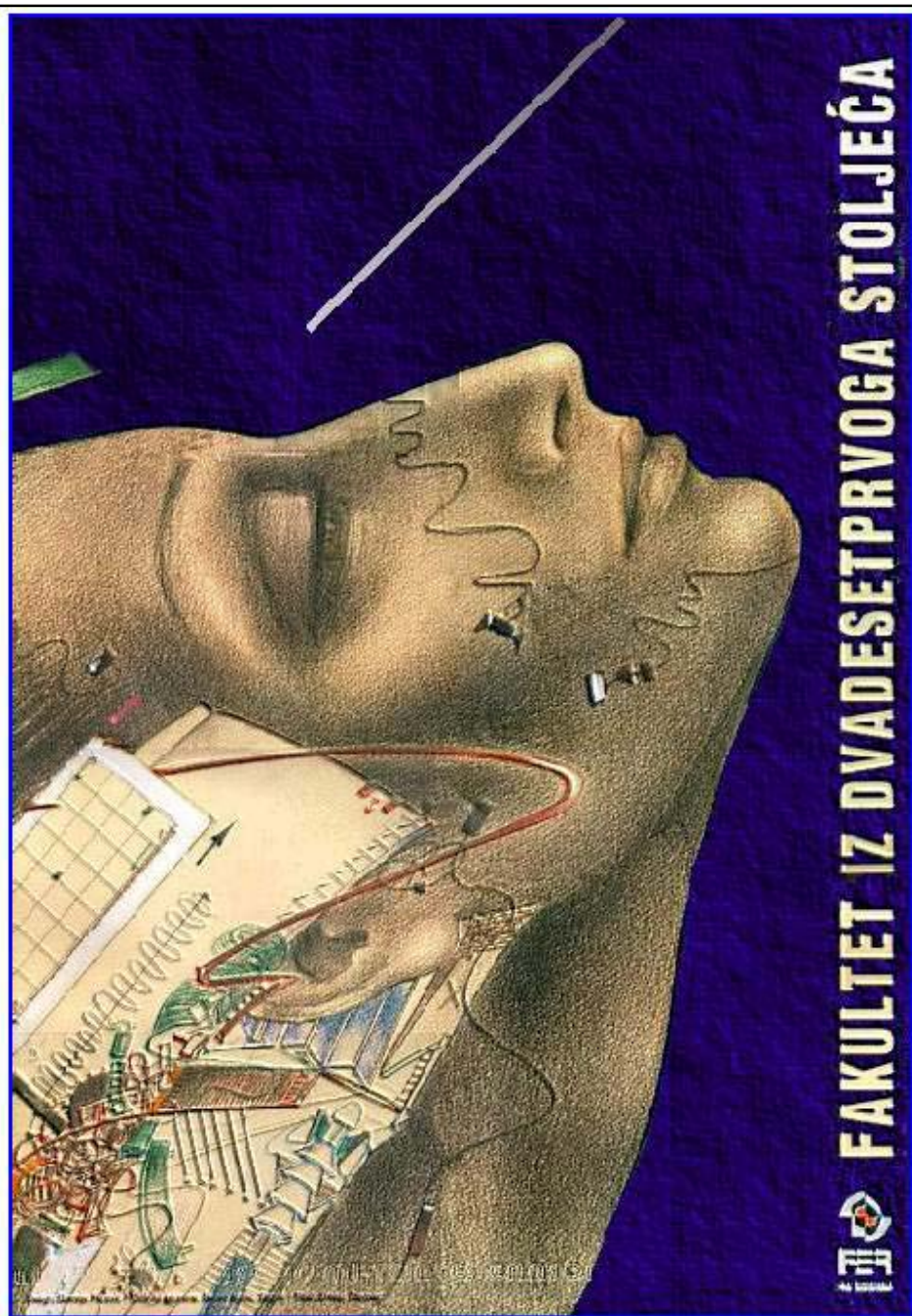


Baze podataka

Predavanja
travanj 2008.

