** i predstavlja jednu od četiri komponente podatka.

Vrste identifikatora tipa entiteta su:

- eksterni - ne pripada skupu obeležja tipa entiteta
- interni - pripada skupu obeležja tipa entiteta

Primer:

Tip entiteta  $\text{Radnik}(\{Mbr, Ime, Prz, JMBG\})$

Eksterni identifikatori tipa entiteta:

$$\text{RBR\_Pojave\_TE} \in \{1, \dots, n\}, \text{Oznaka\_Pojave} \in \{p_1, \dots, p_n\}, \text{MEM\_Adresa\_Pojave} \in \{a_1, \dots, a_n\}$$

Interni identifikatori tipa entiteta:

$$Mbr, JMBG, \{Mbr, Ime, Prz, JMBG\}$$

## Ključ tipa entiteta

**Ključ tipa entiteta** je minimalni interni identifikator tipa entiteta. Formalno, ključ tipa entiteta predstavlja skup obeležja tipa entiteta  $N$ ,  $X \subseteq Q$ ,  $Q = \{A_1, \dots, A_n\}$ , takav da:

1. ne postoje dve pojave tipa entiteta sa istom x-vrednošću (za  $X$ )  
*(svojstvo jednoznačne identifikacije)*
2. ne postoji  $X' \subset X$ , za koji važi 1.  
*(svojstvo minimalnosti)*

Svaki tip entiteta poseduje bar jedan ključ, tj. predstavlja uređenu strukturu:

$$N(Q, C), \text{ gde je}$$

- $N$  - naziv tipa entiteta
- $Q = \{A_1, \dots, A_n\}$  - skup obeležja tipa entiteta
- $C$  - skup ograničenja tipa entiteta
- $K = \{K_1, \dots, K_m\} \subseteq C$  - skup ključeva tipa entiteta ( $K \neq \emptyset$ )

Skup svih pojave tipa entiteta  $SP(N)$  mora zadovoljavati  $C$ .

Primer:

$\text{Radnik}(\{Mbr, Ime, Prz, JMBG\}, \{Mbr, JMBG\})$   $Mbr$  i  $JMBG$  su dva, ekvivalentna ključa t.e. Radnik

**Primarni ključ** je jedan, izabrani, ključ iz skupa ključeva tipa entiteta. Često se označava podvlačenjem.

Primer:

$\text{Radnik}(\{Mbr, Ime, Prz, JMBG\}, \{\underline{Mbr}, JMBG\})$  ili skraćeno  $\text{Radnik}(\underline{Mbr}, Ime, Prz, JMBG)$

## Tip poveznika i pojava tipa poveznika

### Tip poveznika

Entiteti realnog sistema se nalaze u međusobnim odnosima (vezama) - poveznici. Informacioni sistem treba da sadrži model tih veza.

**Tip poveznika** povezuje dva ili više tipova entiteta ili prethodno definisanih tipova poveznika. On predstavlja model veza između pojava povezanih tipova entiteta i tipova poveznika, odnosno između realnih entiteta i veza.

Formalno, tip poveznika je struktura:

$$N(N_1, N_2, \dots, N_m, Q, C), \text{ gde je}$$

- $N$  - naziv tipa poveznika
- $N_i (i \in \{1, \dots, m\})$  - povezani tip (tip entiteta ili prethodno definisani tip poveznika)
- $Q = \{B_1, \dots, B_n\}$  - skup obeležja tipa poveznika
- $C$  - skup ograničenja tipa poveznika
- $K = \{K_1, \dots, K_k\} \subseteq C$  - skup ključeva tipa poveznika ( $K \neq \emptyset$ )

Tip poveznika reprezentuje skup pojava poveznika.

$$SP(N) \{(p_1, \dots, p_m) | P(p_1, \dots, p_m)\}$$

$p_i (i \in \{1, \dots, m\})$  - jedna pojava tipa entiteta ili tipa poveznika  $N_i$

$P(p_1, \dots, p_m)$  - osobina (predikat) tipa poveznika  $N$

Primer:

*Pohađa* (*Student*, *Predmet*,  $\{\text{Semestar}\}$ , *C1*) tip poveznika nad tipovima entiteta *Student* i *Predmet*

*Povera* (*Nastavnik*, *Predmet*,  $\{\text{Datum}\}$ , *C2*) tip poveznika nad tipovima entiteta *Nastavnik* i *Predmet*

*Ispit* (*Pohađa*, *Povera*,  $\{\text{Ocena}\}$ , *C3*) tip poveznika nad tipovima poveznika *Pohađa* i *Povera*

### Pojava tipa poveznika

**Pojava tipa poveznika**  $N(N_1, N_2, \dots, N_m, \{B_1, \dots, B_k\}, C)$  reprezentuje jedan poveznik u realnom sistemu. Oznaka je  $p(N, Vreme)$ , u zadatom trenutku vremena, a  $p(N)$ , ako se vremenska odrednica ne navodi.

Pojava tipa poveznika predstavlja skup podataka  $p(N) = (p_1, \dots, p_m)(N) = \{(B_1, b_1), \dots, (B_k, b_k)\}$ , gde važi da za svaki  $B_i$  mora biti  $b_i \in \text{dom}(B_i)$  i skup svih pojava  $p(N)$  mora zadovoljavati skup ograničenja  $C$ .

### Identifikator tipa poveznika

**Identifikator tipa poveznika** - niz  $(N_1, N_2, \dots, N_m)$  ili neki njegov neprazan podniz, ima ulogu da obezbedi način za jedinstveno (nedvosmisleno) označavanje (identifikaciju) bilo koje pojave tipa poveznika.

Bilo koja vrednost identifikatora tipa poveznika - niz  $(p_1, \dots, p_m)$ , označava najviše jednu pojavu tipa poveznika, naziva se **identifikator pojave tipa poveznika** i niz pojava ili jeste ili nije u vezi.

## Ključ tipa poveznika

**Ključ tipa poveznika** je skup obeležja  $X$ , izведен na osnovu ključeva povezanih tipova  $(N_1, N_2, \dots, N_m)$ . Vrlo često, ali ne i uvek, važi  $X \subseteq K_1 \cup \dots \cup K_m$ , gde  $(\forall i \in \{1, \dots, m\})(K_i \text{ je jedan izabrani ključ povezanog tipa } N_i)$

Formalno, ključ tipa poveznika predstavlja skup  $X = \{A_1, \dots, A_n\}$ , takav da:

1. ne postoje dve pojave tipa poveznika sa istom x-vrednošću (za  $X$ )  
*(svojstvo jednoznačne identifikacije)*
2. ne postoji  $X' \subset X$ , za koji važi 1.  
*(svojstvo minimalnosti)*

## Strukture podataka

**Struktura podataka** je orijentisani graf  $G(V, \rho)$ , gde je:

- $V$  - skup čvorova. Svaki čvor reprezentuje neke podatke. Svakom čvoru je pridružena određena semantika.
- $\rho$  - skup grana.  $\rho \subseteq V \times V$  - binarna relacija. Svaka grana reprezentuje neke veze između podataka. Svakoj grani je pridružena određena semantika.

Prema nivou apstrakcije pridružene semantike, strukture podataka se dele na:

- **logičke strukture obeležja**
- **logičke strukture podataka**
- **fizičke strukture podataka**

Prema mogućem broju direktnih prethodnika i sledbenika čvorova grafa, dele se na:

- **linearne** strukture podataka (cikličke i acikličke)
- strukture **tipa stabla** (drveta)
- **mrežne** strukture podataka

## Logička struktura obeležja (LSO)

**Logička struktura obeležja** je struktura nad skupom tipova entiteta, tipova poveznika i njihovih atributa.

Model dela realnog sistema (resursa):

$$M = (STE, RTE), \quad \text{gde je}$$

- $STE$  - skup tipova (entiteta i/ili poveznika - dva moguća pristupa)
- $RTE$  - relacija koja  $STE$  snabdeva strukturom. Modelira odnose koji postoje između realnih entiteta istih ili različitih klasa. Svaka grana  $RTE$  prikazuje jednu vezu tipa s nekim njegovim povezanim tipom.

Mogući pristupi organizaciji logičke strukture obeležja:

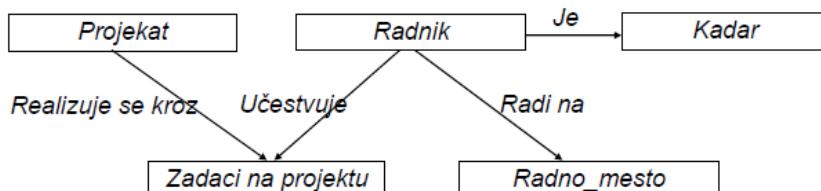
- **(A)** - " i tipovi entiteta i tipovi poveznika su čvorovi " -  $STE$  sadrži skup svih tipova entiteta i tipova poveznika modeliranog dela sistema.  $RTE$  sadrži grane koje prikazuju veze tipa poveznika s njegovim povezanim tipovima. Simboli za vizuelni prikaz čvorova mogu, a ne moraju biti različiti za tipove entiteta i tipove poveznika.
- **(B)** - " tipovi entiteta su čvorovi, a tipovi poveznika su grane " -  $STE$  sadrži skup svih tipova modeliranog dela sistema.  $RTE$  sadrži grane koje prikazuju sve tipove poveznika i veze s njihovim povezanim tipovima. Pristup zahteva redefiniciju pojma tipa poveznika. On ne sme da sadrži skup obeležja  $Q$  i skup ograničenja  $C$ . Ne može, kao povezani tip, da referencira drugi tip poveznika, već samo tip entiteta. Menja se pogled na upotrebu koncepta tipa entiteta. Problem je što iskazivanje tipa poveznika reda većeg od 2 zahteva korišćenje pojma hipergrane grafa.

Nivoi detaljnosti vizuelnog prikaza logičke strukture obeležja:

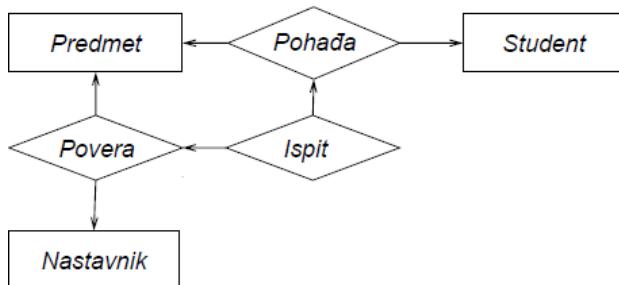
- nivo tipova entiteta i tipova poveznika (globalni prikaz)
- nivo obeležja (globalni prikaz)

Primeri:

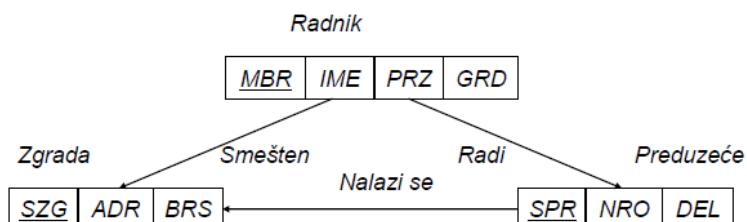
Nivo detaljnosti tipa entiteta i tipa poveznika. Pristup (B)



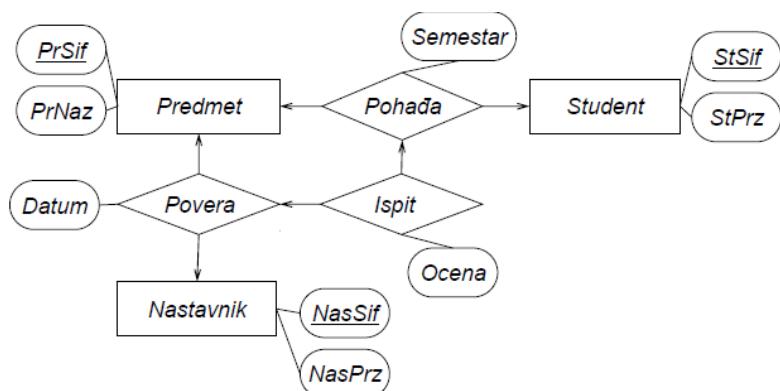
Nivo detaljnosti tipa entiteta i tipa poveznika. Pristup (A)



Nivo detaljnosti obeležja. Pristup (B)



Nivo detaljnosti obeležja. Pristup (A)



## Logička struktura podataka (LSP)

**Logička struktura podataka** definiše se nad skupom podataka, putem posebne relacije. Definiše se u granicama logičke strukture obeležja, koja predstavlja kontekst (model) za logičku strukturu podataka.

Logička struktura obeležja nad kojom je definisana logička struktura podataka, predstavlja **šemu** logičke strukture podataka.

Neke logičke strukture podataka:

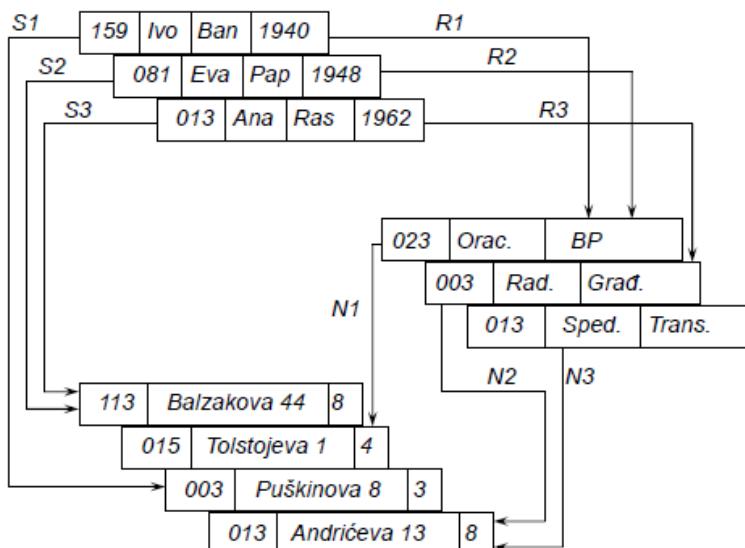
- Pojava tipa entiteta je linearna struktura nad skupom podataka jednog entiteta, datog tipa - *n-torka*, slog. Predstavlja složeni podatak, nad složenim obeležjem, dobijenim na osnovu skupa obeležja tipa entiteta  $Q$ . Kontekstna logička struktura obeležja je linearna struktura skupa obeležja datog tipa entiteta - *tip sloga*.
- Datoteka je struktura nad skupom pojava jednog tipa entiteta. Kontekstna logička struktura obeležja je linearna struktura skupa obeležja datog tipa entiteta - *tip sloga*.
- Baza podataka je logička struktura nad skupom pojava skupa tipova entiteta. Kontekstna logička struktura obeležja je struktura nad skupom tipova entiteta - *šema baze podataka*.

Načini vizuelne, a i memorije (fizičke) reprezentacije logičkih struktura podataka:

- putem grafova
- putem tabela

Primeri:

Reprezentacija logičke strukture podataka putem grafa



Reprezentacija logičke strukture podataka putem tabela

Radnik

MBR	IME	PRZ	GRD
159	Ivo	Ban	1940
081	Eva	Pap	1948
013	Ana	Ras	1962

Zgrada

SZG	ADR	BRS
003	Puškinova 8	3
013	Andrićeva 13	8
015	Tolstojeva 1	4
113	Balzakova 44	8

Preduzeće

SPR	NRO	DEL
03	Rad.	Grad.
13	Sped.	Trans.
23	Orac.	BP

Zaposlen

MBR	SPR
159	23
081	23
013	03

Nalazi se

SPR	SZG
03	013
13	013
23	015

Stanuje

MBR	SZG
159	003
081	113
013	113

## Fizička struktura podataka (FSP)

Fizička struktura podataka je logička struktura podataka smeštena na materijalni nosilac podataka - memorijski medijum. Uključuje podatke o samom načinu smeštanja logičke strukture podataka na memorijski medijum.

Fizička struktura podataka zahteva izbor pristupa i postupaka za:

- upravljanje slobodnim i zauzetim memorijskim prostorom
- izbor lokacija za smeštanje podataka
- kodiranje podataka
- formatiranje i interpretaciju sadržaja lokacija
- memorisanje veza u strukturi podataka
- kreiranje fizičke strukture podataka
- pristupanje podacima i njihovo selektovanje
- ažuriranje i reorganizovanje strukture podataka

# Koncepcija baze podataka

## Motivacija

Vrednost svakog sistema, pa i baza podataka najbolje se shvata ne samo na osnovu poznavanja samog sistema, već na osnovu činjenice da taj sistem predstavlja korak u evoluciji rešavanja onih problema, koje prethodni sistemi nisu mogli da reše.

Da bi se stekla precizna slika obazama, nije dovoljno samo definisati pojam baze podataka, potrebno je prvo baze podataka sagledati u kontekstu njihovog istorijskog razvoja.

## Klasična organizacija datoteka

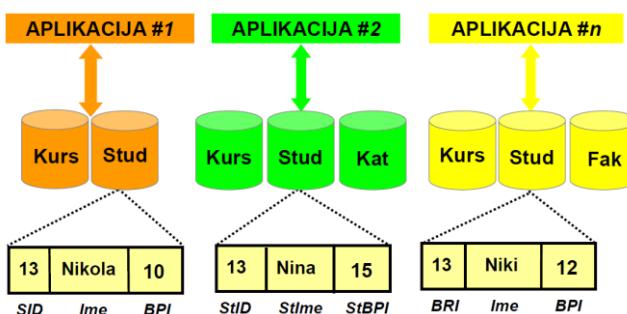
Informacioni sistemi bili su organizovani nad sistemima datoteka. Medijum za trajno memorisanje podataka bio je sistem diskova. Informacioni sistem je sačinjavao skup nezavisnih aplikacija. Svaka aplikacija imala je sopstvene daototeke, "skladištenje podataka" vršilo se u skupu datoteka, podaci o istom entitetu pojavljivali su se u različitim datotekama. Vremenom, taka informacioni sistem dolazi u kontradikciju sa samim sobom.

Osnovni nedostaci klasične organizacije datoteka:

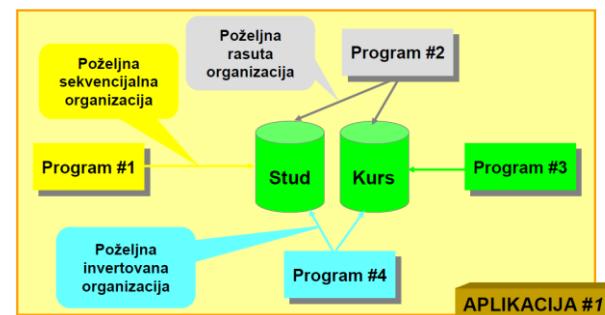
- nepovezanost aplikacija - potreba ručnog prepisivanja istih ili sličnih podataka
- redundantnost podataka - potreba višestrukog memorisanja istih ili sličnih podataka
- čvrsta povezanost programa i podataka - program vodi računa o fizičkoj strukturi podataka datoteke, kako u opisu, tako i u proceduri

Posledice ovih nedostataka su otežano održavanje i otežan dalji razvoj informacionih sistema.

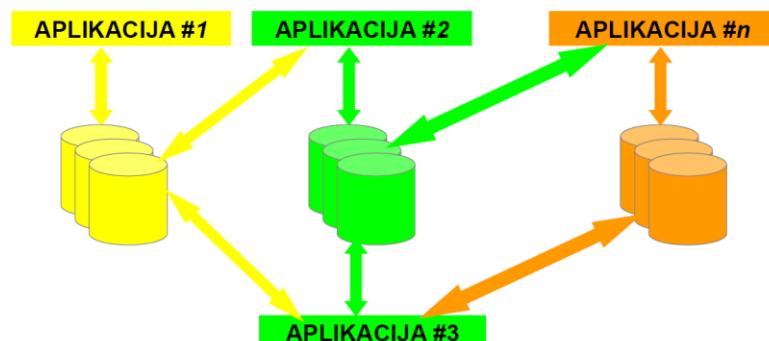
Primer nepovezanosti aplikacija:



Primer čvrste povezanosti programa i podataka:



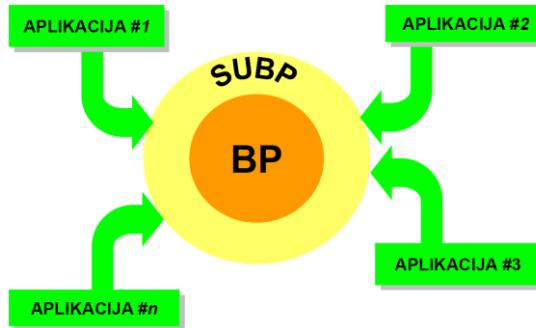
Problemi koji se mogu ublažiti, ili čak razrešiti u klasičnoj organizaciji su nepovezanost aplikacija i redundantnost. Problem koji je gotovo nemoguće ublažiti ili razrešiti je čvrsta povezanost programa i podataka.



## Baze podataka i SUBP

Osnovne ideje:

- da se svi podaci jednog informacionog sistema integrišu u jednu veliku "datoteku" - nastaje pojam baze podataka
- neredundantno memorisanje podataka - izbegavanje nepotrebnog višestrukog memorisanja istih ili sličnih podataka.
- da se uvede poseban softverski proizvod za podršku kreiranja i korišćenja baze podataka - sistem za upravljanje bazama podataka (SUBP). Svi programi bi trebalo da koriste podatke iz baze podataka ili je ažuriraju koristeći isključivo usluge SUBP.



Sistem za upravljanje bazama podataka (SUBP) (*Database Management System (DBMS)*) je softverski proizvod namenjen da omogući izgradnju i korišćenje baza podataka. Sadrži:

- jezik za opis podataka (*Data Definition Language - DDL*)
- jezik za manipulisanje podacima (*Data Manipulation Language - DML*)
- upitni jezik (*Query Language - QL*)

Jezgro SUBP:

- obezbeđenje fizičke organizacije podataka
- rutine za upravljanje podacima
- zaštita od neovlašćenog pristupa i od uništenja
- obezbeđenje višekorisničkog režima rada
- obezbeđenje distribuirane organizacije baze podataka
- obezbeđenje zadavanja šeme baze podataka (nad skupom obeležja ranijih datoteka formira se struktura šeme baze podataka, nad šemom baze podataka se kreira, koristi i ažurira baza podataka)

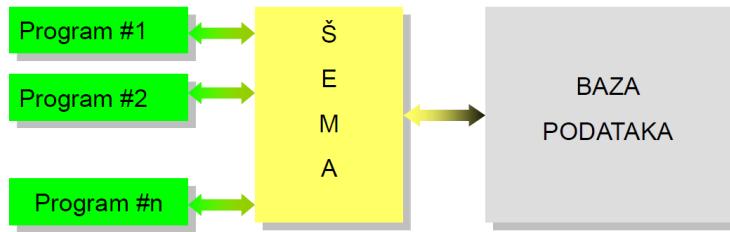
## Šema baze podataka

Program koji koristi usluge SUBP:

- poznaje samo šemu baze podataka, kao logičku kategoriju
- nad šemom baze podataka koristi logičku strukturu podataka (LSP), saglasno konkretnom zadatku
- ne sme da vodi računa o fizičkoj strukturi podataka (FSP), koja, po pravilu, može biti vrlo kompleksna

Preslikavanje LSP ↔ FSP je zadatak SUBP.

Primeri potreba različitih potreba za istim podacima: pristup saglasno rastućim vrednostima primarnog ključa, pristup saglasno vrednostima sekundarnog ključa, direktni pristup, saglasno zadatoj vrednosti primarnog ključa. Da bi se postiglo da svaki program pristupa istim podacima na različit način, potrebno je kombinovati nekoliko vrsta organizacije podataka.



Efekti uvođenja koncepta **šeme baze podataka (globalne šeme)**:

- smanjenje zavisnosti programa i šeme baze podataka od promena FSP
- smanjenje redundantnosti - povećanje konzistentnosti podataka
- uvođenje uloga: projektant baze podataka (šeme i FSP) i administrator baze podataka (DBA)

## Podšema / Eksterna šema

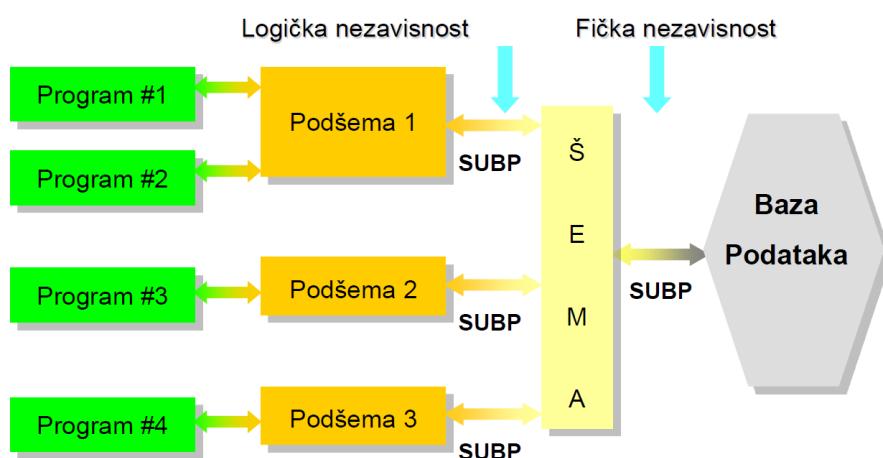
Šema baze podataka je, po pravilu, kompleksna i podložna češćim modifikacijama. U slučaju kada program direktno kojrsti koncepte šeme baze podataka, njene promene mogu izazvati česte i nepotrebne promene postojećih programa, što za posledicu ima otežano održavanje softverske podrške informacionog sistema. Rešenje za ovaj problem je uvođenje novog koncepta i novog sloja - podšeme.

**Podšema ili eksterna šema** je logička struktura obeležja (LSO) dobijana na osnovu dela šeme baze podataka. Potrebna je i dovoljna za realizaciju zadatka jednog ili grupe sličnih transakcionih programa, sličnih sa stanovišta modelovanih procesa poslovanja i korisničkih zahteva.

Podšema predstavlja model dela vaze podataka realnog sistema, za razliku od šeme baze podataka, koja predstavlja model cele baze podataka realnog sistema. Projektuje se, kao i šema baze podataka, u procesu razvoja informacionog sistema. Treba da egzistira kao projektantska specifikacija.

Poželjna je takva organizacija transakcionih programa da koriste bazu podataka isključivo putem podšema.

Preslikavanje Podšema ↔ Šema baze podataka može i poželjno je da bude zadatak SUBP. SUBP prevodi zahtev programa, definisan s obzirom na koncepte podšeme, u zahtev definisan s obzirom na koncepte šeme baze podataka. Podatke strukturirane s obzirom na koncepte šeme baze podataka prevodi u podatke strukturirane s obzirom na koncepte podšeme i obratno. Alternativno, pomenuto preslikavanje može biti delimično ili u celosti zadatak samog transakcionog programa (danas, često onog dela transakcionog programa koji upravlja logičkim strukturama podataka - u višenivovskim arhitekturama nalazi se na tzv. "srednjem" sloju - sloju aplikativne logike).