9. Fizička organizacija podataka



UVOD - Fizička organizacija

- Pojam fizičke organizacije podataka odnosi se na:
 - strukture podataka primijenjene pri pohrani podataka u sekundarnoj memoriji
 - metode pristupa (*access methods*): postupci koji se primjenjuju pri obavljanje operacija nad podacima
- Fizička organizacija podataka ne utječe na rezultate operacija s podacima, ali ima vrlo veliki utjecaj na učinkovitost sustava za upravljanje bazama podataka
 - važna zadaća sustava za upravljanje bazama podataka: obavljati operacije nad velikim količinama podataka **na učinkovit način**
- SUBP skriva od korisnika detalje fizičke organizacije podataka jer za većinu korisnika sustava nisu značajni

Pohrana baze podataka u sekundarnoj memoriji

- Glavna memorija (radni spremnik, *main memory*)
 - velike brzina pristupa podacima (10-100 ns), relativno skupa, kapacitet ~ GB
 - neprikladna za pohranu baze podataka jer:
 - ima kapacitet nedovoljan za pohranu baze podataka (previsoka cijena za pohranu velikih količina podataka)
 - nepostojana (*volatile*) memorija: sadržaj memorije se gubi pri gubitku napajanja ili pri pogrešci sustava

Pohrana baze podataka u sekundarnoj memoriji

- karakteristike medija za pohranu podataka uvjetuju da se većina današnjih baza podataka pohranjuje u **sekundarnoj memoriji** (uobičajeno: magnetski diskovi)
 - podaci se između sekundarne i primarne memorije prenose u blokovima (tipično 512 B, 1 kB, 2kB, 4kB)
- ⇒ **dominiraju troškovi UI (ulazno/izlaznih) operacija:** vrijeme potrebno za obavljanje UI operacije radi prijenosa bloka podataka između sekundarne i primarne memorije znatno je veće od vremena koje će biti utrošeno za obavljanje operacija nad podacima u primarnoj memoriji

Važniji ciljevi fizičke organizacije

- minimizirati broj UI operacija pri pohrani i dohvat u podataka, minimizirati utrošak prostora za pohranu
 - u koji fizički blok pohraniti logički zapis odnosno n-torku
 - koje je dodatne informacije potrebno pohraniti da bi se omogućio učinkovit pristup podacima
- omogućiti različite metode pristupa koje se koriste za pronalaženje fizičke pozicije zapisa (ili fizičke pozicije bloka u kojem se taj zapis nalazi) na temelju vrijednosti ključa pretrage
 - ključ pretrage (*search key*) ne mora nužno biti primarni ili alternativni ključ. Ključ pretrage može biti bilo koji atribut ili skup atributa relacije ("sekundarni ključ").
 - primjenjivost pojedinih metoda pristupa podacima ovisi o primijenjenim strukturama podataka

Strukture podataka i metode pristupa podacima

- Primjena različitih struktura podataka omogućava različite metode pristupa podacima
- Ne postoji "najbolja" metoda fizičke organizacije, ali dvije se u današnjim sustavima za upravljanje bazama podataka koriste najčešće
 - **Neporedana (*heap*) datoteka**
 - Poredana datoteka (*sorted file*)
 - Raspršena datoteka (*hash file*)
 - Indeksno-slijedna organizacija (*index-sequential file*)
 - **B-stablo (*B-tree*)**

Neporedana (*heap*) datoteka

- zapis se upisuje na bilo koje slobodno mjesto u datoteci
- pristup podacima (dohvat podatka sa zadanom vrijednošću ključa pretrage) moguć je isključivo linearnim pretraživanjem



15	Petra
1	Marko
47	Ivana
2	Janko
5	Ana
	...

- koristi se za relacije s malim brojem n-torki ili u relacijama čiji se podaci uvijek obrađuju slijedno

Neporedana (*heap*) datoteka

- Dohvat zapisa prema ključu pretrage
 - prema primarnom ili alternativnom ključu
 - u prosjeku je potrebno obaviti $n/2$ UI operacija (n predstavlja broj fizičkih blokova u kojima su pohranjeni logički zapisi odnosno n -torke)
 - još gore: u slučaju kada traženi zapis ne postoji, sustav će morati obaviti n UI operacija
 - prema ostalim ključevima pretrage ili prema zadanim granicama intervala
 - potrebno je obaviti prijenos n fizičkih blokova

B-Stabla

Literatura:

1. **Silberschatz, Korth, Sudarshan**: Database System Concepts
2. **Garcia-Molina, Ullman, Widom**: Database Systems -The Complete Book

Ovdje će biti opisana varijanta B-stabla koja se naziva **B⁺- stablo**. Opisi ostalih varijanti B-stabala (B*-stablo, B-stablo, ...) mogu se pronaći u literaturi.

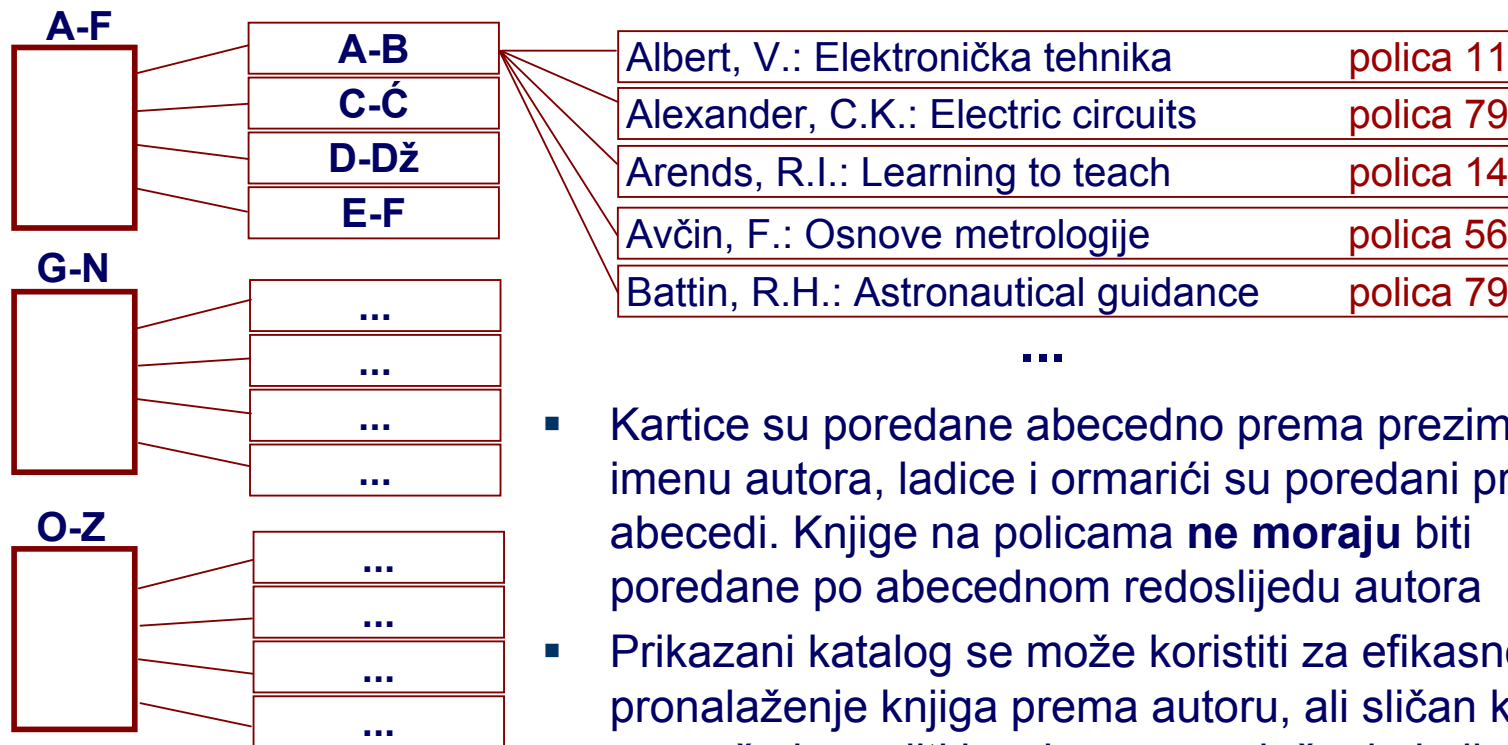
B-stabla

- Ideja se temelji na izgradnji indeksnog kazala na više razina: slično kao kod kataloga u papirnatom obliku u knjižnicima (danas se rijetko koriste)