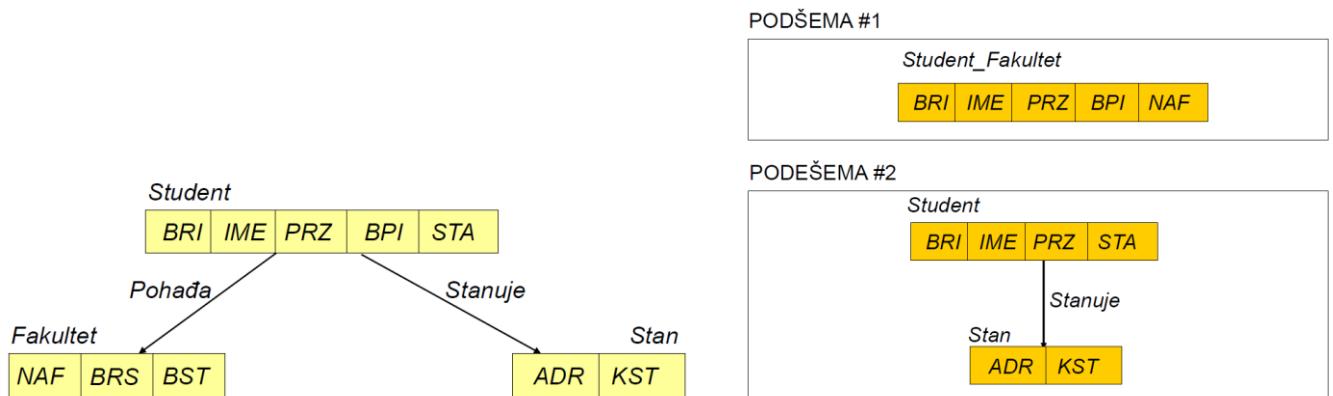


Efekat uvođenja koncepta podšeme je *logička nezavisnost* program od podataka. Promene šeme ne izazivaju promene podšeme i programa.

Efekat uvođenja koncepta šeme baze podataka je *fizička nezavisnost* programa od podataka. Promene FSP ne izazivaju promene šeme, podšeme i programa.

Fizička i logička nezavisnost su uslovne, a ne absolutne kategorije.

Primer male šeme baze podataka u mrežnom modelu podataka:



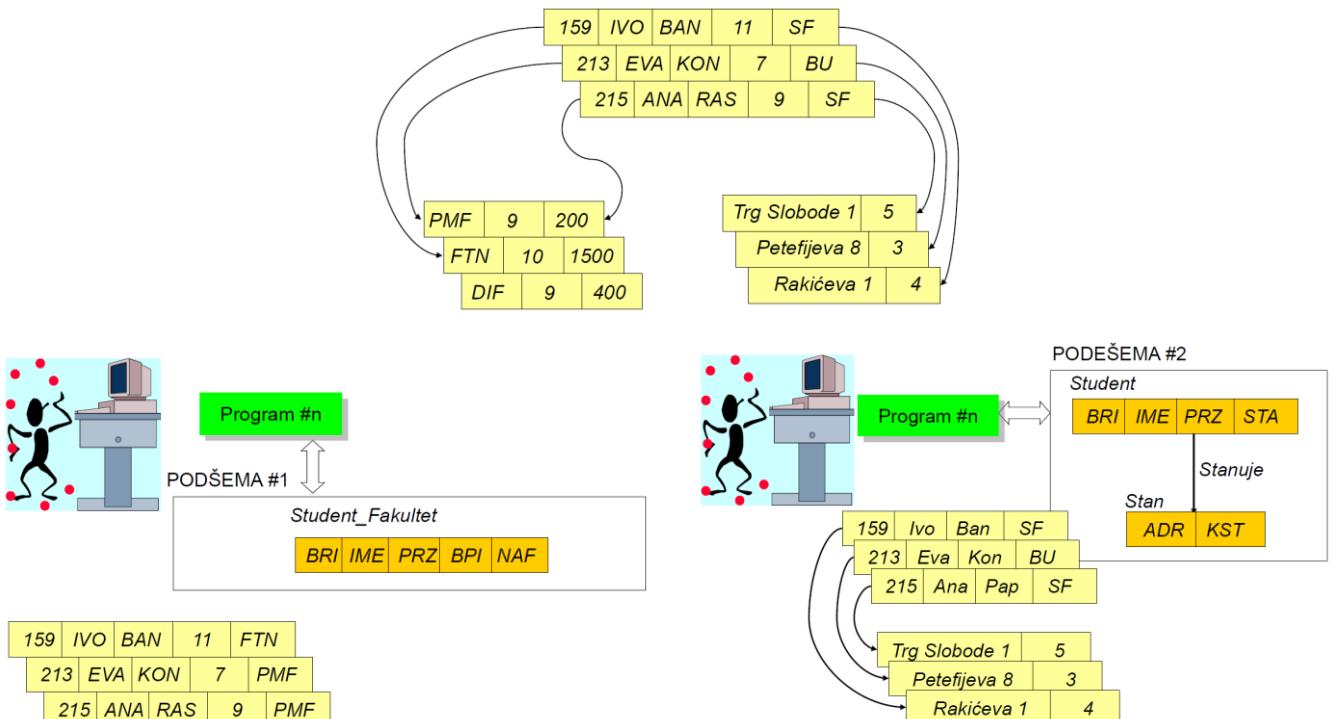
## Pogled

Šema i podšema su modeli na nivou apstrakcije obeležja, dok su globalni pogled i pogled modeli na nivou apstrakcije podataka.

**Pogled** predstavlja pojavu (LSP) nad podšemom. To je slika dela baze podataka kako je vidi programer ili korisnik.

**Globalni pogled** predstavlja pojavu (LSP) nad šemom baze podataka, tj. samu bazu podataka. To je slika stanja modelovanog dela sistema.

Primer globalnog pogleda i pogleda:

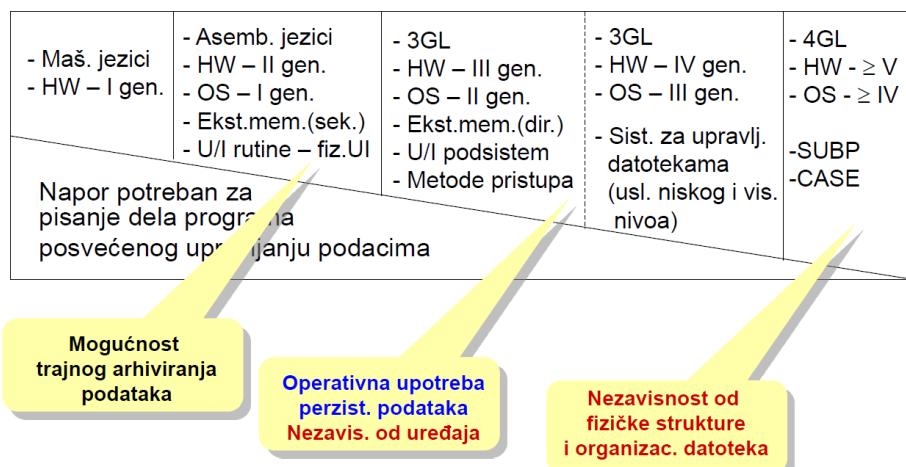


# Sistemi baza podataka

Sistem baze podataka obuhvata:

- bazu podataka
- SUBP, sistemski softver i računare (servere) na kojima je baza podataka kreirana
- šemu baze podataka, implementiranu na SUBP
- jezike i operacije za kreiranje, ažuriranje i korišćenje baze podataka.

Istorijat tehnološkog razvoja:



Gornja slika sugerije da je svaka nova etapa u razvoju programskih jezika i postupaka upravljanja podacima donosila i smanjenje napora potrebnog za pisanje dela programa posvećenog upravljanju podacima.

Opisane karakteristike sistema baza podataka predstavljaju ciljeve kojima treba težiti. U kojoj meri će ti ciljevi biti ostvareni, zavisi od projektanta baze podataka, izabranih koncepata, metoda i tehnika projektovanja, karakteristika SUBP. Razvoj postupaka za organizovanje i upravljanje podacima vodi ka povećanju produktivnosti razvojnog tima i izgradnji integrisanih informacionih sistema.

# Modeli podataka

## Pojam modela podataka

**Model podataka** je matematička apstrakcija, putem koje se gradi šema baze podataka. Šema baze podataka treba da predstavlja model baze podataka informacionog sistema, kao i pogled na strukture (model) posmatranog dela realnog sistema. Model podataka služi za predstavljanje:

- logičkih struktura obeležja (LSO) realnog sistema
- ograničenja u odnosima između podataka o stanjima realnog sistema
- dinamike izmene stanja realnog sistema, putem operacija nad podacima.

Model podataka  $M$  je matematička apstrakcija izražena trojkom  $(S, I, O)$ , gde je:

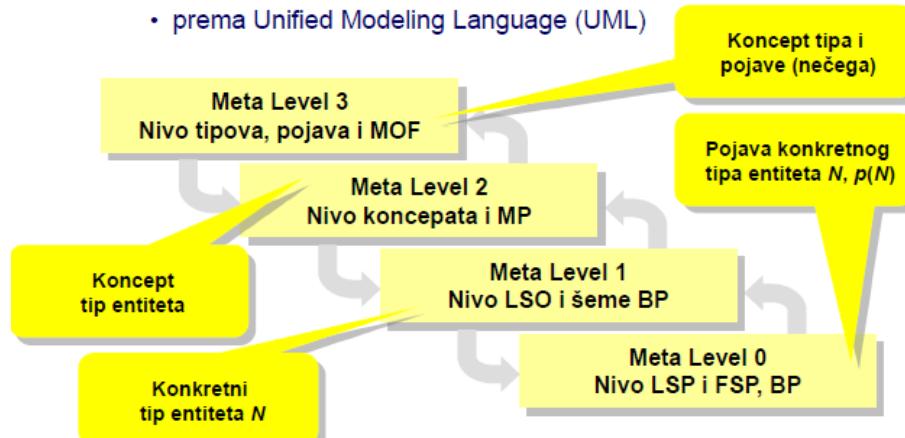
- $S$  - **strukturalna komponenta** - omogućava modeliranje logičke strukture obeležja, kao statičke strukture - šeme baze podataka.
- $I$  - **integritetna komponenta** - omogućava modeliranje ograničenja nad podacima u bazi podataka.
- $O$  - **operacijska komponenta** - podrazumeva modeliranje dinamike izmene stanja podataka u bazi podataka, a i same šeme baze podataka.

Nivoi apstrakcije određeni su modelom podataka. Postoje:

- **nivo intenzije (konteksta)** - nivo tipa (npr. nivo logičke strukture obeležja - šeme). Koristi se za definicioni opis nekog skupa. Definiše skup navođenjem uslova, koje njegovi elementi treba da zadovolje. Intenzija je generalizacija skupa ekstenzija. Definiše sve zajedničke osobine svojih ekstenzija.
- **nivo ekstenzije (konkretizacije)** - nivo pojave tipa (npr. nivo logičke strukture podataka). Odnosi se na prikaz jedne od mogućih pojava skupa nabranjem elemenata.

### • Nivoi apstrakcije

- u oblasti modelovanja sistema pa i sistema BP,
  - prema Unified Modeling Language (UML)



## Strukturalna komponenta modela podataka

**Koncept** je apstraktna (i formalna) predstava jedne klase pojmove, kojima se modeluju delovi realnog sveta.

**Primitivni (atomični) koncept** uvodi se i postoji "per se". Percipira se njegova semantika u realnom svetu. Ne može se dalje dekomponovati na koncepte nižeg reda. Predstavlja primitivni pojam, za koji najčešće nije moguće uvesti formalnu definiciju.

**Strukturalna komponenta** modela podataka sadrži:

- skup primitivnih koncepata - sa skupom datih osobina svakog koncepta, skupom pravila (definicija i osobina) za njihovo korišćenje i opisanom mogućom semantikom.
- skup formalnih pravila za kreiranje složenih koncepata - polazeći od primitivnih koncepata ili prethodno definisanih složenih koncepata, omogućava proširivanje inicijalno definisanog modela podataka.
- skup unapred kreiranih složenih koncepata - sa skupom datih osobina svakog koncepta, skupom pravila (definicija i šablonu) za njihovo korišćenje i opisanom mogućom semantikom.

Skup primitivnih i složenih koncepata, jednog modela podataka, za opis logičke strukture obeležja (nivo intenzije) i logičke strukture podataka (nivo ekstenzije):



## Integritetna komponenta modela podataka

**Integritetna komponenta** sadrži:

- skup tipova ograničenja (uslova integriteta) sa - skupom datih osobina svakog tipa ograničenja, koje uključuju pravila formalnog specificiranja i interpretacije (validacije, provere zadovoljenja), skupom pravila (definicija i šablonu) za njihovo korišćenje i opisanom mogućom semantikom.
- skup formalnih pravila za izvođenje zaključaka o važenju ograničenja (mogus ponens)
- skup formalnih pravila za kreiranje novih tipova ograničenja - polazeći od poznatih koncepata, omogućava proširivanje inicijalno definisanog modela podataka.

Pomoću tipova ograničenja iskazuju se konkretna ograničenja, koja se odnose na moguće vrednosti obeležja (podataka) ili moguće odnose između pojava povezanih tipova.

Baza podataka, čiji sadržaj je u saglasnosti sa svim definisanim uslovima integriteta - **konzistentna** b. p.

Primeri:

$$\begin{array}{ll} Radnik(\{MBR, PRZ, IME, JMBG\}, \{MBR, JMBG\}) & \text{ograničenje ključa (integritet entiteta)} \\ Dom(Ocena) := \{d \in \mathbb{N} \mid d \geq 5 \wedge d \leq 10\} & \text{ograničenje domena} \end{array}$$

Kardinalitet tipa poveznika: jedan nastavnik može predavati najviše jedan predmet.

Validacija ograničenja je provjeravaženja ograničenja. Može se ugraditi u transakcione programe ili specifikaciju šeme baze podataka, sa implementacijom u okviru SUBP. Rešenje kojem se može težiti je: Sva ograničenja podataka ugraditi u šemu baze podataka i prepustiti provjeru SUBP-u. Pojedina ograničenja ugraditi u transakcione programe.

# Operacijska komponenta modela podataka

**Operacijska komponenta** podrazumeva modeliranje dinamike izmene stanja u sistemu baze podataka. Ona sadrži skup tipova operacija sa skupom datih osobina svakog tipa operacije, koje uključuju pravila formalnog specificiranja i izvršenja nad podacima, skupom pravila (definicija i šablon) za njihovo korišćenje i opisanom mogućom semantikom

Operacijska komponenta definiše:

- upitni jezik/jezike (QL) - tipovi operacija za iskazivanje upita
- jezik/jezike za manipulisanje podacima (DML) - tipovi operacija za izmenu stanja baze podataka (ažuriranje), u cilju praćenja izmena stanja podataka u realnom sistemu.
- jezik/jezike za definiciju podataka (DDL) - tipovi operacija za kreiranje i modifikaciju specifikacija šeme b.p., fizičke strukture b.p., prava pristupa i zaštite b.p., novih tipova operacija (programa) za upravljanje podacima.

Ne menja svaka operacija iz operacijske komponente pojavu baze podataka, ali menja njenu stanje. Naime, pored podataka, baza podataka sadrži i niz kontrolnih mehanizama ili indikatora. Vrednosti ovih indikatora i pojava baze podataka čine *stanje baze podataka*. Najkarakterističniji kontrolni mehanizam predstavlja takozvani *indikator tekućeg sloga* (*indikator aktuelnosti*). Vrednost indikatora aktuelnosti često predstavlja adresu lokacije sloga, kojem se poslednje pristupilo. Taj slog naziva se *tekući slog*.

Specifikacija operacije sadrži dve komponente:

- **aktivnost** - specifikacija akcije nad podacima baze podataka
- **selekcija** - specifikacija dela baze podataka (u DML i QL) ili dela šeme baze podataka (u DDL), nad kojim se sprovodi specificirana aktivnost.

Moguće aktivnosti:

- definisanje vrednosti indikatora aktuelnosti (CURRENCY)
- čitanje podataka
- upis novih podataka
- brisanje postojećih podataka
- modifikacija postojećih podataka

Mogući načini selekcije:

- pomoću logičkog mesta u strukturi podataka, na osnovu vrednosti indikatora aktuelnosti
- putem odnosa između podataka
- putem vrednosti obeležja

Selekcija podataka putem vrednosti obeležja naziva se i **asocijativno adresiranje**. Iskazuje se putem logičkih izraza (kvalifikacionih izraza sa osnovnom strukturom (*osobina, uslov, vrednost*), koji se mogu povezivati logičkim operacijama). Moguća upotreba logičkih operatora (AND, OR, NOT), relacionih izraza i operatora {<, >, =, >=, <=, <>}, specijalizovanih operatora (IN, NOT IN, EXISTS,...), numeričkih, alfanumeričkih (string), logičkih i datumskeih izraza, funkcija i operatora, kao i upotreba obeležja kao operanada.

Primer:

Tip entiteta *Radnik*({*MBR, IME, PRZ, ZAN*}, {*MBR*})

*IME* = 'Ivo' AND *ZAN* IN ['Inž', 'Eko']      selekcioni izraz

Operacijska komponenta može biti:

- **proceduralna (navigaciona)** - selekcija vrši izbor jednog objekta iz baze podataka. Selekcija se vrši putem indikatora aktuelnosti ili putem odnosa između podataka. Karakteristična je proceduralnost sa programskim petljama i uslovnim grananjima, tj. za selekciju podataka mora se napisati program, koji, u opštem slučaju, sadrži sve tri osnovne programske strukture: sekvencu, selekciju i iteraciju. Sistemu za upravljanje bazom podataka mora se saopštiti ne samo šta se želi dobiti, već i kako to treba postići.
- **specifikaciona (deklarativna)** - selekcija vrši izbor skupa objekata iz baze podataka. Selekcija se vrši na osnovu vrednosti obeležja. Karakteristična je neproceduralnost. Definiše se samo šta (koji podaci) se želi dobiti, ali ne i kako to treba postići.

## Modeli podataka

Modeli podataka, koji su ostavili trajan trag, kako u teoriji, tako i u praksi baza podataka, su:

- Model tipova entiteta i poveznika (ER)
- Mrežni model
- Hjerarhijski model
- Relacioni model
- Logički i verovatnosni (fuzzy) logički modeli
- Objektno orijentisani model
- Objektno relacioni model
- XML model

**Model tipova entiteta i poveznika (ER)** je semantički model podataka. Postoje modifikacije, kao što su proširen model tipova entiteta i poveznika (EER) i model podataka konceptualnog nivoa ("blizi" korisniku po vrsti primenjenih koncepata). ER model nastao je na osnovama starijih modela, semantičke hjerarhije (Smith i Smith) i semantičkog modela (Hammer i Mcleod)

**Hjerarhijski model** je implementacioni model podataka. Tipične struktura šeme baze podataka je struktura stabla nad tipovima slogova. Operacijska komponenta je proceduralna. (Tipičan predstavnik: IBM DL/I sa programskim jezikom PL/I)

**Mrežni model** je implementacioni model podataka. Tipične struktura šeme baze podataka je struktura mreže nad tipovima slogova, koriste se tipovi setova. Operacijska komponenta je proceduralna. (CODASYL DBTG standard, tipični predstavnici: IDMS, IDS-II sa programskim jezikom Cobol)

**Relacioni model** je implementacioni model podataka. Tipične struktura šeme baze podataka je struktura tabela slogova – relacija, kao skupova n-torki. Operacijska komponenta je deklarativna. (ANSI SQL standard, tipični predstavnici RDBMS: Oracle, MS SQL Server, Ingres, Informix, Sybase, DB2, sa programskim jezikom SQL)

**Logički i verovatnosni (fuzzy) logički modeli** su dalja nadgradnja relacionog modela, uvođenjem dedukcije u baze podataka. Karakteristike:

- baza podataka-činjenica i baza pravila rezonovanja
- pridruživanje verovatnoća podacima u bazi
- rezonovanje u svetu rasplinute logike, na intervalu [0, 1]

**Objektno orijentisani model** zasnovan je na mrežnim i semantičkim modelima i objektno orijentisanoj paradigmi i programskim jezicima (koncepti klase, tipa, operacije i interfejsa). Objedinjeno je posmatranje struktura podataka i operacija nad podacima. Operacijska komponenta je proceduralna (C++, Java)

**Objektno relacioni model** je implementacioni model podataka. Kombinuje sve osobine relacionog i OO modela podataka. Savremeni ORDBMS nastaju evolucijom RDBMS i nasleđuju sve osobine RDBMS

**XML model** zasnovan je na XML jezicima i tehnologijama, paradigmi analognoj hijerarhijskom modelu podataka i tzv. "logičkim vezama". XML model je implementacioni model podataka. Tipična struktura šeme baze podataka je strukture stabla nad elementima i atributima. Šema baze podataka se opisuje putem XML Schema jezika. Operacijska komponenta je deklarativna. (ANSI SQL:2006 standard, XPath i XQuery jezici)

# Model podataka tipova entiteta i poveznika

## Osnovni pojmovi

Osnovni pojmovi ER modela (već uvedeni kroz osnovne pojmove baza podataka) su:

- obeležje i domen
- tip entiteta i pojava tipa entiteta
- tip poveznika i pojava tipa poveznika

## Entitet i klasa entiteta

Entitet je jedinica posmatranja, činilac (resurs) poslovanja u realnom sistemu.

Klasa realnih entiteta je skup "sličnih entiteta", tj. skup entiteta koji poseduje zajedničko svojstvo.

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

## Poveznik i klasa poveznika

Poveznik reprezentuje odnos dva ili više realnih entiteta ili prethodno uspostavljenih poveznika.

Klasa poveznika je skup veza između klasa realnih entiteta ili prethodno identifikovanih klasa poveznika, tj. skup poveznika koji poseduje isto svojstvo. Formalno:  $S = \{e_1, \dots, e_m \mid P(e_1, \dots, e_m)\}$ , gde je  $e_i (i \in \{1, \dots, m\})$  jedan realni entitet ili prethodno uspostavljeni poveznik.

## Strukturalna komponenta

Primitivni koncepti strukturalne komponente ER modela podataka:

- vrednost
- (predefinisani) domen
- obeležje

## Domen

Vrednost je bilo koja konstanta, iz bilo kog skupa.

Domen je specifikacija mogućih vrednosti obeležja, sa definisanim dozvoljenim realcijama i operacijama nad datim skupom. Vrste domena su:

- predefinisani (primitivni) domen - predstavlja predefinisani, atomični tip podataka, ugrađen u definiciju modela podataka
- korisnički definisani (izvedeni) domen - definiše se korišćenjem već postojećeg domena, putem pravila za definisanje domena, ugrađenih u definiciju (ER) modela podataka. Može predstavljati skup atomičnih podataka ili složenih podataka.

Pravila za definisanje korisnički definisanog (izvedenog) domena, ugradena su u definiciju ER modela podataka i ona definišu ugrađene relacije i operacije. Pravila su:

- pravilo nasleđivanja,
- pravilo tipa sloga,
- pravilo tipa skupa (kolekcije),
- pravilo tipa izbora

Primeri:

- $DPOZOCENA ::= \{d \in DOCENA \mid d \geq 6\}$
- $DTSLOG ::= Tuple\{(A_1; D_1), \dots, (A_n; D_n)\}$
- $DTISKUP ::= Set\{D_e\}$
- $DIZBOR ::= Choice\{(A_1; D_1), \dots, (A_n; D_n)\}$

## Obeležje (atribut)

Obeležje (atribut) je osobina klase realnih entiteta, iskazana putem predikata  $P(e_i)$ .

## Domen obeležja

Pravilo ER modela podataka:

*Svakom obeležju se pridružuje tačno jedan domen.*

Izvedeni koncepti strukturalne komponenete ER modela podataka:

- podatak
- tip entiteta
- pojava tipa entiteta
- tip poveznika
- pojava tipa poveznika

## Podatak

Podatak je uređena četvorka (*Entitet, Obeležje, Vreme, Vrednost*). Skraćeno (ako je poznat kontekst): (*Obeležje, Vrednost*) ili (*Vrednost*).

## Tip entiteta

Tip entiteta je model klase realnih entiteta u informacionom sistemu. Nastavje od obeležja klase realnih entiteta, bitnih za realizaciju ciljeva informacionog sistema. Predstavlja uređenu strukturu  $N(Q,C)$ .

## Pojava tipa entiteta

Pojava tipa entiteta je model jednog realnog entiteta u informacionom sistemu. Za tip entiteta  $N(Q,C)$ ,  $Q = \{A_1, \dots, A_n\}$ , pojava  $p(N)$  predstavlja skup podataka:  $p(N) = \{(A_1, a_1), \dots, (A_n, a_n)\}$

Ako se u  $Q$  uvede linearno uređenje obeležja, tada:  $p(N) = (a_1, \dots, a_n)$

## Tip poveznika

Tip poveznika je model veza između pojava povezanih tipova entiteta ili tipova poveznika. Predstavlja uređenu strukturu:  $N(N_1, N_2, \dots, N_m, Q, C)$

Identifikator tipa poveznika predstavlja niz  $(N_1, N_2, \dots, N_m)$  ili neki njegov neprazan podniz.

Ključ tipa poveznika izvenden je na osnovu ključeva povezanih tipova  $(N_1, N_2, \dots, N_m)$

Neka je  $K_i$  ključ tipa  $N_i$ . Ključ tipa poveznika je vrlo često, ali ne uvek, pravi ili nepravi podskup skupa ključeva  $K_1 \cup \dots \cup K_m$

$N_1, N_2, \dots, N_m$  ne moraju biti međusobno različiti tipovi. Svaki tip  $N_i$  u okviru tipa poveznika  $N$  ima svoju ulogu. Nad istim tipovima  $N_1, N_2, \dots, N_m$  se može definisati više različitih tipova poveznika.

## Pojava tipa poveznika

Pojava tipa poveznika  $N(N_1, N_2, \dots, N_m, \{B_1, \dots, B_k\}, C)$  reprezentuje jedan poveznik u realnom sistemu.

Oznaka je  $p(N, Vreme)$ , u zadatom trenutku vremena, a  $p(N)$ , ako se vremenska odrednica ne navodi.

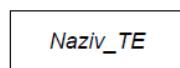
Pojava tipa poveznika predstavlja skup podataka  $p(N) = (p_1, \dots, p_m)(N) = \{(B_1, b_1), \dots, (B_k, b_k)\}$ .

## ER dijagrami

Model statičke strukture realnog sistema, realizovan putem ER modela podataka, po pravilu se predstavlja uz pomoć pogodne dijagramske tehnike, takozvanih **ER dijagrama**. ER model podataka uživa popularnost zbog dijagramskog načina prikaza šeme baze podataka.