ormarići

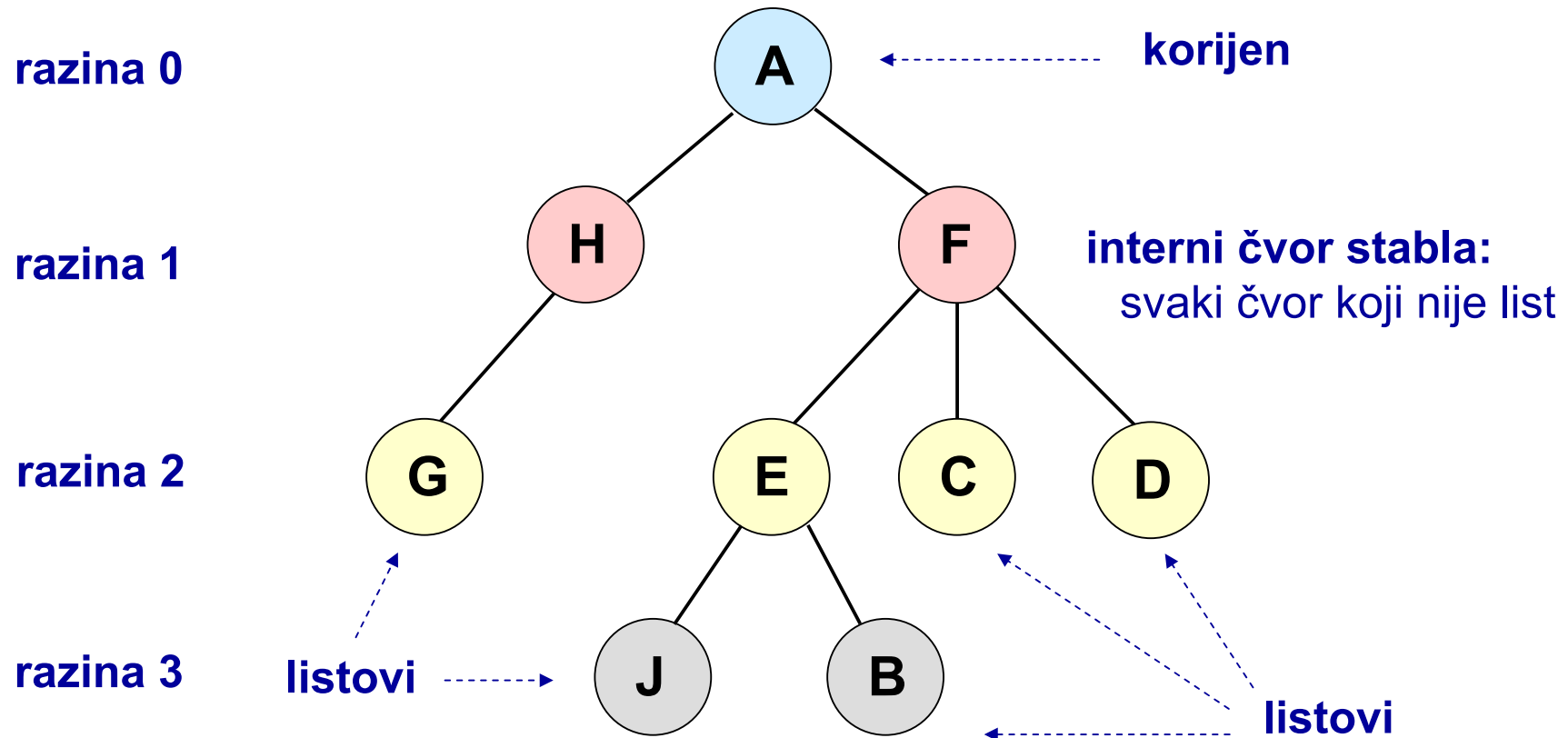
ladice

kartice



- Kartice su poredane abecedno prema prezimenu i imenu autora, ladice i ormarići su poredani prema abecedi. Knjige na policama **ne moraju** biti poredane po abecednom redoslijedu autora
- Prikazani katalog se može koristiti za efikasno pronalaženje knjiga prema autoru, ali sličan katalog se može izgraditi i za brzo pronalaženje knjiga prema naslovu ili prema nekim drugim pojmovima.

Stablo kao struktura podataka

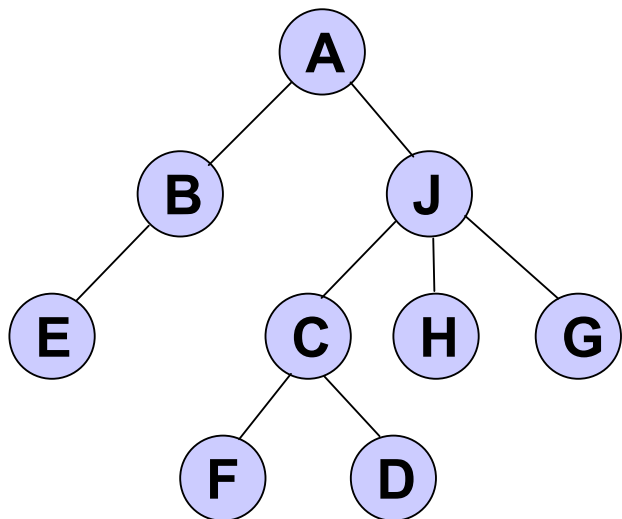


- **razina čvora (*level*):** duljina puta od korijena do čvora
- **dubina stabla (*depth*):** najveća duljina puta od korijena do lista
- **red stabla (*order*):** najveći broj djece koje čvor može imati

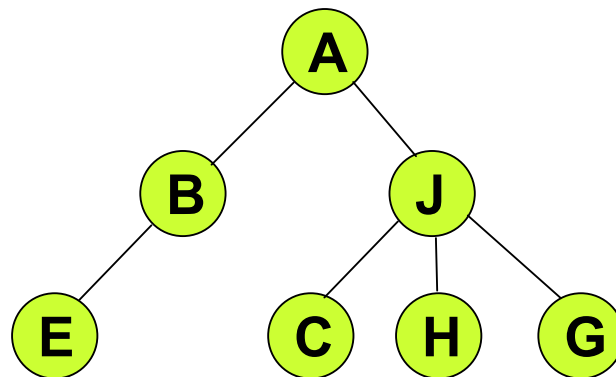
Stablo kao struktura podataka

Stablo je **balansirano** (*balanced*) ukoliko je duljina puta od korijena do lista jednaka za svaki list u stablu