Pravila crtanja pojedinih elemenata:

- Tip entiteta:



Kada se domeni na dijagramu ne prikazuju, vizuelna reprezentacija obeležja je:

- Tip poveznika:

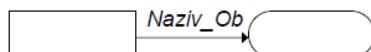


- Domen:



Obeležja primarnog ključa TE se podvlače

- Obeležje:



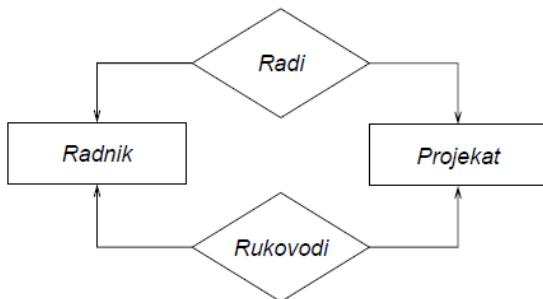
Nivoi detaljnosti ER dijagrama:

- **nivo naziva tipova** (globalni nivo prikaza)
- **nivo naziva obeležja** (detaljni nivo prikaza)

Primeri:

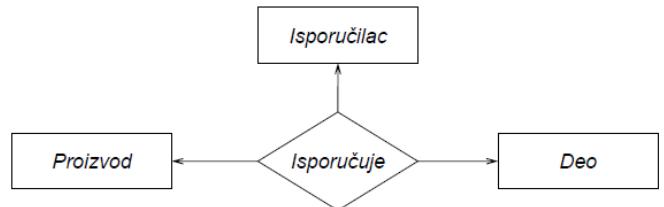
Nivo detaljnosti naziva tipova.

Dva tipa poveznika između istih tipova entiteta



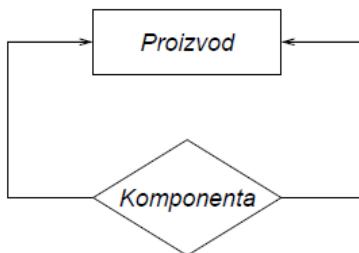
Nivo detaljnosti naziva tipova.

Tip poveznika reda 3 (n-arni tip poveznika)



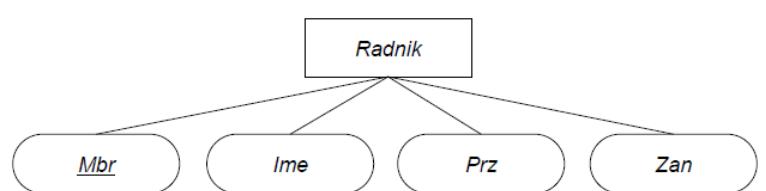
Nivo detaljnosti naziva tipova.

Rekurzivni, binarni tip poveznika.



Nivo detaljnosti obeležja (i domena)

Skup obeležja jednog tipa entiteta



## Integritetna komponenta

Tipovi ograničenja u ER modelu podataka

- ograničenje domena
- ograničenje pojave tipa
- kardinalitet tipa poveznika
- ograničenje ključa (integritet tipa) (za tip entiteta i tip poveznika)

### Ograničenje domena

Specifikacija domena je struktura:  $D(id(D), Predef)$

- $D$  - naziv domena
- $id(D)$  - ograničenje domena
- $Predef$  - predefinisana vrednost domena

**Ograničenje domena  $id(D)$**  definiše se primenom izabranog pravila za specificiranje korisnički definisanog domena. Postoje sledeća pravila:

- pravila nasleđivanja
- pravila tipa sloga
- pravila tipa skupa (kolekcije)
- pravila tipa izbora

U našem slučaju, definisaćemo ograničenje domena primenom pravila nasleđivanja.

#### Pravilo nasleđivanja i $id(D)$

Ograničenje "nasleđenog" domena je struktura:  $id(D) = (Tip, Dužina, Uslov)$

- $Tip$  - tip podatka (oznaka primitivnog domena ili oznaka prethodnog, korisnički definisanog domena).  $Tip$  je jedina obavezna komponenta specifikacije. Nasleđuju se sva ograničenja, relacije i operacije, definisane nad izabranim tipom
- $Dužina$  - dužina tipa podatka. Dužina se navodi samo za tipove podataka (primitivne domene) koji to zahtevaju
- $Uslov$  - logički uslov koji svaka vrednost domena mora da zadovolji. Uslov u (ER) modelu podataka mora biti definisana sintaksa za zadavanje logičkih uslova

$Predef$  mora da zadovolji ograničenja *tipa, dužine i uslova*.

Interpretacija integriteta domena moguća je za bilo koju vrednost - konstantu.

Primeri:

- $DPREZIME((String, 30, \Delta), \Delta)$
- $DDATUM((Date, \Delta, d \geq '01.01.1900'), \Delta)$
- $DOCENA((Number, 2, d \geq 5 \wedge d \leq 10), \Delta)$
- $DPOZOCENA((DOCENA, \Delta, d \geq 6), 6)$

### Nula vrednost

**Nula (nedostajuća) vrednost** je specijalna vrednost obeležja, tj. specijalno ograničenje domena. Označava se simbolom  $\omega$ . (u praksi, to je oznaka NULL)

Formalna interpretacija nula vrednosti - "vrednost obeležja nedostaje – nije zadata".

Moguća značenja nula vrednosti:

- nepoznata - postojeća vrednost obeležja
- nepostojeća vrednost obeležja
- neinformativna vrednost obeležja

Nekada se javlja potreba da obeležje, umesto vrednosti iz domena, poprimi vrednost  $\omega$ .

## Ograničenje vrednosti obeležja

Specifikacija obeležja  $A \in Q$ , datog tipa  $N$  je struktura:  $(id(N, A), Predef)$

- $id(N, A)$  - ograničenje vrednosti obeležja
- $Predef$  - predefinisana vrednost obeležja

**Ograničenje vrednosti obeležja  $id(N, A)$**  definiše se za svako obeležje tipa i predstavlja strukturu:

$$id(N, A) = (Domen, Null)$$

- $Domen$  - oznaka (naziv) pridruženog domena obeležja
- $Null \in \{T, \perp\}$ 
  - $T$  - dozvola dodele nula vrednosti obeležju unutar  $N$
  - $\perp$  - zabrana dodele nula vrednosti obeležju unutar  $N$

$Domen$  i  $Null$  su obavezne komponente specifikacije

$Predef$  - ako se navede, onda je on važeći. U protivnom, važeći je  $Predef$  odgovarajućeg  $Domena$ , ili prvog sledećeg nasleđenog domena, za koji je  $Predef$  definisan

Interpretacija ograničenja moguća je za bilo koju vrednost obeležja.

## Ograničenje pojave tipa

**Ograničenje pojave tipa** definiše ograničenja na moguće vrednosti podataka unutar iste pojave tipa entiteta ili tipa poveznika. Predstavlja skup ograničenja vrednosti obeležja, kojem je pridodat logički uslov.

Formalno, ograničenje pojave tipa  $N$  predstavlja strukturu:  $id(N) = (\{id(N, A) \mid A \in Q'\}, Uslov)$

- $Q'$  - prošireni skup obeležja tipa
  - za tip entiteta je  $Q' = Q$
  - za tip poveznika je  $Q' = Q \cup K_p$ , gde je  $K_p$  skup obeležja primarnog ključa tipa poveznika.
- $Uslov$  - logički uslov koji svaka pojava tipa mora da zadovolji. Može, u ulozi operanda, da sadrži bilo koje obeležje proširenog skupa obeležja datog tipa. U (ER) modelu podataka mora biti definisana sintaksa za zadavanje logičkih uslova

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

Primer:

$Radnik(\{MBR, PRZ, IME, ZAN, BPJZ\}, \{MBR\})$

Radnik	Domen	Null	Predef
MBR	DMBR	$\perp$	$\Delta$
PRZ	DPRZ	$\perp$	$\Delta$
IME	DIME	$\perp$	$\Delta$
ZAN	DZAN	$\perp$	$\Delta$
BPJZ	DBPJZ	T	$\Delta$
Uslov:	$ZAN = 'prg' \Leftrightarrow BPJZ \leftrightarrow \omega$		

Domen	Tip	Dužina	Uslov	Predef
DMBR	Number	4	$d \geq 1$	$\Delta$
DPRZ	String	30	$\Delta$	$\Delta$
DIME	String	15	$\Delta$	$\Delta$
DZAN	String	3	$\Delta$	$\Delta$
DBPJZ	Number	2	$d \geq 0$	0

## Kardinalitet tipa poveznika

**Kardinalitet tipa poveznika** prema povezanom tipu predstavlja par  $(a, b)$ , gde:

- $a \in \{0,1\}$  - minimalni kardinalitet
- $b \in \{1, N\}, N \geq 2$  - maksimalni kardinalitet

On ograničava u koliko pojava tipa poveznika može učestvovati jedna, bilo koja pojava povezanog tipa, minimalno ( $a$ ) i maksimalno ( $b$ ). Definiše se za svaki povezani tip.

Primer:



Kardinaliteti prikazanog tipa poveznika formalizuju ograničenja

- $(1, 1)$  - jedan radnik mora biti raspoređen na tačno jedno radno mesto
- $(0, N)$  - na jedno radno mesto može biti raspoređeno više radnika, ali ne mora ni jedan

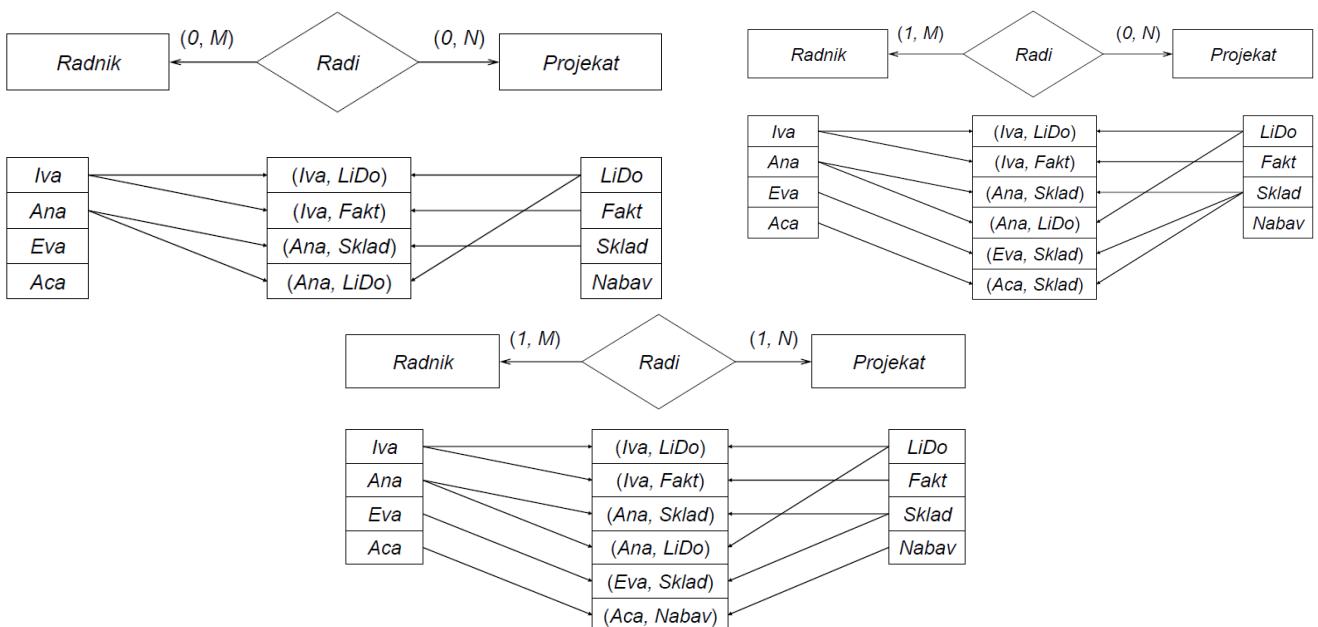
Tri opšte grupe maksimalnih kardinaliteta:

- $M : N$
- $N : 1$
- $1 : 1$

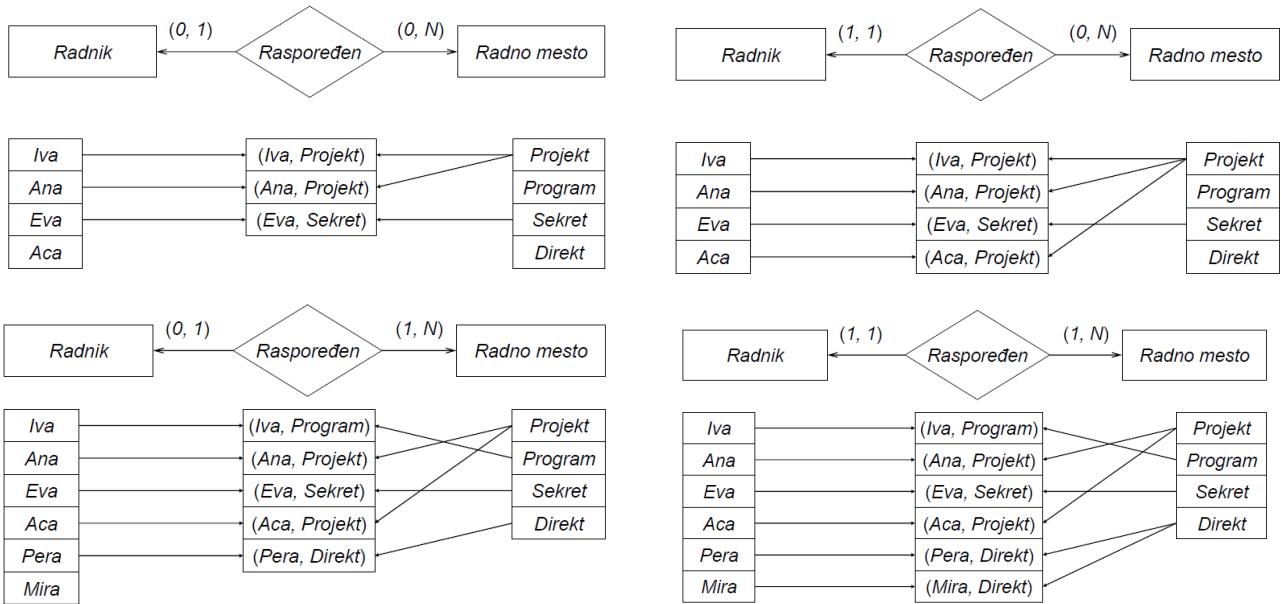
Ove grupe imaju uticaj na formiranje ključeva tipa poveznika.

Primeri pravila definisanja i pisanja kardinaliteta na dijagramima (binarni tipovi poveznika):

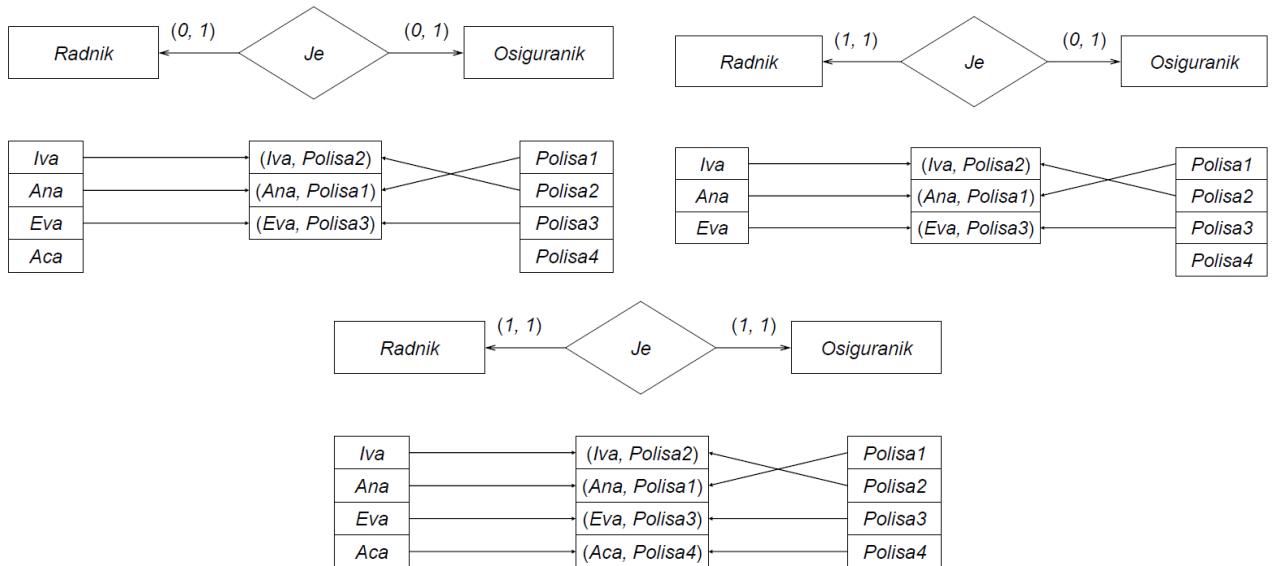
- Grupa M:N (više prema više)



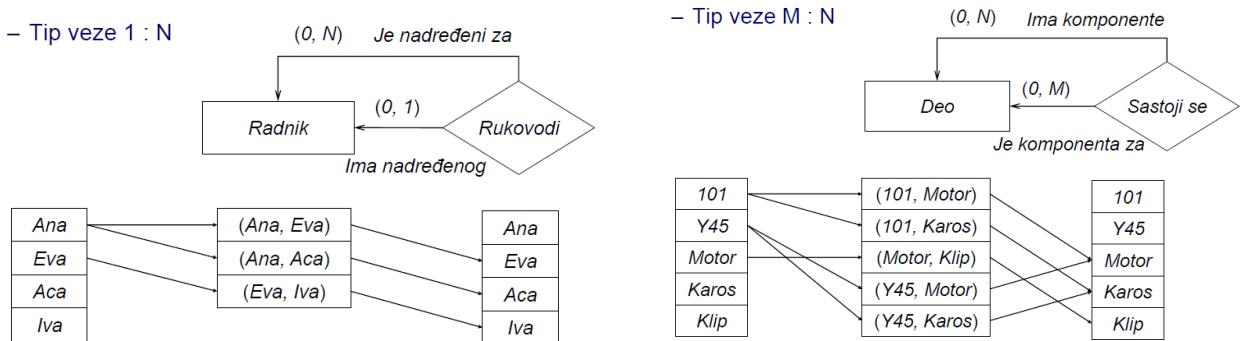
- Grupa N:1 (više prema jedan)



- Grupa 1:1 (jedan prema jedan)



Rekursivni tip poveznika:



Tip veze 1:N definije strukturu tipa stabla, a tip veze M:N mrežnu strukturu.

Kod rekurzivnih tipova poveznika ne treba očekivati 1 na mestu minimalnog kardinaliteta, jer bismo tada imali cikličnu strukturu.

## Integritet tipa poveznika

Integritet tipa entiteta - ograničenje ključa.

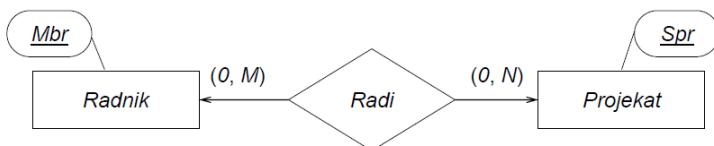
Integritet tipa poveznika

- niz naziva povezanih tipova, ili njegov neprazan podniz
- ograničenje ključa

Tri opšte grupe maksimalnih kardinaliteta ( $M : N$ ,  $N : 1$ ,  $1 : 1$ ) imaju uticaj na formiranje ključeva tipa poveznika.

Primeri formiranja ključeva tipova poveznika:

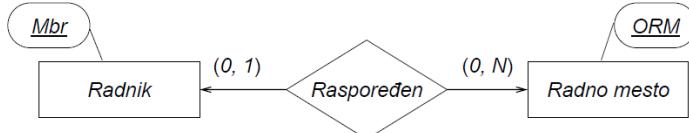
- Grupa M:N (više prema više)



Integritet tipa poveznika (identifikator tipa poveznika) *Radi*:

- (*Radnik, Projekat*)
- $K_p = Mbr + Spr$

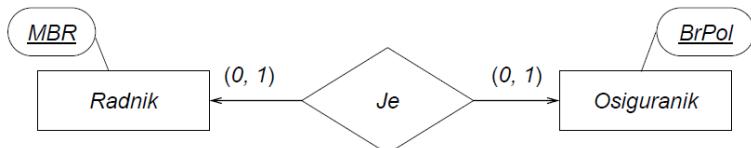
- Grupa N:1 (više prema jedan)



Integritet tipa poveznika (identifikator tipa poveznika) *Raspoređen*:

- (*Radnik*)
- $K_p = Mbr$

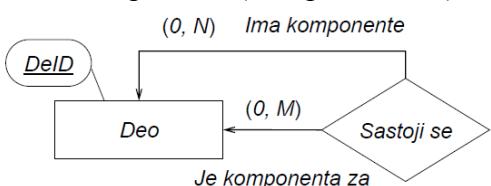
- Grupa 1:1 (jedan prema jedan)



Integritet tipa poveznika (identifikator tipa poveznika) *Je*:

- (*Radnik*) i (*Osiguranik*)
- $K_p = Mbr$  i  $K_2 = BrPol$

- Grupa M:N (više prema više) i rekursivni tip poveznika



Integritet tipa poveznika (identifikator tipa poveznika) *Sastoji se*:

- (*Deo, Deo*), tj. (*Deo(Ima komponente), Deo(Je komponenta za)*)
- $K_p = DelD + DelDkom$

*DeIDkom* – preimenovano obeležje *DeID*

Semantika: *DeID* sa ulogom komponente ugradnje

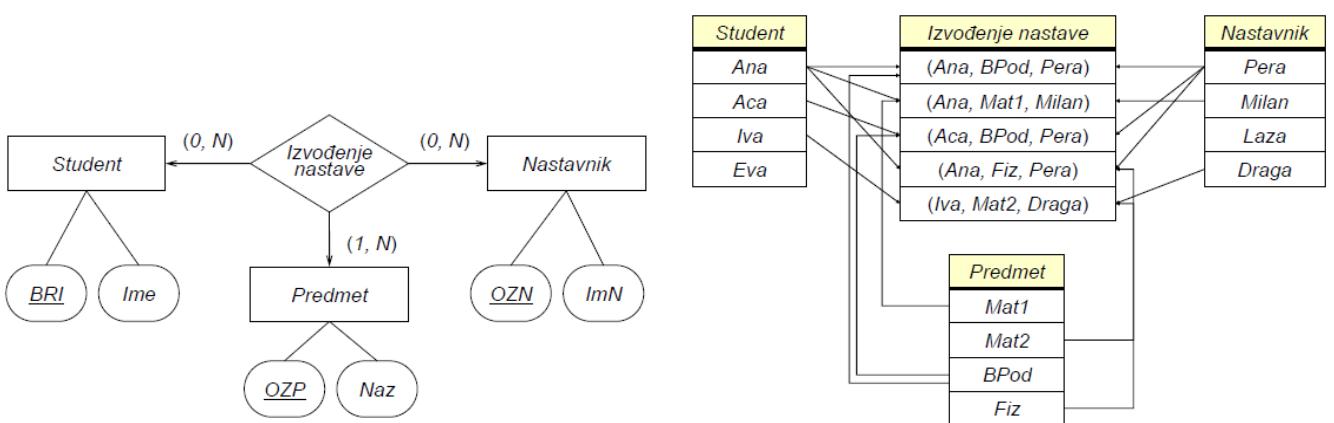
## N-arni tip poveznika ( $n > 2$ )

Tip poveznika može da povezuje više od dva druga tipa.

Kod  $n$ -arnog tipa poveznika (tipa poveznika reda  $n > 2$ ) takođe je potrebno odrediti kardinalitete 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 tipa poveznika.

Primer:

- Tipovi entiteta: *Student*, *Nastavnik*, *Predmet*
- Ograničenja:
  - jedan nastavnik može predavati **više predmeta za više studenata**
  - jedan student može slušati **više predmeta kod više nastavnika**
  - jedan predmet može predavati **više nastavnika za više studenata**
  - postoje nastavnici, koji ne predaju ni jedan predmet bilo kom studentu
  - postoje studenti koji ne slušaju ni jedan predmet kod bilo kog nastavnika
  - **ne postoje predmeti** koje ne predaje ni jedan nastavnik ni jednom studentu



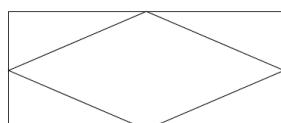
## Gerund i agregacija

### Gerund

**Gerund** predstavlja tip entiteta dobijen transformacijom tipa poveznika, tj. tip poveznika, koji predstavlja povezani tip u nekom drugom tipu poveznika. Gerund ima dvojaku ulogu, istovremeno je i tip entiteta i tip poveznika. Tip poveznika je za neke druge, povezane tipove, a tip entiteta u nekim drugim tipovima poveznika.

Dat je tip poveznika  $N(N_1, N_2, \dots, N_m, Q, C)$ . Neka je neki  $N_i$  takođe tip poveznika.  $N_i$  predstavlja gerund, ponaša se kao tip entiteta u odnosu na  $N$ .

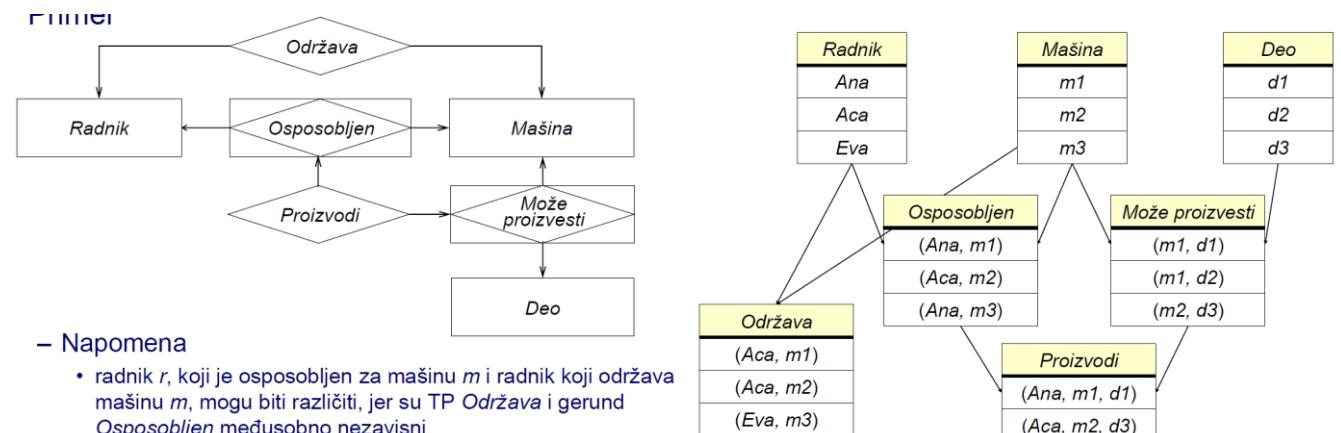
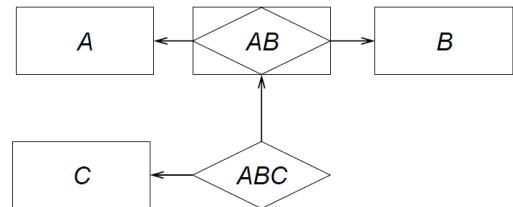
Geometrijska predstava gerunda u ER dijagramima:



Gerund se upotrebljava kada ne mogu proizvoljne kombinacije pojave nekih tipova biti sadržane u pojavi posmatranog tipa poveznika i postoji pravilo koje kombinacije pojave tih tipova mogu biti sadržane u pojavi posmatranog tipa poveznika.