stablo nije balansirano

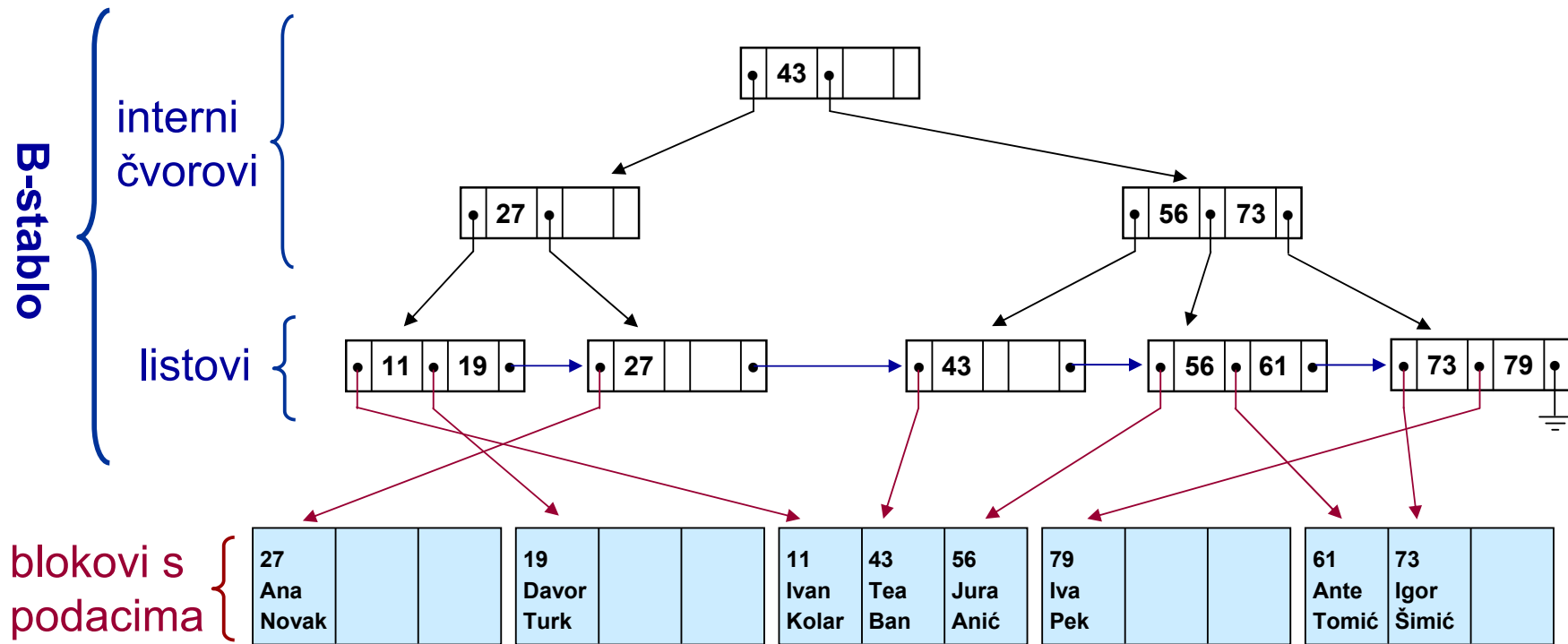


stablo je balansirano



Oznaka **B** u B-stablo znači "**balansirano**"!