Primeri:

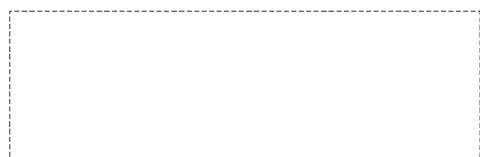
- entiteti klase  $A$ ,  $B$  i  $C$  su u međusobnim vezama tipa  $(a, b, c)$ 
  - uvodi se tip poveznika  $ABC$ , između  $A$ ,  $B$  i  $C$
- ne mogu svi  $(a, b)$  parovi entiteta iz  $A$  i  $B$  učestvovati u vezama  $(a, b, c)$ , nad tipom  $ABC$
- postoji pravilo koji  $(a, b)$  parovi iz  $A$  i  $B$  mogu učestvovati u vezama  $(a, b, c)$ , nad tipom  $ABC$ 
  - uvodi se tip poveznika – gerund  $AB$
  - tip poveznika  $ABC$  povezuje  $AB$  i  $C$
  - pojave tipa poveznika  $ABC$  zavise od egzistencije pojave tipa poveznika  $AB$
  - Klase entiteta
    - Radnik, Mašina i Deo*
  - Odnosi:
    - radnik  $r$  je osposobljen za rad na mašini  $m$
    - na mašini  $m$  se može proizvesti deo  $d$
    - radnik  $r$ , na nekim od onih mašina  $m$ , za koje je osposobljen, izrađuje neke od onih delova  $d$ , koji se na mašini  $m$  mogu proizvesti
    - radnik  $r$  održava mašinu  $m$
    - radnici na održavanju mogu, a ne moraju da rade na proizvodnji delova



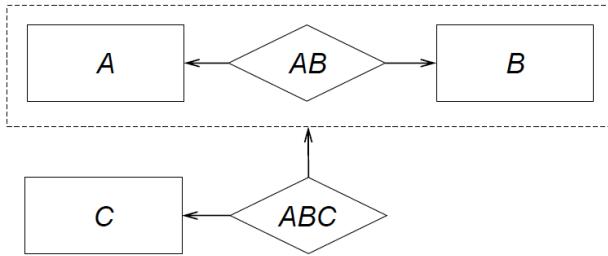
## Aggregacija

Aggregacija obezbeđuje objedinjavanje složenijih ER struktura. Cela ER struktura se posmatra kao jedan tip entiteta. Predstavlja povezani tip za neki tip poveznika. Može predstavljati kojsnički pogled na bazu podataka ("virtuelni" tip entiteta). Najjednostavniji primer agregacije je gerund.

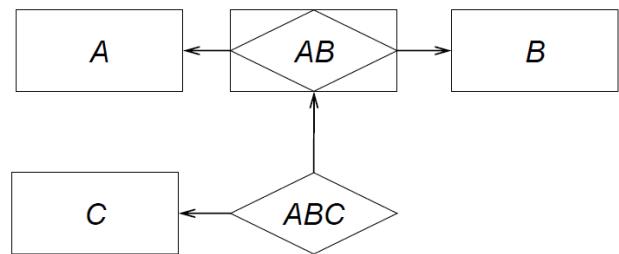
Geometrijska predstava agregacije u ER dijagramima:



Primer:



Alternativni dijagram u ovom primeru:



## Id-zavisnost, IS-A hijerarhija i kategorizacija

### Slabi tip entiteta

**Slabi tip entiteta** je tip entiteta čije su pojave zavisne od pojava nekog drugog tipa entiteta.

Vrste zavisnosti slabih tipova entiteta:

- egzistencijalna
- identifikaciona

### Egzistencijalna zavisnost

**Egzistencijalna zavisnost** između pojava dva tipa entiteta postoji kada je minimalni kardinalitet tipa poveznika jednak 1. ( $a = 1$ )

Regуларни tip entiteta je tip entiteta koji nije u egzistencijalnoj zavisnosti.

Primer:



- Regularni TE: *Radno\_mesto*
- Slabi TE: *Radnik*
  - egzistencijalno zavisn od TE *Radno\_mesto*
    - Ako se ukine radno mesto, radnik gubi posao
    - *Radnik* - egzistencijalno zavisni TE

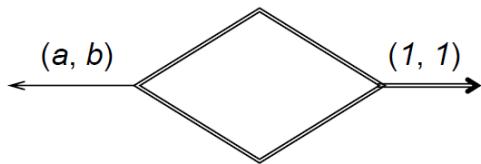
### Identifikaciona zavisnost

**Identifikaciona zavisnost** slabog tipa entiteta je poseban slučaj egzistencijalne zavisnosti. Postoji ako i samo ako su i minimalni i maksimalni kardinalitet tipa poveznika prema slabom tipu entiteta jednaki 1. ( $(a,b) = (1,1)$ )

U semantičkom smislu, identifikaciona zavisnost predstavlja poseban koncept u ER modelu podataka, koji uvodi klasifikaciju tipova poveznika na neidentifikacione i identifikacione.

Identifikacioni tip poveznika reprezentuje identifikacionu zavisnost slabog tipa entiteta. Ukazuje da se svaka pojava zavisnog tipa entiteta može identifikovati samo uz pomoć identifikatora nadređenog tipa entiteta. Identifikator (ključ) zavisnog tipa entiteta se formira korišćenjem identifikatora (ključa) nadređenog tipa entiteta.

Geometrijska predstava u ER dijagramima:

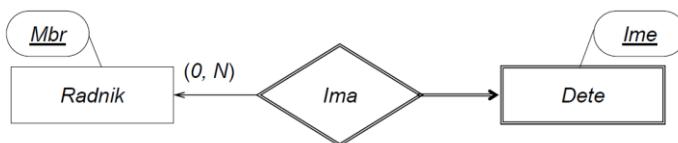


Opcionalno, id-zavisni tip entiteta se može predstaviti oblikom:



Navođenje kardinaliteta (1,1) nije obavezno, podrazumeva se i često se izostavlja.

Primer:



- *Ima* - identifikacioni TP
- *Dete* - identifikaciono zavisni TE
- *Radnik* - nadređeni (regularni) TE

Identifikaciono zavisni tip entiteta može posedovati neprazan skup sopstvenih identifikacionih obeležja (primer za tip entiteta *Dete*: *Ime*). Bilo koja pojava id-zavisnog tipa entiteta se može identifikovati isključivo navođenjem vrednosti njegovih identifikacionih obeležja i vrednosti identifikatora (ključa) nadređenog tipa entiteta.

Identifikator id-zavisnog tipa entiteta  $N_i$  :  $(N, X)$

- $N$  - naziv nadređenog tipa entiteta
- $X$  - skup identifikatorskih obeležja tipa entiteta  $N_i$

Ključ id-zavisnog tipa entiteta  $N_i$  :  $K_i = K \cup X$

- $K$  - ključ nadređenog tipa entiteta

Primer:

- Identifikator id-zavisnog TE *Dete*  
*(Radnik, {Ime})*
- Ključ id-zavisnog TE *Dete*  
 $K_i = Mbr + Ime$
- Napomene
  - regularni TE može učestvovati kao id-zavisani povezani tip u nekom drugom TP
  - id-zavisni TE može učestvovati i kao id-zavisani i kao regularan u više različitih TP

## IS-A hijerarhija

Tip poveznika **IS-A hijerarhija** predstavlja poseban koncept - tip poveznika u EER modelu. Zahteva uvođenje superklase i potklase.

**Superklasa** (nadtip) i **potklasa** (podtip) predstavljaju posebne vrste tipova, tj. pojmove vezani za postupak *specijalizacije*, odnosno *generalizacije*, svojstvene semantičkim modelima podataka.

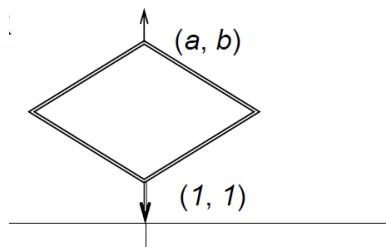
**Specijalizacija** se primenjuje kada neki skup entiteta ili poveznika (superklasa) poseduje prepoznatljive podskupove (potklase) sa samo sebi svojstvenim obeležjima ili samo sebi svojstvenim vezama sa drugim klasama entiteta ili poveznika.

- Date su klase:
  - $E_1 = \{e_i \mid P_1(e_i)\}$
  - $E_2 = \{e_i \mid P_2(e_i)\}$
- Uočava se implikacija:
$$P_2(e_i) \Rightarrow P_1(e_i)$$
- Tada važi:
$$E_2 \subseteq E_1$$
  - $E_1$  se naziva superklasom (nadtipom)
  - $E_2$  se naziva potklasom (podtipom)

Pojmovi superklase i potklase se uvode da bi model statičke strukture realnog sistema bio semantički bogatiji, da bi se izbegle nula vrednosti u ekstenziji i da bi se izbeglo definisanje tipa poveznika, koji nema mnogo smisla.

Specijalizacija se vrši na osnovu vrednosti nekog skupa klasifikacionih obeležja. U tipu entiteta superklase ostaju sva zajednička obeležja i primarni ključ. U tipove entiteta potklase distribuiraju se samo svojstvena, specifična obeležja.

Geometrijska predstava tipa poveznika IS-A hijerarhija u ER dijagramima:



Opcionalno, tip entiteta potklasa se može predstaviti oblikom:



Navođenje kardinaliteta  $(a, b)$  je obavezno i ono određuje tip IS-A hijerarhije. Kardinaliteti  $(1, 1)$  prema potklasama se mogu izostaviti.

Tip IS-A hijerarhije definiše se kardinalitetima tipa poveznika IS-A hijerarhija na strani superklase.

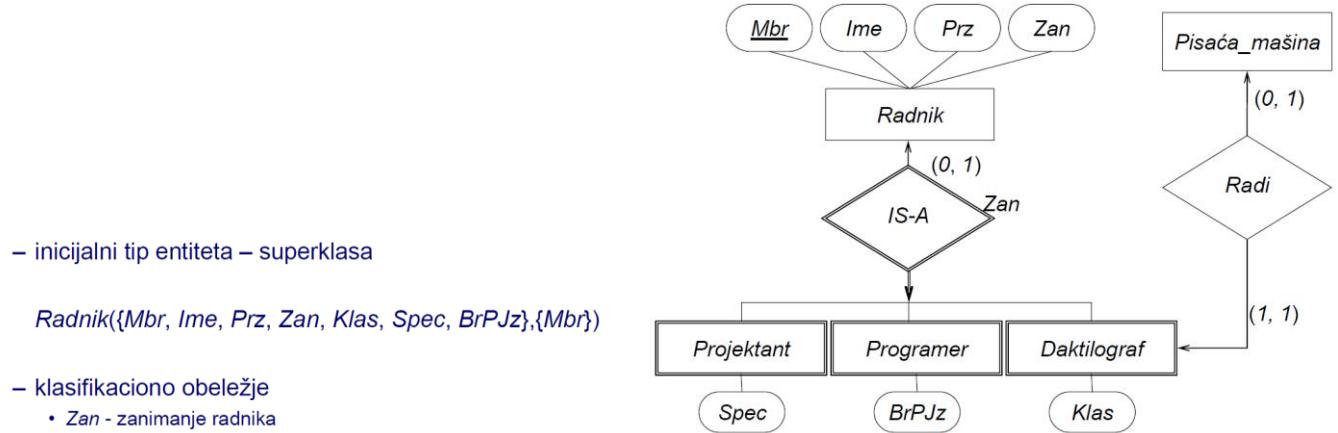
Ako je minimalni kardinalitet  $a$  jednak

- 1 - **totalna** IS-A hijerahija
- 0 - **parcijalna** IS-A hijerarhija

Ako je maksimalni kardinalitet  $b$  jednak

- 1 - **nepresečna** IS-A hijerahija
- $N$  - **presečna** IS-A hijerarhija

Primer:



Bitne karakteristike IS-A hijerahije:

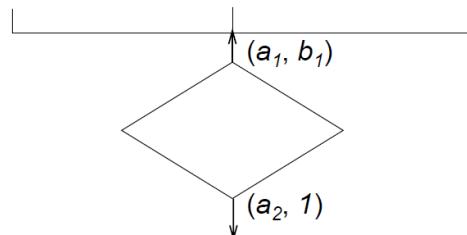
- Nasleđivanje osobina superklase
- Ključ (identifikator) svake potklase je primarni ključ (identifikator) superklase – nasleđivanje ključeva (pojave potklase se identifikuju putem vrednosti primarnog ključa odgovarajuće pojave superklase)
- Potklase mogu imati svoje sopstvene ključeve
- Identifikaciona zavisnost svake potklase prema superklasi
- Potklasa može imati ulogu superklase u drugoj IS-A hijerarhiji
- Nad jednim tipom može se napraviti više različitih IS-A hijerarhija, koristeći različite kriterijume

## Kategorizacija

Tip poveznika **kategorizacija** predstavlja poseban koncept - tip poveznika u EER modelu. To je pojam vezan za postupak klasifikacije (tipizacije), svojstvene semantičkim modelima podataka. Zahteva uvođenje pojma *kategorije*.

**Kategorija** predstavlja posebnu vrstu tipa (tip entiteta ili tip poveznika - gerunda). Jedan tip entiteta se povezuje s više kategorija (barem dve). Svaka pojava posmatranog tipa entiteta pripada najviše jednoj kategoriji ("ekskluzivni tip poveznika" prema kategorijama). Ne postoji id-zavisnost posmatranog tipa entiteta od kategorija ili obratno (posmatrani tip entiteta i kategorije su međusobno nezavisni (regularni) tipovi). Može a ne mora postojati skup klasifikacionih obeležja kategorije.

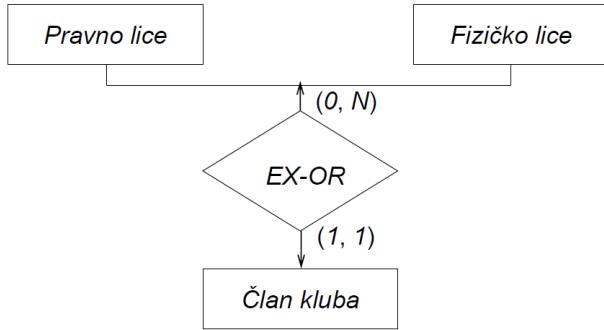
Geometrijska predstava tipa poveznika kategorizacije u ER dijagramima:



Navođenje kardinaliteta  $(a_2, 1)$  je obavezno.  $a_2$  definiše tip kategorizacije. Ako je  $a_2$  jednak:

- 0 - **parcijalna** kategorizacija
- 1 - **totalna** kategorizacija

Primer:



#### - Semantika

- član kluba mora biti ili pravno, ili fizičko lice
- pravno ili fizičko lice može ostvariti više, a ne mora ostvariti ni jedno članstvo u klubu

## Završne napomene

ER model podataka je izuzetno pogodan za rane korake projektovanja. Dijagramska tehnika je pogodna za komunikaciju sa korisnicima. Postoje heuristička pravila projektovanja konceptualne šeme baze podataka (na osnovu deskriptivnog opisa strukture i ograničenja u realnom sistemu). Ne postoje standardi dijagramske reprezentacije.

Neka heuristička pravila:

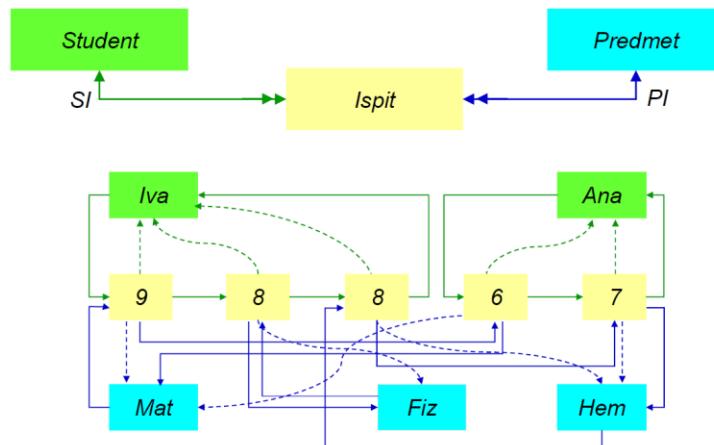
- Imenice ukazuju na potrebu uvođenja tipova entiteta
- Glagolski oblici ukazuju na potrebu uvođenja tipova poveznika ili gerunda
- Fraze oblika "bar jedan", "više", "najmanje jedan" i slične, ukazuju na kardinalitete tipova poveznika ili gerunda
- Postojanje različitih uloga entiteta jednog skupa u vezama sa entitetima drugih skupova, ukazuje na potrebu uvođenja više tipova poveznika između odgovarajućih tipova entiteta
- Preporučljivo je da se uloge entiteta u vezama eksplicitno navedu
- Veze između entiteta jednog skupa ukazuju na potrebu uvođenja rekursivnog tipa poveznika
- Kod rekursivnih veza je posebno važno da se uloge entiteta eksplicitno navedu
- Vremensko prethođenje entiteta jednog skupa u odnosu na entitete nekog drugog skupa, ukazuje na egzistencijalnu zavisnost entiteta drugog skupa od entiteta prvog skupa i potrebu uvođenja minimalnog kardinaliteta  $a = 1$
- Potreba takvog selektivnog povezivanja entiteta tri ili više skupova, kod kojeg u vezi mogu učestovati samo entiteti koji su već u nekoj drugoj vezi sa entitetima jednog ili više drugih skupova, ukazuje na neophodnost korišćenja gerunda
- Postojanje entiteta jednog skupa sa specifičnim osobinama ili vezama sa entitetima drugih skupova, ukazuje na potrebu uvođenja IS-A hijerarhije
- Svako obeležje može pripadati samo jednom tipu entiteta, ili samo jednom tipu poveznika
- Nasleđena obeležja ključa tipa poveznika se ne uključuju u sam skup obeležja tipa poveznika
- Tip entiteta ili tip poveznika sadrži samo ona obeležja realnog skupa entiteta, ili realnog skupa poveznika, koja su bitna za realizaciju ciljeva postavljenih pred informacioni sistem

# Koncepcija relacionog modela podataka

## Mrežni i hijerarhijski model podataka

Mrežni i hijerarhijski model podataka su u korišćenju od 1970-ih godina. Njihovi nedostaci su:

- čvrsta povezanosti programa i fizičke strukture podataka
- strukturalna kompleksnost
- proceduralno orijentisani jezici (navigacioni jezik) za manipulaciju podacima
- "ad hoc" razvijeni modeli - bez značajnije upotrebe matematičkih formalizama



## Relacioni model podataka

Relacioni model podataka (RMP) je u razvoju od 1970-ih godina. Motiv razvoja je:

- otklanjanje nedostataka klasičnih modela podataka (gore nabrojani)
- insistiranje na matematičkim osnovama u razvoju modela podataka - matematička osnova RMP je teorija skupova i relacija, kao i njeno oslanjanje na predikatski račun prvog reda.

Relacionim modelom podataka postignuta je:

- nezavisnost programa od podataka, tj. fizičke strukture podataka (najvažniji cilj)
- strukturalna jednostavnost
- deklarativni jezik

## Nezavisnost programa od podataka

Kod ranijih modela podataka fizički aspekti baze podataka bili su ugrađeni u programe.

Pojam fizičke strukture podataka odnosi se na sve aspekte takozvane "interne reprezentacije podataka", kao i na mehanizme pristupa podacima. Spomenuti mehanizmi su: raspodela slogova po zonama baze podataka, fizički redosled i grupisanje slogova (po stranicama), algoritam transformacije ključa u adresu, lanci slogova povezanih pokazivačima, indeksi (stabla traženja), hijerarhijski redosled slogova (u setovima) i slično.

Rešenje predstavlja **nezavisnost programa od podataka**. Ona se postiže potpunim odvajanjem prezentacionog formata od formata memorisanja. Time se eliminiše potreba ugradnje bilo kakve informacije o fizičkoj strukturi podataka u programe.

U relacionom modelu, kao najveća organizaciona jedinica podataka, odabrana je  $n$ -arna relacija. S obzirom na matematičku definiciju, relacija je skup  $n$ -torki (torki), pri čemu svaka komponenta torke predstavlja vrednost iz jednog skupa, takozvanog "domena obeležja". Broj  $n$  je broj obeležja u opisu relacije na nivou apstrakcije obeležja.

Ovaj apstraktni opis relacije se naziva šema relacije:

$$N(R,C)$$

- $R$  - skup obeležja
- $C$  - skup ograničenja, pri čemu je  $K \subseteq C$  obavezno zadat skup ključeva, koji je neprazan.

Često se u početnim fazama projektovanja šema relacije posmatra kako struktura  $N(R,K)$ .

Primer:

- Fakultet ( $\{SFK, NAZ, BIP\}$ ,  $\{SFK\}$ )
- $r(Fakultet) = \{ (PMF, Matematički, 7), (EKF, Ekonomski, 4), (ETF, Elektrotehnički, 9), (MAF, Mašinski, 7) \}$
- Upis nove torke  
 $(EKF, Elektronski, 8)$ ,
- narušio bi ograničenje ključa (uslov integriteta)

## Strukturalna jednostavnost

Koncept relacije je osnova reprezentacije logičkih struktura podataka u relacionom modelu podataka. Ne sadrži nikakve informacije o fizičkoj organizaciji podataka.

Da bi se obezbedila **strukturalna jednostavnost** modela, kao reprezent relacije usvojena je dovodimenzionalna tabela, jer predstavlja lako razumljiv pojam i za programera i za krajnjeg korisnika. Prilikom korišćenja tabele kao reprezenta relacije, tabela nosi naziv relacije, a vrsta sa nazivima kolona (zaglavlje tabele) predstavlja skup obeležja šeme relacije. Nazivi kolona tabele predstavljaju obeležja, a same kolone sadrže elemente domena odgovarajućih obeležja. Vrste tabele predstavljaju  $n$ -torke.

Primer:

Fakultet			Projektant			
SFK	NAZ	BIP	MBR	IME	PRZ	SFK
FIL	Filozofski	1	M3	Iva	Ban	PMF
PMF	Matematički	7	M1	Ana	Tot	MAF
ETF	Elektrotehnički	9	M4	Ana	Ras	FIL
EKF	Ekonomski	4	M8	Aca	Pap	ETF
MAF	Mašinski	7	M6	Iva	Ban	EKF
			M5	Eva	Tot	ETF

Selekcija podataka u operacijama nad bazom podataka kod relacionog modela podataka postiže se asocijativnim adresiranjem. Isključiva je upotreba simboličkih adresa, vrlo često vrednosti ključa. Svaki podataka u bazi podataka pronalazi se na osnovu naziva relacije, zadatih obeležja i vrednosti ključa. Skup  $n$ -torki (torki) sa zajedničkom osobinom selektira se na uniforman način - zadavanjem istog logičkog ulsova. SUBP vodi računa o transformaciji simboličke u relativnu adresu.

Povezivanje podataka kod relacionog modela podataka vrši se upotrebom simboličkih adresa - prenetih vrednosti ključa. Postoji dva slučaja dobijanja rešenja:

- rešenje putem prostiranja ključa (uvodenje pojma *stranog ključa* i ograničenja *referencijalnog integriteta*)
- rešenje putem kreiranja posebne tabele sa prostiranjem ključeva

U oba slučaja, transakcioni program ne vodi računa o pretvaranju simboličke u relativnu adresu.

## Deklarativni jezik

**Deklarativni jezik** temelji se na primjenjenim tehnikama povezivanja podataka sa prostiranjem ključa. Za potrebe razvoja deklarativnog jezika, definisana su dva alata:

- **relaciona algebra** - pošto su relacije skupovi, definisan je, u okviru relacione algebre i niz skupovnih operatora za manipulisanje podacima (unija, presek, razlika). Definisani su i specijalizovani skupovni operatori (spoj (join), projekcija, selekcija).
- **relacioni račun** - relacioni račun nad torkama i relacioni račun nad domenima definisani su na osnovama predikatskog računa prvog reda.

**SQL** (*Structured Query Language*) je deklarativeni jezik, zasnovan na relacionom računu nad torkama. Radi sa skupovima podataka (torki). Osnovni oblik naredbe za upite SQL-a je:

```
SELECT    <lista obeležja>
FROM      <lista relacija>
WHERE    <logički izraz>
```

Primeri:

- ```
– SELECT IME, PRZ, BIP
FROM Fakultet, Projektant
WHERE BIP > 5 AND
      Fakultet.SFK = Projektant.SFK
```
- ```
– SELECT IME, PRZ, BIP
FROM Fakultet NATURAL JOIN Projektant
WHERE BIP > 5
```

IME	PRZ	BIP
Iva	Ban	7
Ana	Tot	7
Aca	Pap	9
Eva	Tot	9

IME	PRZ	BIP
Iva	Ban	7
Ana	Tot	7
Aca	Pap	9
Eva	Tot	9

## 12 principa relacionih modela podataka

### 0. pravilo

Da bismo jedan sistem baza podataka videli kao relacioni sistem baza podataka, mora biti kvalifikovan kao relacioni, tj. mora da podržava koncepte relacionog modela podataka, mora biti organizovan tako da podržava pojам baze podataka i mora biti sposoban da upravlja bazom podataka.

### 1. Pravilo informativnosti

Sve informacije u bazi podataka moraju biti prezentovane na jedan i samo jedan način i to pomoću vrednosti upisanih u polja odgovarajućih kolona unutar redova u tabeli.

## 2. Pravilo garantovane dostupnosti

Svi podaci moraju biti dostupni bez ikakvih dvosmisenosti. Ovo pravilo za nas predstavlja fundamentalni zahtev za uvođenje pojma primarnog ključa, bilo koja individualna skalarna vrednost u bazi podataka mora biti logički adresabilna tako što ćemo specifirati ime odgovarajuće tabele, ime odgovarajuće kolone u tabeli i vrednost primarnog ključa koja ukazuje na odgovarajući red.

## 3. Sistematični tretman null vrednosti

SUBP mora da dozvoljava zadavanje (ostavljanje) null (prazne) vrednosti za svako moguće polje. Na taj način mi reprezentujemo informaciju nedostajuće informacije ili neprimenljive informacije, koja je na sistematičan način odvaja od svih regularnih vrednosti (0, prazan string). null vrednost su tolerantne za bilo koji domen. Ovakva vrsta reprezentacije nedostajućih vrednosti mora biti pokrivena u smislu upravljanja SUBP-om na sistematičan način. (funkcija nvl, tj. način da utvrdimo da li nešto jeste ili nije null vrednost)

## 4. Aktivni online rečnik podataka baziran na relacionom modelu

Svaki relacioni sistem za upravljanje bazom podataka mora da podržava online (možemo mu pristupati u radnom režimu SUBP-a), inline (ugrađen u strukturu samog SUBP-a), relational (mora biti organizovan po istim principima relacionog modela podataka kao i naša baza podataka) rečnik podataka, kojem je moguće pristupiti od strane autorizovanih korisnika pomoću regularnog upitnog jezika. To znači da korisnici moraju biti u mogućnosti da pristupe strukturi baze podataka putem kataloga korišćenjem istog upitnog jezika, koji i inače koriste za pristup bazi podataka.

## 5. The comprehensive data sublanguage rule

sistem mora da podržava najmanje jedan od relacionih jezika, koji:

- ima linearu sintaksu (čije su naredbe izražajne, moguće ih je kreirati pomoću dobro definisane sintakse, tj. bez dvosmislenosti, kao karakter stringove - svaka naredba je čitljiva u obliku stringa)
- može biti upotrebljavan i na interaktivan način i kroz aplikativne programe
- mora da podržava operacije za definiciju podataka (operacije koje dozvoljavaju kreiranje modifikovanje i uništavanje šeme baze podataka, uključujući i mogućnost defnisanja takozvanih pogleda (views)), operacije za manipulaciju podataka (ažuriranja kao i upite), komande za zaštitu od neovlašćenog pristupa (security) i komande koje omogućuju da implementiramo ograničenja nad našom bazom podataka (integrity constraints) i operacije koje će podržavati transakcionim režimom rada (transakcioni režim rada podrazumeva da se uvek operacije nad bazom podataka organizuju u transakcije, koje u celosti uspevaju ili se u celosti poništavaju)

## 6. Pravilo modifikovanja pogleda

Svi pogledi koji su teoretski podložni modifikaciji (ažuriranju) moraju biti sposobni da budu korišćeni za ažuriranje i od strane SUBP-a.

## 7. Visokonivovski insert, update i delete

Sistem mora da podržava set-at-a-time (u trenutku operacije - jedan korak obrade pokriva sve što je predmet modifikacije, operadni su relacije, a datotekama tuple-at-a-time, operandi su slogovi) insert, update i delete operatore. Ovo znači da svi podaci mogu biti predmet upita iz relacione baze podataka, u skupovima koji su kontruisani od podataka koji su viđeni kao više redova ili iz više tabela. (Rezultat našeg upita je skup torki koji je na osnovu skupova torki jedne ili više različitih tabela). Ove operacije moraju biti podržane za bilo kakav set podataka koji se može selektovati, umesto što bismo koristili uvek obradu jednog reda, u jednoj tabeli, u jednoj koloni.

## 8. Fizička nezavisnost podataka

Promena na fizičkom nivou (kako su podaci fizički sačuvani, u nizovima, povezanim listama,...) ne sme da zahteva bilo kakvu izmenu u aplikaciji koja je bazirana na strukturi.

## 9. Logička nezavisnost podataka

Promene na logičkom nivou (vezane za tabele, kolone, redove itd) ne smeju da zahtevaju bilo kakvu izmenu na aplikaciji kaja je bazirana na upotrebi te strukture. Logičku nezavisnost podataka je mnogo teže dostići nego fizičku nezavisnost podataka. (primer logičke zavisnosti - možeš da menjaš nešto u definiciji tabele, ali to ne bi smelo da utiče na funkcionisanje programa)

## 10. Integritetna nezavisnost

Ograničenja integriteta podataka (ograničenja baze podataka) moraju biti specificirana nezavisno od programa naših aplikacija i sačuvani u katalogu (rečniku podataka). Mora biti moguće da po potrebi menjamo takva ograničenja, što ne bi smelo da utiče ni na koji način na ispravan rad naše aplikacije.

## 11. Distributivna nezavisnost

Distribucija delova baze podataka na različite lokacije (na različite servere baze podataka u našoj arhitekturi sistema) ne sme da bude vidljiva korisnicima baze podataka. Sve postojeće aplikacije treba da mogu da nastave da rade potpuno uspešno:

- kada se distribuirana verzija DBMS-a prvi put uvodi
- kada se postojeća nedistribuirana BP u nekom trenutku pretvori u distribuiranu BP (Kada se delovi BP distribuiraju na različite servere BP (logički jedinstvena a fizički razmeštena na više različitih servera baze podataka koji zajedno čine arhitekturu jendog sistema)

## 12. The nonsubversion rule

Ukoliko bi relacioni sistem imao jezik koji podržava low-level način rada (slog po slog) onda takav način rada ne može upotrebljen ni na koji način da preskoči ili opstruira pravila ograničenja koja se moraju moći iskazati putem jezika višeg nivoa, koji dozvoljava rad sa skupovima torki.

# Osnove relacionog modela podataka

## Strukturalna komponenta I

### Primitivni koncepti u relacionom modelu podataka:

- obeležje (atribut) - reprezentuje osobinu (svojstvo) klase entiteta ili poveznika u realnom sistemu
- domen - specifikacija skupa mogućih vrednosti koje neka obeležja mogu da dobiju

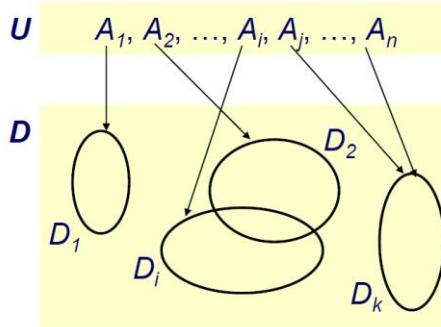
Polazna pretpostavka strukturalne komponente relacionog modela podataka, na kojoj se zasnivaju neke tehnike projektovanja relacione šeme baze podataka:

- o poznat je skup svih obeležja sistema (**univerzalni skup obeležja**)  $U = \{A_1, \dots, A_n\}$
- o poznat je skup svih domena sistema (**univerzalni skup domena**)  $D = \{D_1, \dots, D_k\}$

### Pravilo pridruživanja domena obeležjima:

Svakom obeležju obavezno se pridružuje tačno jedan domen.

$$\text{Dom: } U \rightarrow D, (\forall A_i \in U)(\text{Dom}(A_i) \in D)$$



Primer:

$$U = \{MBR, IME, POL, SPR, NAP\}$$

$$D = \{DIDS, DIME, DPOL, DNAP\}$$

#### – opis semantike uvedenih obeležja

- *MBR* - matični broj radnika
- *IME* - ime radnika
- *POL* - pol
- *SPR* - šifra projekta
- *NAP* - naziv projekta

#### – opis semantike uvedenih domena

- *DIDS* – domen za identifikacione brojeve {1, 2, ..., 100000}
- *DIME* – domen za imena radnika {Ana, Aca, Iva, ...}
- *DPOL* – domen za pol osobe {m, ž}
- *DNAP* – domena za nazive projekata {stringovi do dužine 30}

#### – pridruživanje domena obeležjima

- $\text{Dom}(MBR) = DIDS, \text{dom}(MBR) = \{1, 2, \dots, 100000\}$
- $\text{Dom}(IME) = DIME, \text{dom}(IME) = \{\text{Ana}, \text{Aca}, \text{Iva}, \dots\}$
- $\text{Dom}(POL) = DPOL, \text{dom}(POL) = \{m, \check{z}\}$
- $\text{Dom}(SPR) = DIDS, \text{dom}(SPR) = \{1, 2, \dots, 100000\}$
- $\text{Dom}(NAP) = DNAP, \text{dom}(NAP) = \{\text{stringovi do dužine 30}\}$

Konvencije u označavanju:

- skup obeležja  $X = \{A, B, C\}$  skraćeno se zapisuje u formi  $X=ABC$  ili  $X=A+B+C$
- izraz , gde su  $X$  i  $Y$  skupovi obeležja, skraćeno se zapisuje kao  $XY$

Primitivni koncepti nivoa intenzije:

- domen
- obeležje

Primitivni koncepti nivoa ekstenzije:

- vrednost

Kreiranje svih ostalih (složenih) koncepata strukturalne komponente relacionog modela podataka vrši se kombinovanjem (strukturiranjem) primitivnih koncepata i korišćenjem definisanih pravila u relacionom modelu podataka.

Skup svih primitivnih i složenih koncepata relacionog modela podataka za opis LSO (nivo intenzije) i LSP (nivo ekstenzije):

Nivo intenzije	Nivo ekstenzije
• Domen	• Vrednost
• Obeležje	• Podatak
• Skup obeležja	• Torka (N-torka)
• Šema relacije	• Relacija
• Šema BP	• Baza podataka

## Torka

**Torka** reprezentuje jednu pojavu entiteta ili poveznika. Pomoću torke se svakom obeležju, iz nekog skupa obeležja, dodeljuje konkretna vrednost iz skupa mogućih vrednosti definisanog domenom.

Formalno, za  $U = \{A_1, \dots, A_n\}$  i  $DOM = \cup_{i=1}^n (dom(A_i))$  (skup svih mogućih vrednosti), torka predstavlja preslikavanje:

$$t : U \rightarrow DOM,$$

$$(\forall A_i \in U)(t(A_i) \in dom(A_i))$$

Primer:

– Torka  $t_1$  može se prikazati kao skup podataka

- $U = \{MBR, IME, POL, SPR, NAP\}$

$$t_1 = \{(MBR, 101), (IME, Ana), (POL, \check{z}), (SPR, 1100), (NAP, Univerzitetski IS)\}$$

- Torka  $t_1$  definisana je na sledeći način

$\begin{array}{ll} t_1(MBR) & = 101 \\ t_1(IME) & = Ana \\ t_1(SPR) & = 1100 \\ t_1(POL) & = \check{z} \\ t_1(NAP) & = Univerzitetski IS \end{array}$	$t_2 = \{(MBR, 210), (IME, Aca), (POL, m), (SPR, 0105), (NAP, Polaris)\}$
---	---

– Zadata je i torka  $t_2$

$$t_2 = \{(MBR, 210), (IME, Aca), (POL, m), (SPR, 0105), (NAP, Polaris)\}$$

**Restrikcija** ("skraćenje") torke  $t$  (oznaka:  $t[X]$ ) na skup obeležja  $X \subseteq U$  dobija se kada se svakom obeležju iz skupa X pridruži ona vrednost koju je imala polazna torka  $t$ .

Formalno:

- $X \subseteq U, t: U \rightarrow DOM,$
- $t[X]: X \rightarrow DOM$

$$(\forall A \in X)(t[X](A) = t(A))$$

Primer:

- $t_2 = \{(MBR, 210), (IME, Aca), (POL, m), (SPR, 0105), (NAP, Polaris)\}$

- Neka je  $X = MBR+IME$
- $t_2[X] = \{(MBR, 210), (IME, Aca)\}$

## Relacija

**Relacija** nad skupom obeležja  $U$  predstavlja konačna skup torki. Reprezentuje skup realnih entiteta ili poveznika.

Formalno:

$$r(\mathbf{U}) \subseteq \{t \mid t: \mathbf{U} \rightarrow \text{DOM}\}, |r| \in \mathbb{N}_0$$

$\underbrace{\hspace{10em}}$

Skup svih mogućih torki nad  
skupom obeležja  $\mathbf{U}$  - Tuple( $\mathbf{U}$ )

Primeri:

- $\mathbf{U} = \{\text{MBR}, \text{IME}, \text{POL}, \text{SPR}, \text{NAP}\}$
- $r_1(\mathbf{U}) = \{t_1, t_2\}$ 
  - $t_1 = \{(\text{MBR}, 101), (\text{IME}, \text{Ana}), (\text{POL}, \check{z}), (\text{SPR}, 1100), (\text{NAP}, \text{Univerzitetski IS})\}$
  - $t_2 = \{(\text{MBR}, 210), (\text{IME}, \text{Aca}), (\text{POL}, m), (\text{SPR}, 0105), (\text{NAP}, \text{Polaris})\}$
- $R = \{A, B, C\}, R \subseteq \mathbf{U}$ 
  - $\text{dom}(A) = \{a_1, a_2\}$
  - $\text{dom}(B) = \{b_1, b_2\}$
  - $\text{dom}(C) = \{c_1, c_2\}$
- $t_1 = \{(A, a_1), (B, b_1), (C, c_1)\}$
- $t_2 = \{(A, a_2), (B, b_2), (C, c_2)\}$
- $t_3 = \{(A, a_1), (B, b_1), (C, c_2)\}$
- $r(R) = \{t_1, t_2, t_3\}$

U relaciji se ne mogu pojaviti dve identične torke (to je onda ista torka, samo dva puta prikazana).

Uobičajena reprezentacija relacije je pomoću tabele. Relaciju predstavlja kompletan sadržaj tabele (kratko, tabela). Poredak obeležja (kolona tabele) ne utiče na informacije koje sa sobom nosi relacija (nebitan je).

Primeri:

Radnik	MBR	IME	POL	SPR	NAP
$t_1$	101	Ana	$\check{z}$	1100	Univerzitetski IS
$t_2$	210	Aca	$m$	0105	Polaris

$r(R)$	$A$	$B$	$C$
$t_1$	$a_1$	$b_1$	$c_1$
$t_2$	$a_2$	$b_2$	$c_2$
$t_3$	$a_1$	$b_1$	$c_2$

## Operacijska komponenta

Operacijsku komponentu relacionog modela podataka čine:

- jezik za manipulaciju podacima - podrazumeva operacije za ažuriranje relacija: dodavanje nove torke (Add), brisanje postojeće torke (Delete) i modifikacija podataka postojeće torke (Update).
- jezik za definiciju podataka - podrazumeva operacije za upravljanje šemom baze podataka: kreiranje, brisanje i modifikovanje delova šeme baze podataka.
- upitni jezik - podrazumeva operacije za izražavanje upita nad jednom relacijom ili skupom relacija (pružanje podataka na uvid korisniku). Upitni jezik sačinjavaju: operatori za izražavanje upita, pravila za formiranje operanada upita - izraza, pravila za primenu tih operatora.

Vrste teoretskih upitnih jezika u relacionom modelu podataka su:

- relaciona algebra - zasnovana na teoriji skupova i skupovnih operacija
- relacioni račun - relacioni računi nad torkama i nad domenima zasnovani su na predikatskom računu prvog reda

Osnovne skupovne operacije nad relacijama:

Primer:

### – Unija

$$r(R) \cup s(R) = \{t \mid t \in r \vee t \in s\}$$

$r$	A	B
	$a_1$	$b_1$
	$a_2$	$b_2$
	$a_3$	$b_3$

### – Presek

$$r(R) \cap s(R) = \{t \mid t \in r \wedge t \in s\}$$

$s$	A	B
	$a_1$	$b_1$
	$a_3$	$b_3$

### – Razlika

$$r(R) - s(R) = \{t \mid t \in r \wedge t \notin s\}$$

$r - s$	A	B
	$a_2$	$b_2$

$r \cup s$	A	B
	$a_1$	$b_1$
	$a_2$	$b_2$
	$a_3$	$b_3$

$r \cap s$	A	B
	$a_1$	$b_1$

$r - s$	A	B
	$a_2$	$b_2$

**Selekcija** torki iz relacije omogućava izbor (selektovanje) torki relacije po nekom kriterijumu.

$$\sigma_F(r(R)) = \{t \in r \mid F(t)\}$$

Logičkom formulom  $F$  izražava se kriterijum po kojem se torke relacije  $r$  selektuju. Biće selektovane samo one torke, za koje je formula tačna. Zahteva se formalno definisanje sintakse za zapisivanje selekcionih formula tipa  $F$ .

Primeri:

$$- \sigma_F(r(R)), F ::= PLT > 5000$$

- Upit

- prikazati radnike čija je plata veća od 4000 i rade na projektu sa šifrom 11
- $\sigma_{PLT > 4000 \wedge SPR = 11}(r)$

$r$	MBR	IME	POL	SPR	PLT
	101	Ana	ž	11	3400
	102	Aca	m	14	4200
	110	Ivo	m	11	7000
	111	Olja	ž	11	7200

$\sigma_F$

MBR	IME	POL	SPR	PLT
110	Ivo	m	11	7000
111	Olja	ž	11	7200

**Projekcija (resktrikcija)** relacije je izdvajanje vrednosti pojedinih kolona iz relacije, odnosno projektovanje relacije na podskup skupa obeležja  $X \subseteq R$ .

$$\pi_X(r(R)) = \{t[X] \mid t \in r(R)\}$$

- Primer

- $P$  - pilot
- $A$  - tip aviona
- $L$  - broj leta

$r$	P	A	L
	Aca	747	101
	Ivo	737	101
	Aca	747	102
	Ana	DC9	110

- Upit:
  - prikazati pilote i tipove aviona na kojima lete:
  - $\pi_{PA}(r(PAL))$

P	A
Aca	747
Ivo	737
Ana	DC9

Primer:

- Posmatra se relacija  $r$

$r$	MBR	IME	POL	SPR	PLT
	101	Ana	ž	11	3400
	102	Aca	m	14	4200
	110	Ivo	m	11	7000
	111	Olja	ž	11	7200

### Upit

- prikazati matične brojeve i imena radnika čija plata je veća od 4000, a rade na projektu sa šifrom 11
- $F ::= PLT > 4000 \wedge SPR = 11$
- $\pi_{MBR+IME}(\sigma_F(r))$

MBR	IME
110	Ivo
111	Olja

**Prirodni spoj relacija** je spajanje torki različitih relacija po osnovu istih vrednosti zajedničkih obeležja.  
Date su relacije  $r(R)$  i  $s(S)$ .

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

Primeri:

$r$	A	B	C	$s$	B	C	D
	$a_1$	$b_1$	$c_1$		$b_1$	$c_1$	$d_1$
	$a_1$	$b_2$	$c_2$		$b_1$	$c_1$	$d_2$
	$a_1$	$b_3$	$c_3$		$b_3$	$c_3$	$d_3$

$r \bowtie s$	A	B	C	D
	$a_1$	$b_1$	$c_1$	$d_1$
	$a_1$	$b_1$	$c_1$	$d_2$
	$a_1$	$b_3$	$c_3$	$d_3$

$r$	A	B	$s$	C	D
	$a_1$	$b_1$		$c_1$	$d_1$
	$a_2$	$b_2$		$c_2$	$d_2$

$r \bowtie s$	A	B	C	D
	$a_1$	$b_1$	$c_1$	$d_1$
	$a_1$	$b_1$	$c_2$	$d_2$
	$a_2$	$b_2$	$c_1$	$d_1$
	$a_2$	$b_2$	$c_2$	$d_2$

Radnik

MBR	IME	PLT	POL
101	Ana	3400	ž
102	Aca	4200	m
110	Ivo	7000	m
111	Olja	7200	ž

Radproj

MBR	SPR
101	11
101	14
102	14
110	13
110	11

Projekat

SPR	NAP
11	X25
13	Polaris
14	Univ. IS

### Upit

- izlistati matične brojeve radnika, šifre i nazive projekata na kojima rade
- $\text{Radproj} \bowtie \text{Projekat}$

MBR	SPR	NAP
101	11	X25
101	14	Univ. IS
102	14	Univ. IS
110	13	Polaris
110	11	X25

### Upit

- Izlistati matične brojeve i imena radnika, koji rade na projektu sa šifrom 11
- $\pi_{MBR+IME}(\sigma_{SPR=11}(\text{Radproj}) \bowtie \text{Radnik})$ , ili
- $\pi_{MBR+IME}(\sigma_{SPR=11}(\text{Radproj} \bowtie \text{Radnik}))$

MBR	IME
101	Ana
110	Ivo

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

Date su relacije  $r(R)$  i  $s(S)$ , i važi  $R \cap S = \emptyset$ .

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

**Theta spajanje** relacija je selektovanje torki po nekom kriterijumu iz Dekartovog proizvoda relacija.

$$r(R) \triangleright \triangleleft_F s(S) = \sigma_F(r \times s)$$

Primer:

- date su relacije
  - $r$  - red vožnje Niš – Beograd
  - $s$  - red vožnje Beograd - Novi Sad

$r$	$PNI$	$DBG$
	06:00	09:00
	08:00	10:30
	13:00	16:00

$s$	$PBG$	$DNS$
	10:00	11:15
	12:00	13:30

### Upit

- pregled svih mogućih varijanti za putovanje od Niša do Novog Sada s presedanjem u Beogradu
- $r \triangleright \triangleleft_{DBG < PBG} s = \sigma_{DBG < PBG}(r \times s)$

$r \triangleright \triangleleft_{DBG < PBG} s$	$PNI$	$DBG$	$PBG$	$DNS$
	06:00	09:00	10:00	11:15
	06:00	09:00	12:00	13:30
	08:00	10:30	12:00	13:30

## Strukturalna komponenta II

**Šema relacije** je imenovani par:  $N(R,O)$

- $N$  - naziv šeme relacije (može biti izostavljen)
- $R$  - skup obeležja šeme relacije
- $O$  - skup ograničenja šeme relacije

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

Primer:

- Data je šema relacije
  - Letovi( $\{P, A, L\}$ ,  $O$ )
  - $O = \{\text{"Pilot može da leti samo na jednom tipu aviona"}\}$

$Let1$	$P$	$A$	$L$
	Pop	747	101
	Pop	747	102
	Ana	737	103

$Let2$	$P$	$A$	$L$
	Pop	747	101
	Pop	737	102
	Ana	737	103

- Da li prikazane relacije predstavljaju pojave nad datom šemom relacije?

**Relaciona šema baze podataka** je (imenovani) par:  $(S, I)$

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

Primer:

- Zadate su šeme relacija

- Radnik({MBR, IME, PRZ, DATR},  
{"Ne postoje dva radnika sa istom vrednošću za MBR. Svaki radnik poseduje vrednost za MBR."})
- Projekat({SPR, NAP},  
{"Ne postoje dva projekta sa istom vrednošću za SPR. Svaki projekat poseduje vrednost za SPR."})
- Angažovanje({SPR, MBR, BRC},  
{"Ne može se isti radnik na istom projektu angažovati više od jedanput. Pri angažovanju, vrednosti za MBR i SPR su uvek poznate."})

-  $S = \{\text{Radnik, Projekat, Angažovanje}\}$

-  $I = \{$

"radnik ne može biti angažovan na projektu, ako nije zaposlen";  
"na projektu ne može biti angažovan ni jedan radnik, dok projekat ne bude registrovan"  
}

-  $(S, I)$  predstavlja jednu relacionu šemu BP

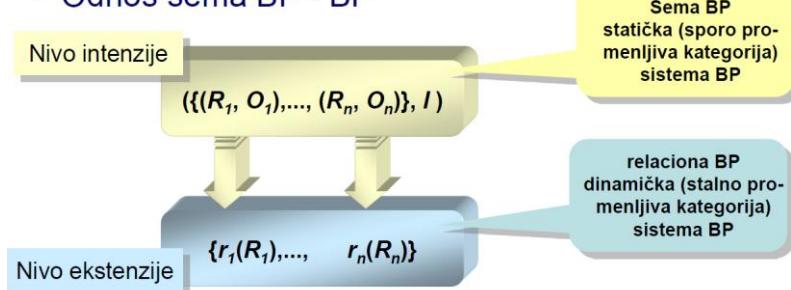
**Relaciona baza podataka** je jedna pojava nad zadatom relacionom šemom baze podataka  $(S, I)$ :

$$s: S \rightarrow \{r_i \mid i \in \{1, \dots, n\}\}, (\forall i) s(R_i, O_i) = r_i$$

Svakoj šemi relacije iz skupa  $S$  odgovara jedna njena pojava. Skup relacija  $s$  mora da zadovoljava sva međurelaciona ograničenja iz skupa  $I$ .

Baza podataka reprezentuje jedno stanje realnog sistema. Ažurira se, jer promene stanja realnog sistema treba da prate odgovarajuće promene podataka u bazi podataka.

- **Odnos šema BP - BP**



Primer šeme BP i BP:

- $S = \{\text{Radnik, Projekat, Angažovanje}\}$
- $RBP = \{\text{radnik, projekat, angažovanje}\}$

Radnik				Projekat		Angažovanje	
MBR	IME	PRZ	DATR	SPR	NAP	MBR	SPR
101	Ana	Pap	12.12.65.	11	X25	101	11
102	Aca	Tot	13.11.48.	13	Polaris	101	14
110	Ivo	Ban	01.01.49.	14	Univ. IS	102	14
111	Olja	Kun	06.05.71.				

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

- **formalno konzistentnom stanju** ako
  - $(\forall r_i \in RBP)(r_i \text{ zadovoljava sva ograničenja odgovarajuće šeme } ())$
  - 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 (u praksi, nivo pojave grešaka u bazama podataka sveden je na ispod 2-3%)

SUBP može da kontroliše formalnu konzistentnost.

## Integritetna komponenta

Integritetna komponenta definisana je putem **tipova ograničenja**.

Karakteristike tipa ograničenja su:

- 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 (definiše se za svaku operaciju koja može dovesti do narušavanja ograničenja)

Tipovi ograničenja u relacionom modelu podataka:

- ograničenje domena
- ograničenje vrednosti obeležja
- ograničenje torke
- integritet entiteta (ograničenje ključa)
- ograničenje jedinstvenosti vrednosti obeležja
- zavisnost sadržavanja
- ograničenje referencijalnog integriteta
- funkcionalna zavisnost

Oblasti definisanosti u relacionom modelu podataka:

- *vanrelaciono* ograničenje - definiše se izvan konteksta šeme relacije
- *jednorelaciono* (unutarrelaciono, lokalno) 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 modelu podataka:

- 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
- medurelaciono ograničenje - interpretira se nad barem dve, bilo koje relacije

Napomena: "bilo koja relacija" predstavlja jednu relaciju iz baze podataka ili relaciju koja je nastala primenom izraza relacione algebre nad jednom ili više drugih relacija - pogled (moguća je i primena operatora spajanja).

## Ograničenje domena

Specifikacija domena je struktura:  $D(id(D), Predef)$

- $D$  - naziv domena
- $id(D)$  - ograničenje domena
- $Predef$  - predefinisana vrednost domena

**Ograničenje domena**  $id(D)$  je struktura:  $id(D) = (Tip, Dužina, Uslov)$

- $Tip$  - tip podatka (oznaka primitivnog domena ili oznaka prethodnog, korisnički definisanog domena).  $Tip$  je jedina obavezna komponenta specifikacije ograničenja domena.
- $Dužina$  - dužina tipa podatka. Dužina se navodi samo za tipove podataka (primitivne domene) koji to zahtevaju
- $Uslov$  - logički uslov, koji svaka vrednost iz skupa mogućih vrednosti domena mora da zadovolji.

$Predef$  mora da zadovolji ograničenja *tipa*, *dužine* i *uslova*.

Interpretacija integriteta domena moguća je za bilo koju vrednost - konstantu  $d$  ( oznaka:  $id(D)(d)$  ).

Primeri:

- $DPrezime((String, 30, \Delta), \Delta)$
- $DDatum((Date, \Delta, d \geq '01.01.1900'), \Delta)$
- $DOcena((Number, 2, d \geq 5 \wedge d \leq 10), \Delta)$
- $DPozOcena((DOcena, \Delta, d \geq 6), 6)$ 
  - $\Delta$  - komponenta u specifikaciji nije zadata

## Nula vrednost

**Nula (nedostajuća) vrednost** je specijalna vrednost obeležja, tj. specijalno ograničenje domena. Označava se posebnim simbolom  $\omega$ . (u praksi, to je oznaka NULL)

Moguća značenja nula vrednosti:

- nepoznata - postojeća vrednost obeležja
- nepostojeća vrednost obeležja
- neinformativna vrednost obeležja

Skup mogućih vrednosti svih domena proširuje se nula vrednošću:  $DOM \cup \{\omega\}$

Nula vrednost "a priori" zadovoljava svako ograničenje domena.

## Ograničenje vrednosti obeležja

Specifikacija obeležja  $A \in R$  šeme relacije  $N(R, O)$  je struktura:  $(id(N, A), Predef)$

- $id(N, A)$  - ograničenje vrednosti obeležja
- $Predef$  - predefinisana vrednost obeležja

**Ograničenje vrednosti obeležja**  $id(N, A)$  definiše se za svako obeležje tipa i predstavlja strukturu:

$$id(N, A) = (Domen, Null)$$

- $Domen$  - oznaka (naziv) pridruženog domena obeležja
- $Null \in \{T, \perp\}$  - ograničenje nula vrednosti obeležja
  - $T$  - dozvola dodele nula vrednosti obeležju unutar  $N$
  - $\perp$  - zabrana dodele nula vrednosti obeležju unutar  $N$

$Domen$  i  $Null$  su obavezne komponente specifikacije

$Predef$  - ako se navede, onda je on važeći. U protivnom, važeći je  $Predef$  odgovarajućeg  $Domena$ , ili prvog sledećeg nasleđenog domena, za koji je  $Predef$  definisan

Interpretacija ograničenja moguća je za bilo koju vrednost obeležja  $d$  ( oznaka:  $id(N, A)(d)$  ).

## Ograničenje torke

**Ograničenje torke** izražava ograničenja na moguće vrednosti unutar jedne torke. Predstavlja skup ograničenja vrednosti obeležja, kojem je pridodat logički uslov.

Formalno, za šemu relacije  $N(R,O)$ :

$$\text{id}(N) = \text{id}(R) = (\{\text{id}(N, A) \mid A \in R\}, \text{Uslov})$$

- *Uslov* - logički uslov koji svaka torka mora da zadovolji. Može, u ulozi operanda, da sadrži bilo koje obeležje date šeme relacije.

Interpretacija ograničenja moguća je za bilo koju torku  $t$  nad skupom obeležja  $R$  ( oznaka:  $\text{id}(N)(t)$  ).

Primer:

- *Radnik*( $\{MBR, PRZ, IME, ZAN, BPJZ\}$ , O)

Rađnik	Domen	Null	Predef
MBR	MBRD	⊥	Δ
PRZ	PRZD	⊥	Δ
IME	IMED	⊥	Δ
ZAN	ZAND	⊥	Δ
BPJZ	BPJZD	T	Δ
<b>Uslov:</b> $ZAN = 'prg' \Leftrightarrow BPJZ <> \emptyset$			

Domen	Tip	Dužina	Uslov	Predef
MBRD	Number	4	$d \geq 0$	Δ
PRZD	String	30	Δ	Δ
IMED	String	15	Δ	Δ
ZAND	String	3	Δ	Δ
BPJZD	Number	2	$d \geq 0$	0

## Integritet entiteta (ograničenje ključa)

Ključ šeme relacije je minimalan podskup skupa pobeležja šeme relacije, na osnovu kojeg se jedinstveno može identifikovati svaka torka relacije nad datom šemom.

Formalno, X je ključ ako

- 1<sup>0</sup>  $(\forall u, v \in r(R))(u[X] = v[X] \Rightarrow u = v)$
- 2<sup>0</sup>  $(\forall Y \subset X)(\neg 1^0)$

Oblast interpretacije je skup torki (relacija) nad datom šemom relacije.

U određenim situacijama (u procesu projektovanja šeme baze podataka) skup ograničenja šeme relacije zadaje se samo kao skup ključeva:  $N(R,K)$

Primer:

- šema relacije *Radnik*( $R, K$ )
  - $R = \{MBR, IME, PRZ, DATR, POL, MESR, RBRE\}$
  - $K = \{MBR, DATR+MESR+POL+RBRE\}$
- *Radnik*( $\{MBR, IME, PRZ, DATR\}$ ,  $\{MBR\}$ )
- *Projekat*( $\{SPR, NAP\}$ ,  $\{SPR\}$ )
- *Angažovanje*( $\{SPR, MBR, BRC\}$ ,  $\{SPR+MBR\}$ )

**Ograničenje ključa (integritet entiteta)** šeme relacije  $N(R,K)$  predstavljeno je oznakom  $\text{Key}(N, X)$ .

Za sva obeležja ključa nula vrednosti su zabranjene:

$$(\forall K_i \in K)(\forall A \in K_i)(\text{Null}(N, A) = \perp)$$

Vrste obeležja, s obzirom na ključeve:

- **primarno** (ključno) obeležje - priprada barem jednom ključu šeme relacije
- **neprimarno** (sporedno) obeležje - ne pripada nijednom ključu šeme relacije

Svaka šema relacije mora posedovati barem jedan ključ ( $K \neq \emptyset$ ) - proizilazi iz definicije pojma relacije.

**Ekvivalentni ključevi** su svi ključevi skupa ključeva  $K$ .

**Primarni ključ**  $K_p(N)$  je jedan izabrani ključ, od svih ekvivalentnih ključeva. Svaka šema relacije treba da poseduje tačno jedan primarni ključ. Koristi se u ulozi asocijativne (simboličke) adrese za povezivanje podataka u relacijama.

Ograničenje jedinstvenosti vrednosti obeležja

**Ograničenje jedinstvenosti** obeležja šeme relacije  $N(R, O)$  predstavljeno je oznakom  $\text{Unique}(N, X)$ .

- $X$  - skup obeležja,  $X \subseteq R$

Ograničenje jedinstvenosti zahteva da ne-nula kombinacija vrednosti obeležja bude jedinstvena u relaciji nad  $N(R, O)$ .

Formalno:

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

Oblast interpretacije ograničenja jedinstvenosti je skup torki (relacija) nad datom šemom  $N(R, O)$ .

Skup svih ograničenja jedinstvenosti u šemi  $N(R, O)$ :

$$\text{Uniq} = \{\text{Unique}(N, X) \mid X \subseteq R\}$$

Primer:

$\text{Radnik}(\{MBR, IME, PRZ, DATR, JMBG\}, O)$

- $\text{Uniq} \subseteq O$
- $\text{Uniq} = \{\text{Unique}(\text{Radnik}, JMBG)\}$
- $\text{Unique}(\text{Radnik}, JMBG)$ 
  - zahteva da ako radnik poseduje ne-nula vrednost za  $JMBG$ , onda je ta vrednost jedinstvena u relaciji nad šemom  $\text{Radnik}$

Praktično, kada šemu relacije treba implementirati u datom SUBP, **skup svih ograničenja šeme relacije** zadaje se kao unija:

- skupa ključeva
- ograničenja jedinstvenosti i
- ograničenja torke

$$N(R, K \cup \text{Uniq} \cup \{id(R)\})$$

Primer:

$\text{Radnik}(\{MBR, PRZ, IME, ZAN, BPJZ, JMBG\},$   
 $K \cup \text{Uniq} \cup \{id(R)\})$

- $K = \{MBR\}$
- $\text{Uniq} = \{\text{Unique}(\text{Radnik}, JMBG)\}$
- $id(R)$  - prethodno zadat, u tabelarnom obliku

## Zavisnost sadržavanja

- date su šeme relacije  $N_i(R_i, O_i)$  i  $N_j(R_j, O_j)$
- dati su domenski kompatibilni nizovi obeležja

$$X = (A_1, \dots, A_n), (\forall I \in \{1, \dots, n\})(A_I \in R_i),$$

$$Y = (B_1, \dots, B_n), (\forall I \in \{1, \dots, n\})(B_I \in R_j),$$

$$(\forall I \in \{1, \dots, n\})(dom(A_I) \subseteq dom(B_I))$$

- oznaka (pravilo zapisivanja)

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

**Zavisnost sadržavanja** važi ako je za bilo koje dve relacije  $r(R_i, O_i)$  i  $s(R_j, O_j)$  zadovoljeno:

$$(\forall u \in r)(\exists v \in s)(\forall I \in \{1, \dots, n\})(u[A_I] = \omega \vee u[A_I] = v[B_I])$$

Oblast definisanosti je niz od dve šeme relacije.

Oblast interpretacije su relacije nad šemama  $N_i$  i  $N_j$ .

Primer:

- date su relacije  $r(N_i)$  i  $s(N_j)$
- važi zavisnost sadržavanja  $N_i[B] \subseteq N_j[B]$
- važi zavisnost sadržavanja  $N_i[(A, B)] \subseteq N_j[(C, D)]$

$r$	$A$	$B$
	$a_1$	$b_1$
	$a_2$	$b_2$

$s$	$B$	$C$
	$b_1$	$c_1$
	$b_2$	$c_1$
	$b_3$	$c_2$

$r$	$A$	$B$
	$a_1$	$b_1$
	$a_2$	$\omega$

$s$	$C$	$D$
	$a_1$	$b_1$
	$a_2$	$b_2$
	$a_3$	$b_2$

## Ograničenje referencijalnog integriteta

**Ograničenje referencijalnog integriteta** je zavisnost sadržavanja  $N_i[X] \subseteq N_j[Y]$ , kada je Y ključ šeme relacije  $N_j(R_j, O_j)$ .

- $N_i$  - **referencirajuća** šema relacije
- $N_j$  - **referencirana** šema relacije

Primer:

- Angažovanje[MBR]  $\subseteq$  Radnik[MBR]
- Angažovanje[SPR]  $\subseteq$  Projekat[SPR]

Radnik

MBR	IME	PRZ	DATR
101	Ana	Pap	12.12.65.
102	Aca	Tot	13.11.48.
110	Ivo	Ban	01.01.49.
111	Olja	Kun	06.05.71.

Projekat

SPR	NAP
11	X25
13	Polaris
14	Univ. IS

Angažovanje

MBR	SPR
101	11
101	14
102	14

## Funkcionalna zavisnost

**Funkcionalna zavisnost** je izraz oblika  $f : X \rightarrow Y$

- $X$  i  $Y$  - skupovi obeležja (podskupovi skupa  $U$ )
- $f$  - oznaka funkcionalne zavisnosti (često se izostavlja)

Semantika: Ako je poznata  $X$  vrednost, poznata je i  $Y$  vrednost. Svakoj  $X$  vrednosti odgovara samo jedna  $Y$  vrednost.

Relacija  $r$  zadovoljava funkcionalnu zavisnost  $X \rightarrow Y$  ako važi:

$$(\forall u, v \in r)(u[X] = v[X] \Rightarrow u[Y] = v[Y])$$

Oblast interpretacije funkcionalne zavisnosti je relacija  $r(N)$  ili  $r(U)$ .

Primer:

- $MBR \rightarrow IME$ 
  - ako dve torke imaju istu vrednost za  $MBR$ , moraju imati istu vrednost i za  $IME$

**Skup funkcionalnih zavisnosti** označava se sa  $F$ .

- $F = \{MBR \rightarrow IME, MBR+MES+GOD \rightarrow BRC, \dots\}$

**Trivijalna** funkcionalna zavisnost je svaka funkcionalna zavisnost koja je zadovoljena u bilo kojoj relaciji, tj. svaka funkcionalna zavisnost  $X \rightarrow Y$ , za koju važi  $Y \subseteq X$ .

Primer:

- $MBR \rightarrow MBR, MBR \rightarrow \emptyset, AB \rightarrow A, \dots$

Primer:

- semantika uvedenih obeležja skupa  $U$ 
  - $BRI$  - broj indeksa
  - $IME$  - ime studenta
  - $PRZ$  - prezime studenta
  - $BPI$  - broj položenih ispita
  - $OZP$  - oznaka predmeta
  - $NAP$  - naziv predmeta
  - $NAS$  - prezime nastavnika
  - $OCE$  - ocena na ispitu

*Student*

BRI	IME	PRZ	BPI	OZP	NAP	NAS	OCE
159	Ivo	Ban	13	P1	Mat	Han	09
159	Ivo	Ban	13	P2	Fiz	Kun	08
013	Ana	Tot	09	P1	Mat	Pap	06
119	Eva	Kon	15	P3	Hem	Kiš	07
159	Ivo	Ban	13	P3	Hem	Kiš	10
119	Eva	Kon	15	P1	Mat	Han	09
159	Ivo	Ban	13	P4	Mat	Car	10
037	Eva	Tot	01	P4	Mat	Car	10

- Relacija *Student* zadovoljava sledeće FZ

$$\begin{aligned} F = & \{BRI \rightarrow IME + PRZ + BPI, IME + PRZ \rightarrow BRI, OZP \rightarrow NAP, \\ & NAS \rightarrow OZP + NAP, BRI + OZP \rightarrow OCE + NAS\} \end{aligned}$$

- Relacija *Student* ne zadovoljava sledeće FZ  
 $BRI \rightarrow OCE, OZP \rightarrow NAS, \dots$

Važeće funkcionalne zavisnosti identifikuju se na osnovu odnosa i pravila poslovanja, koji postoji u realnom sistemu.

Funckionalna zavisnost  $f$  je **logička posledica** od skupa funkcionalnih zavisnosti  $F$  (oznaka  $F \models f$ ) ako svaka relacija  $r$  koja zadovoljava  $F$  zadovoljava i  $f$ .

$$(\forall r \in SAT(U))(r \models F \Rightarrow r \models f)$$

Skup funkcionalnih zavisnosti  $F_2$  je logička posledica od skupa funkcionalnih zavisnosti  $F_1$  (oznaka  $F_1 \models F_2$ ) ako

$$(\forall f \in F_2)(F_1 \models f)$$

Rešiti **implikacioni problem** znači utvrditi da li važi  $F \models f$ .

Dva skupa su **ekvivalentna** (oznaka  $F_1 \equiv F_2$ ) ako

$$F_1 \models F_2 \wedge F_2 \models F_1$$

**Zatvarač (zatvoreno)** skupa funkcionalnih zavisnosti (oznaka  $F^+$ ) je skup koji sadrži sve logičke posledice od  $F$ .

Za svaku funkcionalnu  $F$  zavisnost važi  $F \subseteq F^+$ .

Za dve funkcionalne zavisnosti  $F_1$  i  $F_2$  važi:

$$F_1 \models F_2 \text{ akko } F_2^+ \subseteq F_1^+$$

Ekvivalentnost skupova funkcionalnih zavisnosti  $F_1$  i  $F_2$ :

$$F_1 \equiv F_2 \text{ akko } F_1^+ = F_2^+$$

### – Armstrongova pravila izvođenja

- refleksivnost
  - $Y \subseteq X \vdash X \rightarrow Y$
- proširenje
  - $X \rightarrow Y, W \subseteq V \vdash XV \rightarrow YW$
- pseudotranzitivnost
  - $X \rightarrow Y, YV \rightarrow Z \vdash XV \rightarrow Z$

### – Izvedena pravila izvođenja

- uniranje desnih strana
  - $X \rightarrow Y, X \rightarrow Z \vdash X \rightarrow YZ$
- dekompozicija desnih strana
  - $X \rightarrow Y, V \subseteq Y \vdash X \rightarrow V$
- tranzitivnost
  - $X \rightarrow Y, Y \rightarrow Z \vdash X \rightarrow Z$

Primer:

- varijante u označavanju
  - primena pravila dekompozicije i uniranja desnih strana  
 $\{BRI \rightarrow IME, BRI \rightarrow PRZ\} \equiv \{BRI \rightarrow IME + PRZ\}$
  - proizvoljno dekomponovanje levih strana nije dozvoljeno  
 $\{BRI + OZP \rightarrow OCE\} \not\equiv \{BRI \rightarrow OCE, OZP \rightarrow OCE\}$

### – Sistem armstrongovih pravila izvođenja je

- refleksivnost, proširenje i pseudotranzitivnost
- **korektan (neprotivurečan)**
  - svaka FZ koja se izvede primenom AP iz nekog skupa FZ predstavlja logičku posledicu tog skupa FZ
- **kompletan**
  - svaka logička posledica nekog skupa FZ može se izvesti primenom AP iz tog skupa
- **neredundantan (minimalan)**
  - ne može se eliminisati kao suvišno ni jedno od tri pravila izvođenja, a da prethodna dva svojstva ostanu očuvana

Funkcionalna zavisnost  $X \rightarrow Y \in F$  je **nepotpuna** ako sadrži logički suvišno obeležje na levoj strani.

$$(\exists X' \subset X)(X' \rightarrow Y \in F^+)$$

Primer:

- $BRI+IME \rightarrow PRZ$ , zbog  $BRI \rightarrow IME$
- redukuje se u  $BRI \rightarrow PRZ$

Funkcionalna zavisnost  $X \rightarrow Y$  je **tranzitivna** ako važi  $X \rightarrow Y \in F^+$  i  $Y \rightarrow Z \in F^+$ , a ne važi da je  $Y \rightarrow X \in F^+$ .

Primer:

- $NAS \rightarrow OZP$ ,  $OZP \rightarrow NAP$ ,  $\neg(OZP \rightarrow NAS)$ 
  - $NAS \rightarrow NAP$  je tranzitivna i logički suvišna

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

- 1<sup>0</sup> iz  $F$  sledi  $X \rightarrow R$  ( $X \rightarrow R \in F^+$ )
- 2<sup>0</sup>  $X$  je minimalni skup obeležja s osobinom 1<sup>0</sup>
  - $\neg(\exists X' \subset X)(X' \rightarrow R \in F^+)$

Zatvarač (zatvorene) skupa obeležja  $X$  je skup svih obeležja koja funkcionalno zavisi od  $X$ .

$$X_F^+ = \{A \subseteq U \mid X \rightarrow A \in F^+\}$$

#### • Algoritam za izračunavanje zatvarača $X_F^+$

- $X_0 \leftarrow X$
- (Za  $i \geq 0$ ) ( $X_{i+1} \leftarrow X_i \cup \{A \subseteq U \mid (\exists V \rightarrow W \in F)(V \subseteq X_i \wedge A \in W)\}$ )
- (Za  $n \geq 0$ ) ( $X_{n+1} = X_n \Rightarrow X_F^+ = X_n$ )

#### • Generisanje jednog ključa šeme relacije

- polazi se od  $R$  i vrši se redukcija
  - izbacivanjem obeležja i izračunavanjem zatvarača ostatka
  - $X \leftarrow R$
  - Redukcija  $Red(X)$ :  $(\forall A \in X)(A \in (X \setminus \{A\})_F^+ \Rightarrow X \leftarrow X \setminus \{A\})$

#### • Generisanje svih alternativnih ključeva

- polazi se od prvog generisanog ključa  $X$ ,  $K \leftarrow \{X\}$ 
  - $(\forall X \in K)(\forall V \rightarrow W \in F)(X \cap W \neq \emptyset \Rightarrow X_{newk} \leftarrow (X \setminus W)V)$
  - Redukcija:  $Red(X_{newk})$ :  $K \leftarrow K \cup \{Red(X_{newk})\}$

Primer:

$$\begin{aligned} F = \{ & BRI \rightarrow IME + PRZ + BPI, IME + PRZ \rightarrow BRI, OZP \rightarrow NAP, \\ & NAS \rightarrow OZP + NAP, BRI + OZP \rightarrow OCE + NAS \} \end{aligned}$$

- šema relacije *Student* ima četiri ključa
  - $K_1 = BRI + NAS$ ,  $K_2 = IME + PRZ + NAS$ ,
  - $K_3 = BRI + OZP$ ,  $K_4 = IME + PRZ + OZP$

Pojam ključa je fundamentalan za teoriju i praksu relacionog modela podataka. Ne projektuju se ostala ograničenja šeme baze podataka, dok se ne preciziraju ključevi svih šema relacija.

## Projekcija skupa funkcionalnih zavisnosti na skup obeležja

- dati su skup fz  $F$  i skup obeležja  $X \subseteq U$
- projekcija  $F|_X$  predstavlja skup svih funkcionalnih zavisnosti koje logički slede iz  $F$ , a definisane su u skupu obeležja  $X$
- formalno

$$F|_X = \{V \rightarrow W \mid F \models V \rightarrow W \wedge VW \subseteq X\}$$

Primer:

- $F = \{A \rightarrow B, B \rightarrow C, BE \rightarrow F, A \rightarrow D\}$
- $F|_{ACDEF} = \{A \rightarrow C, AE \rightarrow F, A \rightarrow D\}$ , sve trivijalne fz

## Osnovne projektantske pretpostavke

### • Pretpostavka o postojanju šeme univerzalne relacije (ŠUR)

- $U$  - univerzalni skup obeležja
- $OGR$  - skup svih ograničenja realnog sistema
- **šema univerzalne relacije**  
**( $U, OGR$ )**
  - pretpostavka da uvek egzistira u imaginarnom svetu
  - Posledica
    - jedinstvena uloga svakog obeležja u ŠUR
      - ne postoje dva obeležja s istom ulogom (**sinonimi**)
      - ne postoji obeležje s više od jedne uloge (**homonimi**)
    - svako obeležje u budućoj šemi BP identificuje se isključivo putem svog naziva

### • Univerzalna relacija

- pojava nad ŠUR,  $r(U, OGR)$
- reprezentuje stanje realnog sistema
- apstraktни pojam – preuzet iz imaginarnog sveta
- praktično, nemoguće je implementirati je pod nekim SUBP
  - prepreke logičkog karaktera i
  - prepreke vezane za moguću fizičku organizaciju podataka

### • teoretski zahtev

- stanje relacione baze podataka nad ( $S, I$ ) treba, u informativnom smislu, da odgovara sadržaju univerzalne relacije

### • Šema BP ( $S, I$ ) treba da zadovolji sledeće kriterijume u odnosu na ( $U, OGR$ )

- da predstavlja dekompoziciju ŠUR:

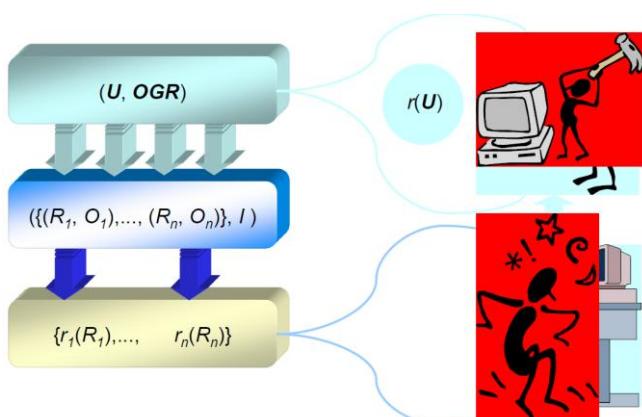
$$(\forall N_i \in S)(R_i \neq \emptyset) \wedge \cup_{N_i \in S}(R_i) = U$$

- skup svih ograničenja da bude ekvivalentan polaznom skupu ograničenja  $OGR$

$$\cup_{N_i \in S}(O_i) \cup I \equiv OGR$$

- spojivost bez gubitaka informacija

$$r(U, OGR) = \triangleright \triangleleft_{N_i \in S}(r_i(R_i))$$



# Fizičke strukture podataka i eksterni memorijski uređaji

## Datoteka

Datoteka:

- Kao logička struktura podataka (LSP)
  - struktura nad skupom pojava jednog tipa entiteta
    - struktura slogova, nad datim tipom sloga
    - često se posmatra kao linearna struktura slogova
- Kao fizička struktura podataka (FSP)
  - predstavlja jednu LSP
  - koja može biti viđena kao
    - linearna struktura (niz) slogova ili
    - niz znakova ili bajtova
  - smeštena na eksterni memorijski uređaj
  - zajedno sa informacijama o samom načinu smeštanja LSP na uređaj
- Motivacija za upotrebu eksternih memorijskih uređaja
  - potreba trajnog memorisanja podataka
  - potreba memorisanja velikih količina podataka
  - potreba tolerantno brzog pristupa i operativnog korišćenja velike količine trajno memorisanih podataka
  - potreba postizanja niske cene memorisanja po jedinici kapaciteta

Izbor memorijskog uređaja:

- OM – nepogodan izbor
  - prednosti
    - kada bi celu datoteku mogla stati odjednom u OM, svi postupci vezani za obradu i organizovanje podataka sveli bi se na teoriju algoritama i strukturu podataka u OM
    - kratko vreme pristupa (reda x10 ns) svakoj ćeliji, kao najmanjoj adresibilnoj jedinici
    - RAM pristup
      - » vreme pristupa ne zavisi položaja lokacije na memorijskom medijumu
      - » opredeljeno je radom elektroničkih komponenti
  - nedostaci
    - nedovoljan kapacitet
    - nemogućnost trajnog memorisanja podataka
    - i dalje značajno skuplje memorisanje po jedinici kapaciteta
- magnetni disk
  - praktično, i dalje realan izbor već više decenija
  - prednosti
    - veliki kapacitet
    - mogućnost trajnog memorisanja podataka
    - značajno jeftinije memorisanje po jedinici kapaciteta
    - direktni pristup
      - » pristupa se grupi ćelija (bitova) direktno, a zatim svakoj ćeliji u grupi sekvencialno
      - » mogućnost operativne upotrebe podataka
  - nedostaci
    - vreme pristupa zavisi položaja lokacije na memorijskom medijumu i bitno je duže nego u slučaju OM
    - sekundarni tip uređaja – nemogućnost direktnog prihvatanja podataka od strane centralnog procesora (CPU)

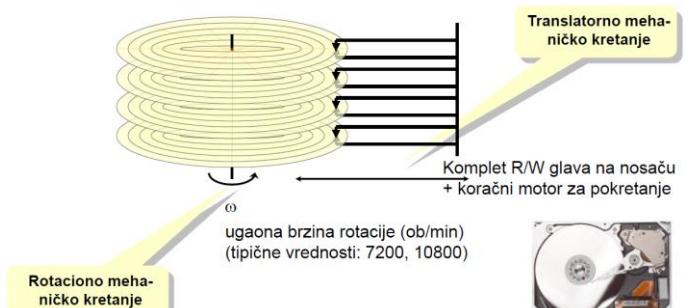
## Jedinice magnetnih diskova

- Opšta struktura memorijskog uređaja
  - upravljačka jedinica uređaja
    - upravljačka logika
    - adresni registar uređaja
    - registar podataka (prihvativa memorija) uređaja
    - registar statusa uređaja
    - sklop za vremensko vođenje uređaja
  - jedinica za memorisanje podataka
    - adresni mehanizam
    - memorijski medijum
    - pobudna kola
    - izlazni pojačavači

## Magnetni disk

- Magnetni disk
  - karakteristični predstavnik eksternih memorijskih uređaja s rotacionim kretanjem medijuma
    - ostali predstavnici: DVD/CD jedinice
    - isti princip organizacije
      - adresnog prostora na memorijskom medijumu i
      - adresnog mehanizma
    - različita tehnologija memorisanja podataka
  - memorijski medijum
    - jedna ili više kružnih ploča na rotirajućoj osovini
      - sa slojem feromagnetskog materijala sa obe strane
  - adresni mehanizam
    - komplet upisno-čitajućih glava na nosaču

- organizacija adresnog prostora i adresni mehanizam



## Organizacija adresnog prostora

- zasnovana na cilindričnom koordinatnom sistemu
  - koordinate vektora:  $(\rho, \varphi, z)$  - poluprečnik, ugao i visina
- **staza (track)**
  - kružnica koju opisuje upisno-čitajuća (R/W) glava na zadatom poluprečniku
  - podaci se upisuju na stazu "poduzno", u obliku niza bitova
- **ćelija diska = 1 bit**
  - najmanja jedinica upisa/čitanja podataka
- **cilindar (cylinder)**
  - matematički, skup svih staza istog poluprečnika
  - praktično, disk sadrži 1-2 ploče, tj. 2 ili 4 R/W glave
    - veći broj staza po jednom cilindru, tj. veći broj glava, postiže se softverskim putem
- **sektor (sector)**
  - luk na stazi, konstantnog ugla
  - staza se deli na konstantan broj sektora
  - ne retko, broj sektora na stazi:  $S = 2^n$
  - između svaka dva sektora postoji **međusektorski razmak**
- **najmanja adresibilna jedinica diska = sektor**
  - svakom sektoru pristupa se direktno
  - svakom bitu unutar sektora pristupa se sekvensijalno

## – vrste diskova s obzirom na kapacitet staze

- **diskovi sa stazama konstantnog kapaciteta**
  - promenljiva podužna gustina zapisa
    - » najveća na stazi najmanjeg poluprečnika i obratno
    - » gubici u postizanju maksimalnog mogućeg kapaciteta
- **diskovi sa stazama promenljivog kapaciteta**
  - konstantna podužna gustina zapisa
    - » najveća moguća na stazi, ili grupi susednih staza
    - » komplikovanija organizacija adresnog prostora
    - » približavanje maksimalnom mogućem kapacitetu

## • Sektorska organizacija adresnog prostora

- kod diskova sa stazama konstantnog kapaciteta
  - definisana podelom staze na konstantan broj sektora, konstantnog kapaciteta

## Sektorska organizacija adresnog prostora

### Uspostava adresnog prostora diska

- fabrička priprema + formatiranje diska od strane OS
- fabričke karakteristike
  - $C$  – ukupan broj cilindara diska
    - diktira maksimalni kapacitet diska
  - $T$  – ukupan broj staza po cilindru (broj glava)
    - tipično,  $T \geq 16$
  - $S$  – ukupan broj sektora na stazi
    - tipično,  $S \geq 64$

## – adresni prostor je diskretizovan, numeracijom cilindara, staza i sektora

- numeracija cilindara:  $\{0, \dots, C - 1\}$ 
  - nulli cilindar – cilindar najvećeg poluprečnika
- numeracija staza na jednom cilindru:  $\{0, \dots, T - 1\}$ 
  - određena redosledom glava na nosaču
- numeracija sektora na jednoj stazi:  $\{0, \dots, S - 1\}$ 
  - svaki sektor, u zaglavljtu, ima upisan svoj redni broj
  - jedan od sektora odabran je da bude početni
    - » on značava i početak staze

### – adresa sektora na disku

- $(u, c, t, s)$
- $u$
- $c \in \{0, \dots, C - 1\}$
- $t \in \{0, \dots, T - 1\}$
- $s \in \{0, \dots, S - 1\}$
- adresa uređaja (jedinice diska)
- redni broj cilindra (poluprečnik)
- redni broj staze na cilindru (visina)
- redni broj sektora na stazi (ugao)

## Kapacitet diska

### – kapacitet sektora – konstanta

- $K_s$  – efektivni kapacitet sektora
  - obuhvata prostor za korisne podatke
  - veličina, tipično:  $K_s = 512 \text{ B}$
- $K_s^h$  – kapacitet zaglavljiva sektora
  - obuhvata prostor za upisivanje identifikacionog broj identifikacionog broja zamenskog sektora
    - » ukoliko je dati sektor van upotrebe
- $K_s^e$  – kapacitet pratećeg dela sektora
  - obuhvata prostor za kontrolni (ECC, CRC) kod
    - » za detekciju i korekciju grešaka
    - » garantuje, verovatnoču nastanka do, na primer neoporavljivih grešaka
- $K_s^u$  – ukupni kapacitet sektora  $K_s^u = K_s + K_s^h + K_s^e$

### – efektivni kapacitet staze

$$K_t = SK_s$$

### – efektivni kapacitet cilindra

$$K_c = TK_t$$

### – efektivni kapacitet diska

$$K_d = CK_c$$

Primer:

- $K_s = 512 \text{ B}$  (korisnih podataka)
- $S = 180$  (sektora na stazi)
- $T = 80$  (staza po cilindru)
- $C = 50972$  (cilindara na disku)

### – efektivni kapacitet staze

$$K_t = SK_s = 180 * 512 \text{ B} = 92160 \text{ B} = 90 \text{ KB}$$

### – efektivni kapacitet cilindra

$$K_c = TK_t = 80 * 90 \text{ KB} = 7200 \text{ KB} = 7,03125 \text{ MB}$$

### – efektivni kapacitet diska

$$K_d = CK_c = 50972 * 7,03125 \text{ MB} = 358396,875 \text{ MB} \approx 350 \text{ GB}$$

## Vreme pristupa sektoru (access time, latency)

### – vreme pristupa podacima na disku

### – komponente, za zadatu adresu ( $u, c, t, s$ )

- vreme pozicioniranja kompleta glava na zadati cilindar  $c$ 
  - opredeljeno brzinom kretanja kompleta glava po poluprečniku
  - mehaničko (translatorno) kretanje
  - reda veličine ms
- vreme aktiviranja R/W glave za zadatu stazu  $t$ 
  - opredeljeno radom elektroničkih komponenata
  - bar za par redova veličina kraće – zanemaruje se
- vreme pozicioniranja R/W glave na početak zadatog sektora  $s$  – rotaciono kašnjenje
  - opredeljeno brzinom rotacije paketa diskova
  - mehaničko (rotaciono) kretanje
  - reda veličine ms

$$t_p = t_c + t_h + t_r$$

$$\bullet 0 \leq t_c \leq t_c^{\max}$$

- za predeni put kompleta glava  $i \in \{0, \dots, C - 1\}$ 
  - » iskazuje se brojem predenih cilindara od cilindra na kojem su glave prethodno bile pozicionirane do traženog cilindra
  - »  $t_c$  se može posmatrati kao linearna funkcija od  $i$ 
    - » za  $i = 0 \Rightarrow t_c = 0 \text{ ms}$
    - » za  $i = 1 \Rightarrow t_c = 0,8 - 2 \text{ ms}$  (tipično)
    - » za  $i = C - 1 \Rightarrow t_c = 14 - 24 \text{ ms}$  (tipično)
    - » zaključak: najbolje je da suksesivno traženi podaci budu smešteni na istom, ili bar susednom cilindru

$$\bullet t_h \approx 0$$

$$\bullet 0 < t_r \leq 1 / \omega$$

- zavisi obrnuto proporcionalno od ugaone brzine rotacije
- slučajna veličina sa uniformnom raspodelom na  $(0, 1 / \omega]$

### • Srednje vreme pristupa sektoru

$$-\bar{t}_p = \bar{t}_c + \bar{t}_r$$

### – srednje vreme pristupa cilindru

$$\bullet \text{tipično oko } 8 \text{ ms}$$

### – srednje rotaciono kašnjenje

$$\bullet \bar{t}_r = 1 / 2\omega$$

$$-\text{za } 7200 \text{ ob/min} \Rightarrow \bar{t}_r \approx 4,17 \text{ ms}$$

$$-\text{za } 10800 \text{ ob/min} \Rightarrow \bar{t}_r \approx 2,78 \text{ ms}$$

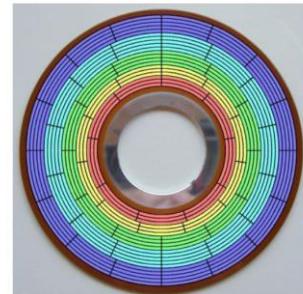
### – srednje vreme pristupa sektoru

$$\bullet \text{tipično oko } 9 - 12 \text{ ms}$$

## Zonsko sektorska organizacija adresnog prostora

- kod diskova sa stazama promenljivog kapaciteta
  - definisana podelom staza na sektore
  - svaki sektor je konstantnog kapaciteta
  - broj sektora na stazi - promenljiv
    - zavisi od poluprečnika staze
    - staze većeg poluprečnika sadrže više sektora
  - grupisanje staza po broju sadržanih sektora
  - **zona**
    - grupa susednih staza, sa istim brojem sektora
- tehniku organizacije adresnog prostora
  - **zoned bit recording (ZBR) / multiple zoned recording**

– ilustracija sa 5 zona (označene različitim bojama)



## Zaključne napomene

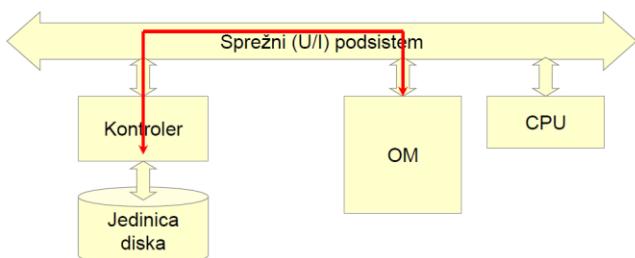
- za više od pet redova veličine duže vreme pristupa nego kod OM
  - disk, srednje vreme pristupa: ~ 9 – 12 ms
    - već duži niz godina
  - OM, ciklus (uključuje i srednje vreme pristupa): ~ 60 ns

## Potrebne mere

- poboljšati efikasnost prenosa podataka kroz U/I podsistem
  - efikasno korišćenje propusnog opsega sprežnog pod sistema
- smanjiti potreban broj pristupa
  - unapređivanjem sistema diskova
  - izborom pogodne fizičke organizacije

## Sprežni (U/I) podsistem

- sistem veza i algoritama za fizički prenos podataka između kontrolera periferijskog uređaja i OM
  - linije podataka
  - adresne linije
  - upravljačke i informacione linije



- osnovna karakteristika
  - **propusni opseg (bandwidth)**
    - mogući broj prenetih bajtova u jedinici vremena (Bps)
  - dominantno zavisi od
    - broja linija podataka ("širine") i ciklusa sprežnog pod sistema
    - ali i, efektivno, od propusnog opsega i ciklusa jedinice diska
      - » bitno sporiji uređaj od sprežnog pod sistema i OM
- zahteva definisanje fiksne jedinice prenosa podataka
  - na nivou operativnog sistema

# Osnovni koncepti

## – Fizički blok (blok)

- organizaciona jedinica memorisanja podataka
  - nedeljiva jedinica smeštanja podataka (lokacija) na jedinici eksternog memorijskog uređaja
  - osnovna (nedeljiva) jedinica alokacije prostora na eksternom memorijskom uređaju, **fiksнog kapaciteta**
  - u slučaju jedinice diska, niz sukcesivnih sektora na istoj stazi diska

## – Blok podataka (blok)

- organizaciona jedinica prenosa podataka
  - osnovna (nedeljiva) jedinica prenosa podataka, **fiksнog kapaciteta**

## – Fizički blok na disku zauzima

- uvek ceo broj sektora
  - fizički susednih sektora
  - na istoj stazi diska
- prednost
- garantuje se pristup celokupnom sadržaju bloka uz potrošnju najviše jednog vremena pristupa
    - srednjeg, ili u najgorem slučaju
- nedostaci
- spoljnja fragmentacija prostora
    - neiskorišćeni sektori na kraju staze – mogu služiti kao rezervni
  - unutrašnja fragmentacija prostora
    - ne mora ceo blok biti zauzet isključivo korisnim podacima

## – Kapacitet bloka / fizičkog bloka

- fiksna veličina
  - definisana unapred, na nivou operativnog sistema
  - kreće se u rasponu [512 B, 8 KB]
  - tipične veličine: 2 KB, 4 KB, 8 KB
- motivacija
- postizanje što boljeg iskorišćenja propusnog opsega
  - olakšano upravljanje prenosom podataka
- pravila
- U/I podsistem vrši prenos samo celih blokova podataka
  - jedinica diska obezbeđuje smeštanje i preuzimanje samo celih fizičkih blokova podataka

## – Sistemski bafer (sistemske prihvavnik)

- prostor u OM koji se alocira za potrebe smeštanja sadržaja jednog bloka podataka
  - pripada sistemskom (zaštićenom) delu OM
  - podaci razmenjeni sa eksternim memorijskim uređajem smeštaju se u bafere OM, samo u jedinicama blokova
- posledica
- zahteva se veći kapacitet OM
    - jer se u OM, putem blokova, prenose i podaci koji korisniku nisu neophodni u obradi i
    - neće biti preneti u memoriju korisničkog programa

# Kontroler jedinice diska

## – Zadaci

- dekodiranje i izvršavanje R/W komande, dobijene od CPU
- prijem adrese fizičkog bloka na disku
- upravljanje adresnim mehanizmom u cilju pozicioniranja na traženu adresu
  - izdavanje naloga (signala) upravljačkoj logici uređaja
- konverzija sadržaja bloka
  - iz bajt-serijskog oblika u niz memorijskih reči, pri čitanju i
  - iz niza memorijskih reči u bajt-serijski oblik, pri upisu podataka
- ispitivanje statusa spremnosti jedinice diska za predaju ili preuzimanje podataka
- privremeno memorisanje sadržaja bloka
  - "cache" memorija na jedinici diska
  - tipičnog kapaciteta 16 MB

# U/I podsistem za fizički prenos podataka

## • Zadaci

- inicijalizacija prenosa podataka
  - zadavanje
    - vrste R/W operacije
    - adrese bloka podataka na disku
    - adrese bafera u OM
    - kapaciteta bloka podataka za prenos
  - uvek je realizuje CPU
- fizička razmena podataka na relaciji kontroler – OM
  - iterativni postupak, razmena memorijska reč po reč
- ispitivanje statusa spremnosti uređaja

## • Vrste

- klasični "programirani" prenos
  - uslovni – sa ispitivanjem statusa spremnosti uređaja
    - uposleno čekanje procesora
  - bezuslovni – bez ispitivanja statusa spremnosti uređaja
- prenos iniciran prekidima
- Direct Memory Access (DMA) prenos
  - angažovanje DMA kontrolera za
    - fizičku razmenu podataka i
    - ispitivanje statusa spremnosti uređaja
- prenos putem specijalizovanih (kanalskih) procesora

## Efikasnost razmene podataka

- Parametri koji imaju dominantan uticaj

- srednje vreme pristupa bloku (sektoru) na disku
  - $t_p^{sr} = 9 - 12 \text{ ms}$
- vreme učitavanja / pisanja sadržaja bloka na disk
  - $K_b$  – kapacitet bloka
  - $K_t = SK_s$  – efektivni kapacitet staze
  - $T_b = K_b / (\omega K_t)$  – vreme učitavanja / pisanja sadržaja bloka
- propusni opseg diska (brzina razmene podataka)
  - $v_d = \omega K_t [\text{MB/s}]$

Primer:

- $K_s = 512 \text{ B}$ ,
- $S = 170, K_t = 87040 \text{ B} = 85 \text{ KB}$ ,
- $\omega = 10800 \text{ ob/min}$

$K_b$	512 B	1 KB	2 KB	4 KB	8 KB
$T_b$ (ms)	0,033	0,065	0,131	0,261	0,523

- zaključak:  $T_b << t_p^{sr}$ , čak i za veće vrednosti  $K_b$
- propusni opseg diska  $v_d = \omega K_t \approx 14,941 \text{ MB/s}$

Primer:

- $K_t = 85 \text{ KB}, T = 80,$
- $K_b = 4 \text{ KB}, T_b = 0,261 \text{ ms},$
- $t_p^{sr} = 10 \text{ ms}$
- ukupan broj blokova na stazi:  $b_t = \lfloor K_t / K_b \rfloor = 21$
- ukupan broj blokova na cilindru:  $b_c = Tb_t = 1680$

– (A) pretpostavka

- sukcesivno se učitava  $b_c = 1680$  blokova jednog cilindra
  - FSP datoteke to omogućava
- potrebno vreme razmene podataka
- $t_u^A = t_p^{sr} + b_c T_b = 10 \text{ ms} + 1680 * 0,261 \text{ ms} \approx 0,45 \text{ s}$

– (B) pretpostavka

- učitava se  $b_c = 1680$  blokova, slučajno raspoređenih po različitim cilindrima diska
- potrebno vreme razmene podataka
- $t_u^B = b_c(t_p^{sr} + T_b) = 1680 * (10 \text{ ms} + 0,261 \text{ ms}) \approx 17,24 \text{ s}$

– zaključak

- $t_u^B >> t_u^A$ , odnos  $t_u^B / t_u^A \approx 38$  puta
- potrebna je pogodna fizička organizacija datoteke

– (C) pretpostavka

- prenosi se sadržaj jednog bloka datoteke,  $K_b = 4 \text{ KB}$
- sa diska
  - potrebno vreme razmene podataka
  - $t_u^{CD} = t_p^{sr} + T_b = 10 \text{ ms} + 0,261 \text{ ms} \approx 10,261 \text{ ms}$
- iz memorije, kapaciteta reči  $K_w = 4 \text{ B}$  i ciklusa  $t_c = 60 \text{ ns}$ 
  - $t_u^{CM} = (K_b / K_w)t_c = 0,06144 \text{ ms}$

– zaključak

- $t_u^{CD} >> t_u^{CM}$ , odnos  $t_u^{CD} / t_u^{CM} = 167$  puta
- OM je brži uređaj
- pokazuje daleko bolju efikasnost razmene podataka u slučaju male količine podataka

– (D) pretpostavka

- prenosi se sadržaj sukcesivnih  $b_c = 1680$  blokova jednog cilindra,  $K_b = 4 \text{ KB}$
- sa diska
  - potrebno vreme razmene podataka
  - $t_u^{DD} = t_p^{sr} + b_c T_b = 10 \text{ ms} + 1680 * 0,261 \text{ ms} \approx 0,45 \text{ s}$
- iz memorije, kapaciteta reči  $K_w = 4 \text{ B}$  i ciklusa  $t_c = 60 \text{ ns}$ 
  - $t_u^{DM} = b_c(K_b / K_w)t_c = 1680 * 0,06144 \text{ ms} \approx 0,10 \text{ s}$

– zaključak

- $t_u^{DD} \sim t_u^{DM}$ , odnos  $t_u^{DD} / t_u^{DM} = 4,5$  puta
- bitno poboljšana efikasnost razmene podataka sa diska
- favorizacija upotrebe blokova većeg kapaciteta

## Sistemi disk jedinica

### – klasterske arhitekture sistema disk jedinica

- više nezavisnih jedinica diskova, povezanih jednim sprežnim sistemom
- jedinstveni adresni sistem i načini pristupa
  - od strane različitih procesorskih jedinica u arhitekturi

### – nizovi disk jedinica

- **Redundant Array of Independent Disks (RAID) sistemi**
- više jedinica diskova koje se ponašaju kao jedna
  - redundantno memorisanje podataka
  - diskovi su po potrebi izmenljivi

### – obezbeđuju razmeštanje istih, ili sukcesivno traženih blokova na više nezavisnih disk jedinica

- statistički, nije potrebno uvek čekati ukupno vreme pristupa

## Performanse obrade podataka

### Tehnike obezbeđenja dobrih performansi

- operativne obrade perzistentnih podataka

### – vrste tehniki

- (T) korišćenje uticaja tehnologije i tehnoloških parametara
- (A) projektovanje odgovarajuće arhitekture sistema diskova
- (O) izbor odgovarajućeg OS i podešavanje parametara OS
- (P) projektovanje odgovarajuće FSP datoteka

### – skraćenje srednjeg vremena pristupa

- izbor disk jedinica boljih proizvođačkih karakteristika, npr. veće brzine rotacije, kraćeg vremena pristupa, itd. (T)
- upotreba sistema diskova (RAID, klasteri) sa simultanim pristupom blokovima (A)
- raspoređivanje slučajno, a sukcesivno traženih blokova na različite disk jedinice (A+P)

### – efikasno korišćenje propusnog opsega diska

- izbor većeg kapaciteta bloka (O)
- izbor odgovarajuće FSP datoteke, saglasno potrebama programa (P)
- efikasnija upotreba raspoloživog kapaciteta bloka (P)

### – minimizacija potrebnog broja pristupa

- "keširanje" dela sadržaja diska u memoriji kontrolera (T)
- povećanje kapaciteta OM (T)
- "keširanje" dela sadržaja diska u OM (O)
- rezervacija većeg broja bafera za datoteku u OM (O)
- izbor odgovarajuće FSP datoteke, saglasno potrebama programa (P)

### – skraćenje vremena prenosa i obrade podataka

- izbor sprežnog podsistema boljih karakteristika, npr. većeg propusnog opsega (T)
- izbor OM boljih karakteristika, npr. kraćeg ciklusa (T)
- izbor CPU boljih karakteristika, npr. više frekvencije, sa više keš memorije ili više procesorskih jezgara (T)

## Organizacija datoteke i OS

### • Operativni sistem (OS)

- omogućava organizovanje različitih FSP datoteka
  - na jedinicama diskova
  - vodi računa o organizaciji podataka i upotrebi svih datoteka na jedinicama diskova
- može da pruža različite poglede na FSP datoteke
  - kao linearne strukture (niza) slogova
    - najčešće za potrebe korisničkih programa
  - kao niza znakova ili bajtova
    - za potrebe sistemskih programa, a
    - može i za potrebe korisničkih programa
  - kao (linearne ili neke drugačije) strukture blokova
    - za potrebe memorisanja i razmene podataka kroz U/I podsistem

### • Organizacija podataka na jedinici diska

- OS održava na jedinici diska strukture podataka o
  - proizvođačkim karakteristikama same disk jedinice
  - ispravnim i neispravnim sektorima, kao i o zamenskim sektorima za neispravne sektore
  - slobodnom i zauzetom prostoru (fizičkim blokovima) na disku
  - katalogu (hijerarhijskoj strukturi foldera) sa pokazivačima na opise datoteka
  - sistemskoj i alokacionoj tabeli svake datoteke
    - sistemski tabela sadrži osnovne podatke o datoteci
    - alokaciona tabela sadrži pokazivače na područja diska koja su alocirana za potrebe datoteke

# Datotečki sistem operativnog sistema

## Usluge OS u organizaciji datoteka

### • **Datotečki (File, U/I) sistem OS-a**

- upravljanje datotekama
  - fizičkim strukturama podataka (FSP) na eksternim memorijskim uređajima
- upravljanje i realizacija razmene podataka između aplikativnih programa i datoteka
- obezbeđenje mehanizama zaštite od neovlašćenog pristupa datotekama i oštećenja podataka
  - u uslovima višekorisničkog i multiprogramskega režima rada
- obezbeđenje podrške različitih pogleda na LSP datoteke
  - preslikavanja LSP ↔ FSP
  - obavlja jedan deo ili sve zadatke prethodna tri tipa

## Usluge datotečkog sistema

### – **usluge niskog nivoa**

- pokrivaju pobrojane zadatke, ali
- obezbeđuju pogled na LSP datoteke
  - samo kao niza bajtova (znakova) i
  - njeno preslikavanje u FSP niza blokova

### – **usluge visokog nivoa**

- obavezno uključuju sve usluge niskog nivoa
- obezbeđuju različite poglede na LSP datoteke
  - kao različitih struktura nad skupom slogova, blokova pa i bajtova i
  - njihovih preslikavanja u FSP niza blokova
- obezbeđuju izgradnju specijalnih pomoćnih struktura za poboljšanje efikasnosti obrade podataka
- obezbeđuju traženja, zasnovana na vrednostima podataka

### – **usluge niskog nivoa**

- servisi koji isključivo pripadaju operativnom sistemu

### – **usluge visokog nivoa**

- servisi koji mogu biti ugrađeni u
  - **operativni sistem**
    - » karakteristično za OS-ove mainframe računara
  - **pakete (biblioteke) funkcija programskog jezika**
    - » tipično za sve savremene programske jezike
  - **sistem za upravljanje bazama podataka**
    - » ovi servisi su obavezna komponenta svakog SUBP
- ili mogu biti prepušteni nivou aplikativnog programa
  - najčešće u slučaju razvoja specijalizovanih aplikacija i/ili
  - korišćenja specijalizovanih procesorskih uređaja za prikupljanje, memorisanje i razmenu podataka

### – **usluge niskog nivoa**

- 1. upravljanje prostorom eksternog memorijskog uređaja
- 2. upravljanje katalogom
- 3. upravljanje fizičkom razmenom podataka
- 4. obezbeđenje veze programa i datoteke
- 5. sistemski pozivi

### – **usluge visokog nivoa**

- 6. metode pristupa

- svaka od navedenih usluga uključuje odgovarajuće rutine i specijalizovane strukture podataka

## Upravljanje memorijskim prostorom

- Rutine za upravljanje prostorom eksternog memorijskog uređaja – jedinice diska
  - uspostava adresnog prostora i fajl sistema
    - formatiranje diska, niskog i visokog nivoa
    - kreiranje strukture podataka sa evidencijom slobodnog i zauzetog prostora diska (indeks slobodnih/zauzetih blokova)
  - održavanje strukture podataka sa evidencijom slobodnog i zauzetog prostora na disku
    - alociranje slobodnog prostora, na zahtev drugih rutina OS
    - dealociranje slobodnog prostora, na zahtev drugih rutina OS
    - reorganizacija (defragmentacija) slobodnog prostora, na zahtev administratora sistema
  - arhiviranje, restauracija i oporavak sadržaja diska
    - backup, restore i recovery rutine

## Uparavljanje katalogom

- Struktura kataloga (Katalog)
  - hijerarhijska struktura direktorijuma (foldera)
    - struktura tipa stabla
    - koren – korenski direktorijum
      - formira se automatski u postupku formatiranja diska
    - svaki čvor u strukturi može biti direktorijum ili datoteka
    - uvodi se pojam tekućeg direktorijuma
  - formiran na jednoj ili grupi jedinica eksternih memorijskih uređaja s direktnim pristupom
  - dozvoljava relativno i apsolutno referenciranje čvorova u strukturi
- Rutine za upravljanje katalogom
  - kreiranje, brisanje, preimenovanje i prevezivanje direktorijuma u strukturi
  - izlistavanje i pretraživanje sadržaja direktorijuma i datoteka
  - kreiranje i brisanje datoteka u direktorijumu
    - upotreba odgovarajućih sistemskih poziva
  - preimenovanje, kopiranje i premeštanje datoteka u strukturi
  - dodela i ukidanje prava pristupa nad direktorijumima i datotekama
    - konkretnim korisnicima ili ulogama na nivou OS
  - izmena nekih atributa datoteka

## Upravljanje fizičkom razmenom podataka

- Rutine za upravljanje fizičkom razmenom podataka
  - rutine fizičkog U/I (U/I supervizor, drajveri uređaja)
  - upravljanje razmenom blokova kroz U/I podsistem, između kontrolera eksternog memorijskog uređaja i OM
    - inicijalizacija fizičkog prenosa podataka jednog fizičkog bloka
      - zadavanje parametara prenosa
    - razmena poruka sa programima za fizički prenos podataka
      - koji se realizuje bilo kao CPU ili DMA prenos
    - prosleđivanje statusa uspešnosti obavljenog prenosa podataka

# Obezbeđenje veze programa i datoteke

- Upravljanje strukturama podataka o
  - (A) eksternim memorijskim uređajima
  - (B) datotekama na eksternom memorijskom uređaju
  - (C) upotrebi datoteka u aplikativnim programima
  - (D) datotekama u operativnoj upotrebi
- **Tabela OS**
  - struktura podataka za opis nekog resursa kojim OS operativno upravlja
  - gore navedene strukture podataka su, jednim delom, predstavljene putem tabela OS
- Tabele OS, važne za obezbeđenje veze programa i datoteke
  - tabela uređaja
  - sistemska tabela datoteke
  - alokaciona tabela datoteke
  - tabela logičkih imena datoteke
  - tabela procesa
  - tabela otvorenih datoteka
  - tabela opisa datoteke

## Podaci o eksternom memorijskom uređaju - jedinici diska (A)

- OS održava na jedinici diska strukture podataka o
  - proizvođačkim karakteristikama same disk jedinice
    - naziv, adresa i tip uređaja
    - ukupan broj cilindara, staza po cilindru i sektora po stazi
  - numeraciju sektora
  - ispravnim i neispravnim sektorima
  - zamenskim sektorima za neispravne sektore
  - definisanom kapacitetu bloka
  - korenskom direktorijumu i sistemu kataloga
  - slobodnom prostoru

### – Tabela uređaja (TU)

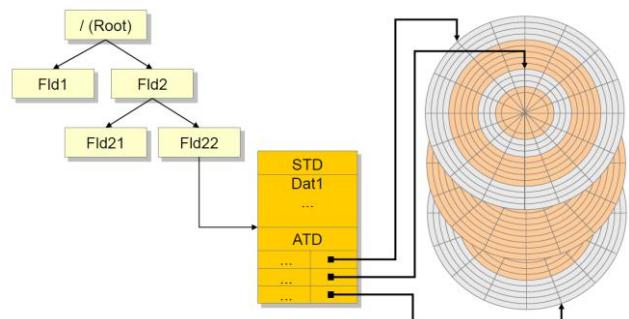
- zapis sa podacima, neophodnim za korišćenje uređaja
- formira se prilikom inicijalizacije ("start up-a") OS ili montiranja uređaja
- sadrži podatke o karakteristikama uređaja - disk jedinice
- sadrži adrese rutina za upravljanje fizičkim UI za dati uređaj (*device driver-a*)
- koriste je rutine za upravljanje fizičkim UI (drajveri)
- primer OS Unix
  - tabela drajvera uređaja
    - » svaki zapis odnosi se na jedan uređaj i sadrži adresu drajvera uređaja – reprezentuje deo TU
    - » podaci o karakteristikama uređaja nalaze se u drajveru

## Podaci o datotekama na eksternom memorijskom uređaju

- OS održava na jedinici diska strukture podataka o
  - datotekama, razmeštenim po sistemu kataloga
- **Sistemska tabela datoteke (STD)**
  - Index Node (Inode) Table (Unix)
  - predstavlja trajan zapis o datoteci koji održava OS
    - formira se prilikom kreiranja datoteke
    - uništava se prilikom brisanja datoteke
    - učitava se u OM, kada se datoteka operativno koristi
    - modifikuje se prilikom izmena sadržaja datoteke
  - spregnuta sa sistemom kataloga
    - adresirana iz pripadajućeg direktorijuma
  - sadržaj
    - naziv datoteke
    - ekstenzija i verzija datoteke
    - vrsta datoteke (bin/txt)
    - podaci o vlasniku i ovlašćenjima za korišćenje datoteke
    - veličina datoteke
    - datum i vreme kreiranja ili poslednje modifikacije sadržaja
    - podaci o ostalim atributima (sistemska, skrivena datoteka, itd.)
    - alokaciona tabela datoteke
  - konkretan sadržaj i struktura zavise od izabranog OS

### – Alokaciona tabela datoteke (ATD)

- mapa alociranog prostora diska za datoteku
- neprazan niz parova tipa (*pokazivač, broj blokova*)
  - *pokazivač*
    - » (c, t, s) adresa početka zone (granule) alociranog prostora
  - *broj blokova*
    - » ukazuje na veličinu zone alociranog prostora, isezanu brojem fizičkih blokova
- svako alociranje nove granule prostora izaziva formiranje novog para tipa (*pokazivač, broj blokova*) u nizu
- svako dealociranje nepotrebne granule prostora izaziva brisanje para tipa (*pokazivač, broj blokova*) iz niza
- predstavlja jedno od mogućih rešenja vođenja evidencije o alociranom prostoru datoteke



## Podaci o upotrebi datoteka u aplikativnim programima

- omogućavaju vezu između aplikativnih programa i OS
- formira ih kompjaler, a koristi i dopunjava OS
  - na osnovu specifikacija upotrebe datoteka u programu
- nalaze se u delu OM rezervisanom za aplikativni program
- sadržaj ovih podataka može zavisiti od
  - nivoa usluga OS (usluge niskog / visokog nivoa)
    - mogu biti uključeni podaci vezani za
      - » format sloga ili bloka datoteke
      - » željene načine upotrebe datoteke
      - » željene načine pristupa podacima datoteke

### – Tabela logičkih imena datoteka (TLI)

- File Descriptor Table (Unix)
- niz parova tipa (*ime datoteke, pokazivač*)
  - indeks niza – redni broj specificirane datoteke (fajl deskriptor)
    - » uobičajeno
      - 0 – standardna ulazna datoteka (pridružena tastaturi)
      - 1 – standardna izlazna datoteka (pridružena monitoru)
      - 2 – standardna datoteka grešaka (pridružena monitoru)
  - *ime datoteke*
    - » ukazuje na povezani naziv datoteke (iz STD)
  - *pokazivač*
    - » ukazuje na zapis sa podacima o otvorenoj datoteci

## Podaci o datotekama u operativnoj upotrebi

- formira ih i koristi OS
- kada je potrebno operativno korišćenje datoteke iz aplikativnog programa
  - otvaranje datoteke
    - priprema datoteke za operativno korišćenje podataka od strane procesa OS - aplikativnog programa
- obezbeđuju uvezivanje podataka, sadržanih u TU, STD i TLI
- nalaze se u sistemskom delu OM, kojim upravlja OS

### – Tabela otvorenih datoteka (TOD)

- niz zapisa o otvorenim datotekama u celom sistemu
  - svako otvaranje datoteke – jedan zapis u TOD
  - indeks niza – redni broj otvorene datoteke u sistemu
- svaki zapis sadrži podatke, neophodne da bi se obavljala razmena podataka između datoteke i aplikativnog programa

### – Sadržaj svakog zapisa u TOD

- mogući načini korišćenja datoteke
- pokazivač na tekuću poziciju u datoteci (tekući pokazivač)
- niz pokazivača prema rezervisanim sistemskim baferima
- veze između blokova i sistemskih bafera, u obliku
 

(rbr\_bloka, status, bafer)

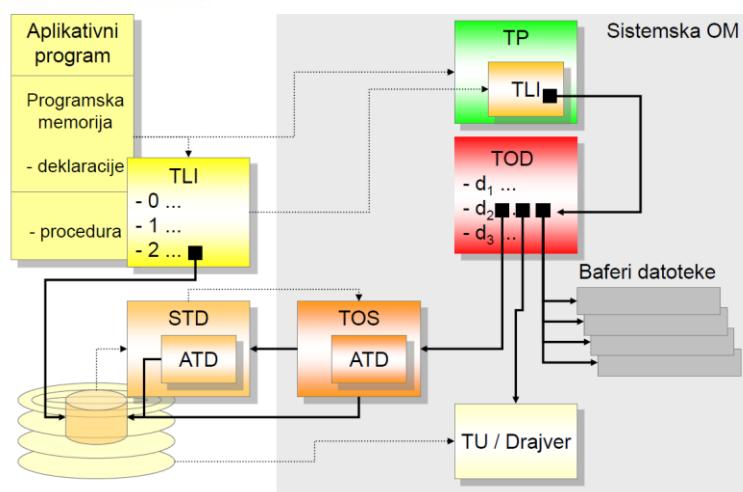
  - *rbr\_bloka* – redni broj učitanog bloka u sistemski bafer
  - *status* – indikator izmene sadržaja bloka nakon poslednjeg učitavanja
  - *bafer* – oznaka sistemskog bafera u koji je blok učitan
- pokazivač prema TU / drajveru uređaja u tabeli drajvera
- pokazivač na tabelu opisa datoteke (TOS)

### – Tabela procesa (TP)

- sadrži id oznaku procesa i neophodne podatke o procesu, kreiranom na osnovu aplikativnog programa
- uključuje podatke o trenutnom stanju procesa na nivou OS
- uključuje podatke iz TLI
  - tabela otvorenih datoteka procesa (TOP)
    - » TLI u sistemskom delu OM
    - » sa pokazivačima prema zapisima o otvorenim datotekama

### – Tabela opisa datoteke (TOS)

- Inode Table (Unix)
- predstavlja prekopirani sadržaj STD sa ATD u OM
- ažurira se tokom upotrebe datoteke
- prenosi se ažurirani sadržaj u STD, pri završetku rada s datotekom



# Sistemski pozivi

## Pozivi rutina OS za upravljanje datotekama

- pružaju usluge niskog nivoa, tj. obezbeđuju
  - pogled na datoteku kao niza bajtova
    - razmena podataka između aplikativnog programa i datoteke, organizovanih kao nizovi bajtova
  - upravljanje blokovima datoteke
    - grupisanje nizova bajtova u blokove
    - razmena kompletnih blokova između eksternog memorijskog uređaja i sistemskih bafera u OM
  - upravljanje sistemskim baferima
    - nalozi za rezervisanje i otpuštanje bafera
    - evidencija smeštanja blokova u bafere
  - nezavisnost aplikativnog programa od fizičkih karakteristika eksternog memorijskog uređaja
    - transformacija rednog broja bajta u redni broj bloka i rednog broja bloka u absolutnu adresu bloka na disku
- vode računa o karakteristikama datoteke
  - **početak datoteke**
    - označen pozicijom bajta s rednim brojem 1 (ili alternativno 0)
  - **kraj datoteke**
    - označen pozicijom poslednjeg bajta, uvećanom za 1 (ili alternativno samo pozicijom poslednjeg bajta)
  - **indikator tekuće pozicije (tekući pokazivač, indikator aktuelnosti)**
    - iskazan kao redni broj bajta na kojem započinje operacija (ili alternativno, na kojem je završila prethodna operacija)
- podržavaju **sekvenčalni (redosledni) pristup** bajtovima datoteke
  - pri operacijama učitavanja / zapisivanja
    - zahtevaju zadavanje ukupnog broja bajtova za operaciju
    - automatski održavaju vrednost tekućeg pokazivača
- podržavaju **direktni pristup** bajtovima datoteke
  - pri operacijama pozicioniranja
    - zahtevaju zadavanje vrednosti tekućeg pokazivača
      - » rednog broja bajta u odnosu na koji započinje sledeća operacija
- preuzimaju parametre poziva iz pozivajućeg okruženja
  - aplikativnog programa ili
  - okruženja, zaduženog za pružanje usluga visokog nivoa
- prosleđuju u pozivajuće okruženje informacije o statusu izvršenja sistemskog poziva
  - osnova za obradu izuzetaka
  - **izuzetak (exception)**
    - događaj koji izaziva prekid normalnog toka obrade podataka
    - neki primeri
      - » greška pri učitavanju ili zapisivanju podataka
      - » pokušaj otvaranja nepostojeće ili već otvorene datoteke
      - » pokušaj čitanja nepostojećih bajtova (preko kraja datoteke)

- **Tipovi sistemskih poziva (Unix)**

– create	- kreiranje datoteke
– open	- otvaranje datoteke
– read	- učitavanje dela sadržaja datoteke
– write	- zapisivanje podataka u datoteku
– seek	- pozicioniranje na željenu lokaciju
– close	- zatvaranje datoteke
– sync	- pražnjenje izmenjenih bafera
– delete	- brisanje i uništavanje datoteke
– truncate	- brisanje sadržaja datoteke
– stat	- preuzimanje informacija o datoteci

## Create

- sistemski poziv za kreiranje datoteke
  - na osnovu zadatih parametara
    - naziv sa putanjom, ovlašćenja, tip, veličina, itd. – zavisi od OS
  - Unix:

```
int creat(char *naziv, int ovlašćenja)
```
- kreiranje potpuno nove datoteke i otvaranje za pisanje
  - proverava sve uslove, neophodne za kreiranje datoteke
    - postojanje volumena, putanje, raspoloživog prostora, prava
  - alocira inicijalni prostor za novu datoteku na disku
  - kreira STD sa ATD
  - formira zapis u direktorijumu i uvezuje ga sa STD
  - vraća podatak o uspešnosti operacije i fajl deskriptor

# Open

- sistemski poziv za otvaranje datoteke
  - na osnovu zadatih parametara
    - naziv sa putanjom, način otvaranja, eventualno i ovlašćenja
  - Unix:

```
int open(char *naziv, int način_otvar [int ovlašćenja])
```
- priprema datoteke za operativnu upotrebu
  - u operativnoj upotrebi datoteke zahteva se pristup podacima u STD i TLI
  - pogodno je smestiti te potrebne podatke u OM
    - da bi se izbeglo repetitivno pristupanje disku i višestruko obavljanje istih pripremnih operacija, za potrebe svake pojedinačne R/W operacije
- načini otvaranja datoteke
  - otvaranje nepostojeće datoteke
    - zahteva se inicialno kreiranje datoteke (O\_CREAT)
  - otvaranje postojeće datoteke
  - otvaranje u režimu dozvole čitanja i pisanja podataka na željenoj poziciji (O\_RDWR)
  - otvaranje u režimu ekskluzivnog čitanja sadržaja (O\_RDONLY)
  - otvaranje u režimu ekskluzivnog pisanja sadržaja
    - sa dodavanjem novog sadržaja na kraj datoteke (O\_APPEND)
    - sa prepisivanjem postojećeg sadržaja datoteke (O\_WRONLY)
    - sa prethodnim brisanjem postojećeg sadržaja datoteke (O\_TRUNC)

- **Upravljanje sistemskim baferima**
  - rezervisanje i otpuštanje bafera datoteke
    - vrši se korišćenjem servisa jezgra OS
    - otvorenoj datoteci mora biti dodeljen najmanje jedan bafer
  - evidentiranje smeštanja blokova u bafere
    - u zapisu iz TOD, ili u referenciranoj strukturi podataka koju održava jezgro OS
- **Tehnike rezervisanja i otpuštanja bafera**
  - fiksna (statička) dodela bafera datoteci
  - dinamička dodela bafera datoteci (udruživanje)
    - buffer pooling

## Višestruko otvaranje iste datoteke

- nije dozvoljeno kod nekih OS
  - datoteka se ekskluzivno otvara i zaključava od strane jednog procesa
  - za svaku datoteku, u TOD može postojati najviše jedan zapis
  - ostali procesi mogu, eventualno, samo otvoriti "read-only" kopiju datoteke
    - za potrebe uvida u trenutno stanje datoteke
  - očuvana je konzistentnost podataka u višekorisničkom režimu rada
  - bitno snižen stepen mogućeg paralelizma u obradi i korišćenju podataka datoteke
- zadaci
  - ukoliko je specificiran zahtev kreiranja nove datoteke, sprovođenje poziva tipa Create
  - u slučaju zahteva za otvaranje postojeće datoteke, provera ostvarenosti potrebnih uslova
    - postojanja traženog volumena, specificirane putanje sa direktorijumom i traženim fizičkim nazivom datoteke
    - ovlašćenja za izvođenje zahtevane operacije tipa Open
  - prenos sadržaja STD u OM
    - kreiranje TOS i uvezivanje sa STD
  - formiranje zapisa u TOD i uvezivanje sa TOS i TLI u TP
    - povezivanje fizičkog imena i logičke oznake datoteke u TLI
    - povezivanje adresa odgovarajućih programa za opsluživanje sistemskih poziva za razmenu podataka
  - inicijalizacija vrednosti tekućeg pokazivača
    - na početak datoteke, ili na kraj (u slučaju O\_APPEND)
  - eventualno, inicijalno rezervisanje sistemskih bafera i početno punjenje bafera
    - upisivanje pokazivača na bafere u odgovarajući zapis u TOD
    - punjenje bafera početnim blokovima datoteke
      - » u slučaju otvaranja datoteke za čitanje
  - predaja pozivajućem okruženju podataka
    - o uspešnosti izvedene operacije i
    - fajl deskriptora (logičke oznake datoteke)

## • Tehnike rezervisanja i otpuštanja bafera

- **Fiksna (statička) dodela bafera datoteci**
  - fiksni broj bafera ( $\geq 1$ ) ekskluzivno se dodeljuje pri otvaranju
  - svi baferi se otpuštaju pri zatvaranju datoteke
    - često, datoteci samo za čitanje / pisanje dodeljuje se 1 bafer, a
    - datoteci otvorenoj i za čitanje i za pisanje 2 bafera
      - » jedan za operacije čitanja i drugi za operacije pisanja
- **Dinamička dodela bafera datoteci (udruživanje)**
  - svi sistemski baferi OS-a su na raspolaganju svim otvorenim datotekama
  - baferi se dodeljuju i otpuštaju dinamički, prema potrebi
    - pri realizaciji operacija čitanja ili pisanja podataka
  - kriterijum izbora slobodnog bafera za dodelu, ili izbora bafera za otpuštanje
    - najduže nekorišćeni bafer (Least Recently Used - LRU)

## – dozvoljeno je kod nekih OS

- primer Unix
- datoteka se može otvoriti
  - od strane više različitih procesa istovremeno (open) ili
  - od strane jednog procesa multiplikovano
    - » sistemski pozivi dup i dup2
    - » formira se barem dva zapisa sa fajl deskriptorima u TLI
- svako otvaranje datoteke formira poseban zapis u TOD
  - koji je uvezan sa odgovarajućim zapisom u TLI iz TP
  - precizno se evidentira koji proces je izvršio koje otvaranje
- konzistentnost podataka u višekorisničkom režimu rada ne mora biti očuvana
  - postoji tehnika "savetodavnog" zaključavanja dela podataka
- visok stepen mogućeg paralelizma u obradi podataka

## Read

- sistemski poziv za učitavanje niza bajtova iz otvorene datoteke

- na osnovu zadatih parametara
  - logička oznaka datoteke, odredišna promenljiva u OM, broj bajtova za učitavanje

- Unix:

```
int read(int fd, void *var, int size)
```

- stvarni prenos dela sadržaja datoteke u pozivajuće okruženje

- zadati broj bajtova, od pozicije tekućeg pokazivača

### – zadaci

- provera da li datoteka sa oznakom *fd*
  - je otvorena
  - dozvoljava zahtevani (read) način pristupa
- izračunavanje rednog broja prvog bloka i dužine niza izvornih blokova, u kojima se nalaze traženi podaci
  - na osnovu rednog broja bajta u tekućem pokazivaču
  - pretvaranje rednog broja bajta tekućeg pokazivača u par (*redni broj prvog bloka u nizu, redni broj bajta u bloku*)

- provera da li se traženi blokovi već nalaze u baferima
- ako ne
  - izbor (ili alokacija) sistemskih bafera za smeštanje blokova
    - » uz eventualno oslobođanje prethodnog sadržaja bafera
  - izračunavanje apsolutnih adresa traženih blokova
  - inicijalizacija fizičkog prenosa blokova sa jedinice diska
  - obrada statusa uspešnosti fizičkog prenosa blokova
- prenos traženog sadržaja iz bafera u ciljnu promenljivu *var*
- uvećavanje vrednosti tekućeg pokazivača
  - na prvi sledeći bajt, nakon poslednje prenetog bajta
- predaja pozivajućem okruženju podataka
  - o uspešnosti izvedene operacije i
  - broju stvarno preuzetih bajtova

## Write

- sistemski poziv za zapisivanje niza bajtova u otvorenu datoteku

- na osnovu zadatih parametara
  - logička oznaka datoteke, izvorna promenljiva u OM, broj bajtova za zapisivanje

- Unix:

```
int write(int fd, void *var, int size)
```

- stvarni prenos niza bajtova iz pozivajućeg okruženja u datoteku

- zadati broj bajtova, od pozicije tekućeg pokazivača

### – zadaci

- provera da li je potrebno imati dopremljen prethodni sadržaj ciljnih blokova
  - u O\_RDWR načinu otvaranja, tipično jeste
    - » ne mora se pozivom zahtevati pisanje celokupnog sadržaja bloka
  - u O\_APPEND ili O\_WRONLY načinu otvaranja, nije
- ako da, provera da li se traženi blokovi već nalaze u baferima
  - ako ne
    - » izbor (ili alokacija) sistemskih bafera za smeštanje blokova
    - » uz eventualno oslobođanje prethodnog sadržaja bafera
    - » izračunavanje apsolutnih adresa traženih blokova
    - » inicijalizacija fizičkog prenosa blokova sa jedinice diska
    - » obrada statusa uspešnosti fizičkog prenosa blokova

### – zadaci

- provera da li datoteka sa oznakom *fd*
  - je otvorena
  - dozvoljava zahtevani (write) način pristupa
- izračunavanje rednog broja prvog bloka i dužine niza ciljnih blokova, u koje treba smestiti izvorne podatke
  - na osnovu rednog broja bajta u tekućem pokazivaču
  - pretvaranje rednog broja bajta tekućeg pokazivača u par (*redni broj prvog bloka u nizu, redni broj bajta u bloku*)

### – zadaci

- prenos traženog sadržaja iz izvorne promenljive *var* u bafere
  - uz eventualno oslobođanje sadržaja kompletno napunjениh bafera
    - » iniciranje fizičkog prenosa blokova na jedinicu diska
    - » buffer flushing
- uvećavanje vrednosti tekućeg pokazivača
  - na prvi sledeći bajt, nakon poslednje prenetog bajta
- predaja pozivajućem okruženju podataka
  - o uspešnosti izvedene operacije i
  - broju stvarno predatih bajtova

## Seek

- pozicioniranje (pristup) na željenu lokaciju
  - na osnovu zadatih parametara
    - logička oznaka datoteke, pomak, referentna tačka za pomak
  - Unix:  
`int lseek(int fd, int offset, int moveDirection)`
- upravljanje sadržajem tekućeg pokazivača
  - u cilju obezbeđenja direktnog pristupa željenom bajtu datoteke

### – zadaci

- provera da li je datoteka sa oznakom *fd* otvorena
- postavljanje nove vrednosti tekućeg pokazivača
  - pomeranjem za zadati broj bajtova *offset*
    - » prema kraju datoteke, ako je *offset* > 0
    - » prema početku datoteke, ako je *offset* < 0
    - » bez pomeranja, ako je *offset* = 0
  - u odnosu na referentnu tačku, zadatu sa *moveDirection*
    - » 0 (SEEK\_SET)                    – za početak datoteke
    - » 1 (SEEK\_CURR)                – za trenutnu poziciju pokazivača
    - » 2 (SEEK\_END)                    – za kraj datoteke
- predaja pozivajućem okruženju podatka o uspešnosti izvedene operacije

## Close

- sistemski poziv za zatvaranje otvorene datoteke
  - na osnovu zadate logičke oznake datoteke
  - Unix:  
`int close(int fd)`
- uredan prestanak operativne upotrebe datoteke
  - u opštem slučaju, garantuje da će datoteka ostati memorisana na disku u konzistentnom stanju
- zatvaranje datoteke nastaje
  - automatski, završetkom programa koji je otvorio datoteku
  - eksplicitno, pokretanjem sistemskog poziva Close

### – zadaci

- oslobođanje sadržaja zauzetih bafera, modifikovanog sadržaja
  - **buffer flushing**
  - **dirty buffer**
    - » bafer u kojem je izmenjeno ili potpuno novo stanje bloka u odnosu na njegov sadržaj na disku
    - iniciranje fizičkog prenosa blokova na jedinicu diska
- oslobođanje svih zauzetih bafera i vraćanje OS-u
  - ova i prethodna aktivnost nisu obavezne kod svih OS
    - » Unix close ne podržava pražnjenje i oslobođanje bafera pri zatvaranju datoteke
  - prenos sadržaja TOS sa ATD na jedinicu diska
    - ažuriranje sadržaja STD sa ATD
  - uništavanje TOS i odgovarajućeg zapisa u TOD
    - raskidanje svih uspostavljenih veza prema TLI, TP i STD
- predaja pozivajućem okruženju podatka o uspešnosti izvedene operacije

## Sync

- sistemski poziv za "pražnjenje" dirty bafera
  - Unix:
  - `void sync(void)`
  - izvršava se na zahtev, ili automatski u zadatim intervalima

## Delete

- sistemski poziv za brisanje imena i uništavanje datoteke
  - na osnovu zadatog naziva sa putanjom
  - Unix – brisanje imena fajla (tzv. linka)  
`int unlink(char *naziv)`
    - može u strukturi kataloga biti kreirano više imena - linkova
    - brisanje poslednjeg linka briše i uništava datoteku
- zadaci fizičkog brisanja i uništavanja datoteke
  - dealociranje prostora datoteke na disku
  - uništavanje STD sa ATD
  - uništavanje zapisa o datoteci u direktorijumu
  - vraćanje podatka o uspešnosti operacije

## Truncate

- sistemski poziv za brisanje sadržaja datoteke
  - dealocira prostor datoteke na disku, ali zadržava STD
- `int truncate(const char *path, off_t length)`
- `int ftruncate(int fd, off_t length)`
- "odsecanje" sadržaja datoteke do na zadati broj bajtova
- briše sadržaj datoteke i dealocira oslobođeni prostor datoteke na disku
- uvek zadržava STD

## Metode pristupa

- Podržavaju usluge visokog nivoa
  - koriste ili uključuju usluge niskog nivoa izabranog OS
  - obezbeđuju poglede na LSP datoteke
    - kao različitih struktura nad skupom slogova
      - a obavezno kao niza slogova
    - kao različitih struktura nad skupom blokova
      - a obavezno kao niza blokova
  - obezbeđuju preslikavanje ovih pogleda u FSP niza fizičkih blokova
  - obezbeđuju izgradnju specijalnih pomoćnih struktura za poboljšanje efikasnosti obrade podataka
  - obezbeđuju traženja, zasnovana na vrednostima podataka
- Zahtevaju razrešavanje pitanja
  - organizovanja i memorisanja polja, slogova i blokova
  - načina adresiranja i načina memorisanja logičkih veza
  - mogućih vrsta usluga na nivou sloga ili bloka
  - podrške različitih vrsta organizacije datoteka
  - podrške opštih postupaka upravljanja sadržajem datoteka
- Servisi metoda pristupa mogu biti ugrađeni u
  - **operativni sistem**
  - **programski jezik sa pridruženim paketima (bibliotekama) funkcija**
  - **sistem za upravljanje bazama podataka**