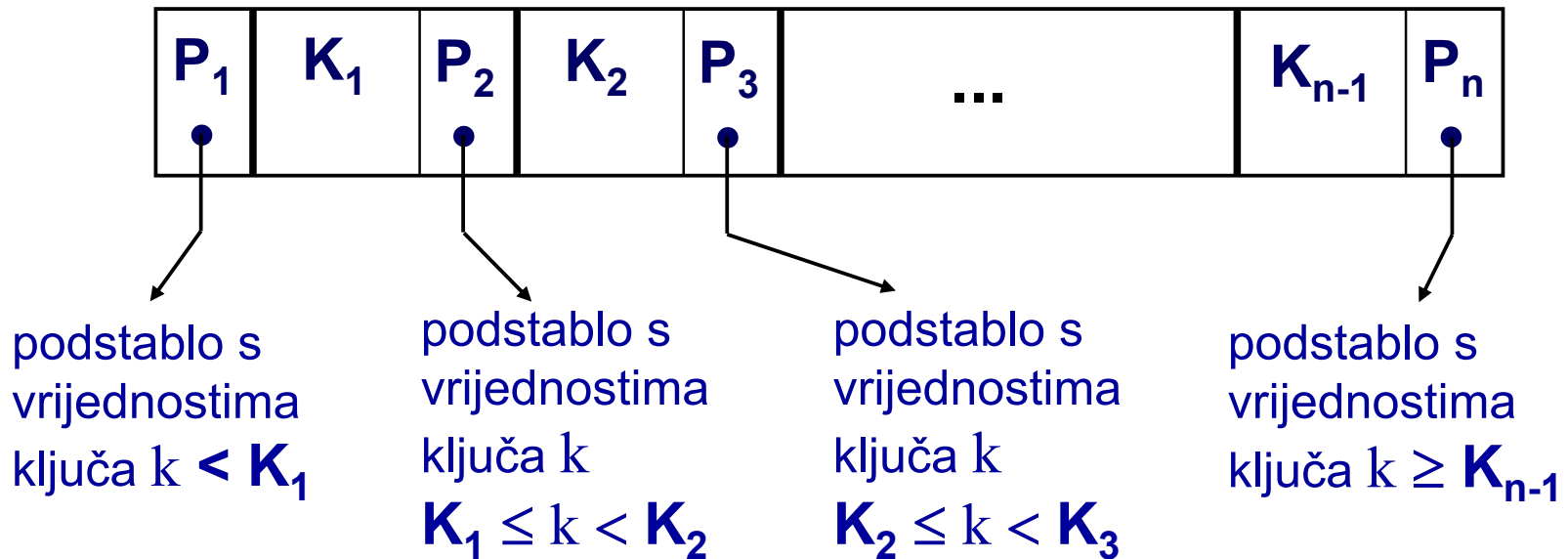
Struktura B⁺-stabla



- Shema: mbr, ime, prezime; B⁺-stablo je izgrađeno za atribut mbr
- Moguće metode pristupa podacima (za bilo koji ključ pretrage):
 - linearnim pretraživanjem (kao kod neporedane datoteke)
 - ako je ključ pretrage mbr, može se koristiti B-stablo

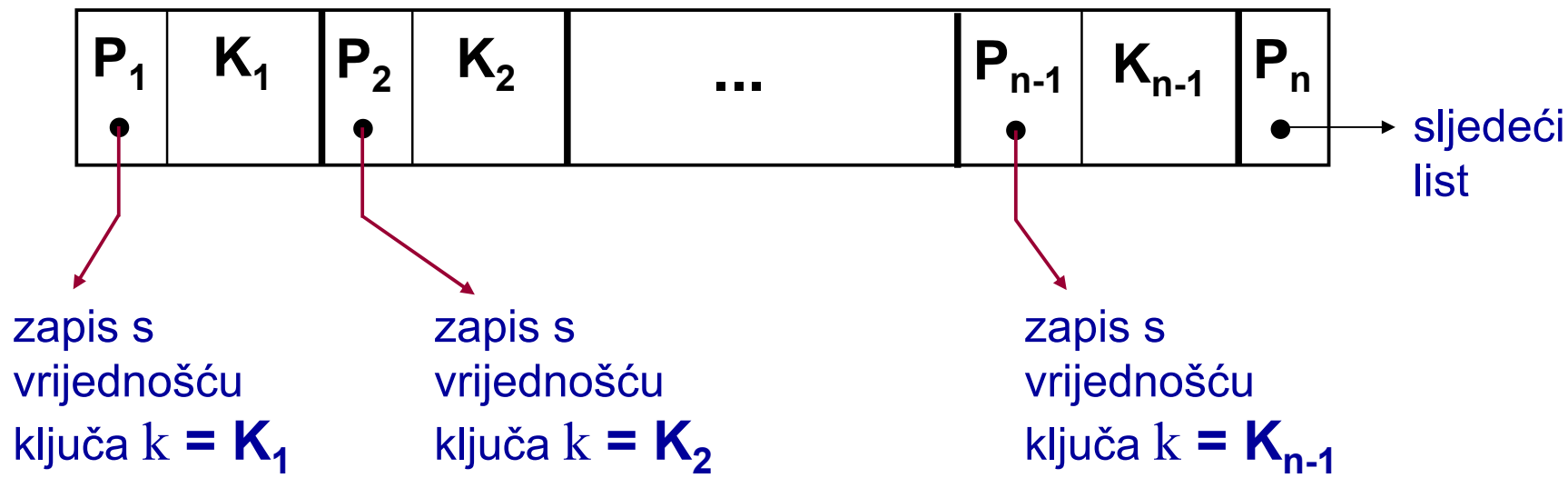
Struktura internog čvora B⁺-stabla

- U B⁺-stablu reda **n**, **interni čvor** sadrži:
 - najviše **n** kazaljki
 - najmanje $\lceil n/2 \rceil$ kazaljki → $\lceil a \rceil$ je najmanji cijeli broj $\geq a$
 - ovo ograničenje ne vrijedi za korijen (najmanji broj kazaljki je 2)
 - uz **p** kazaljki u čvoru, broj pripadnih vrijednosti K_i u čvoru je **p-1**
 - K_i je vrijednost ključa



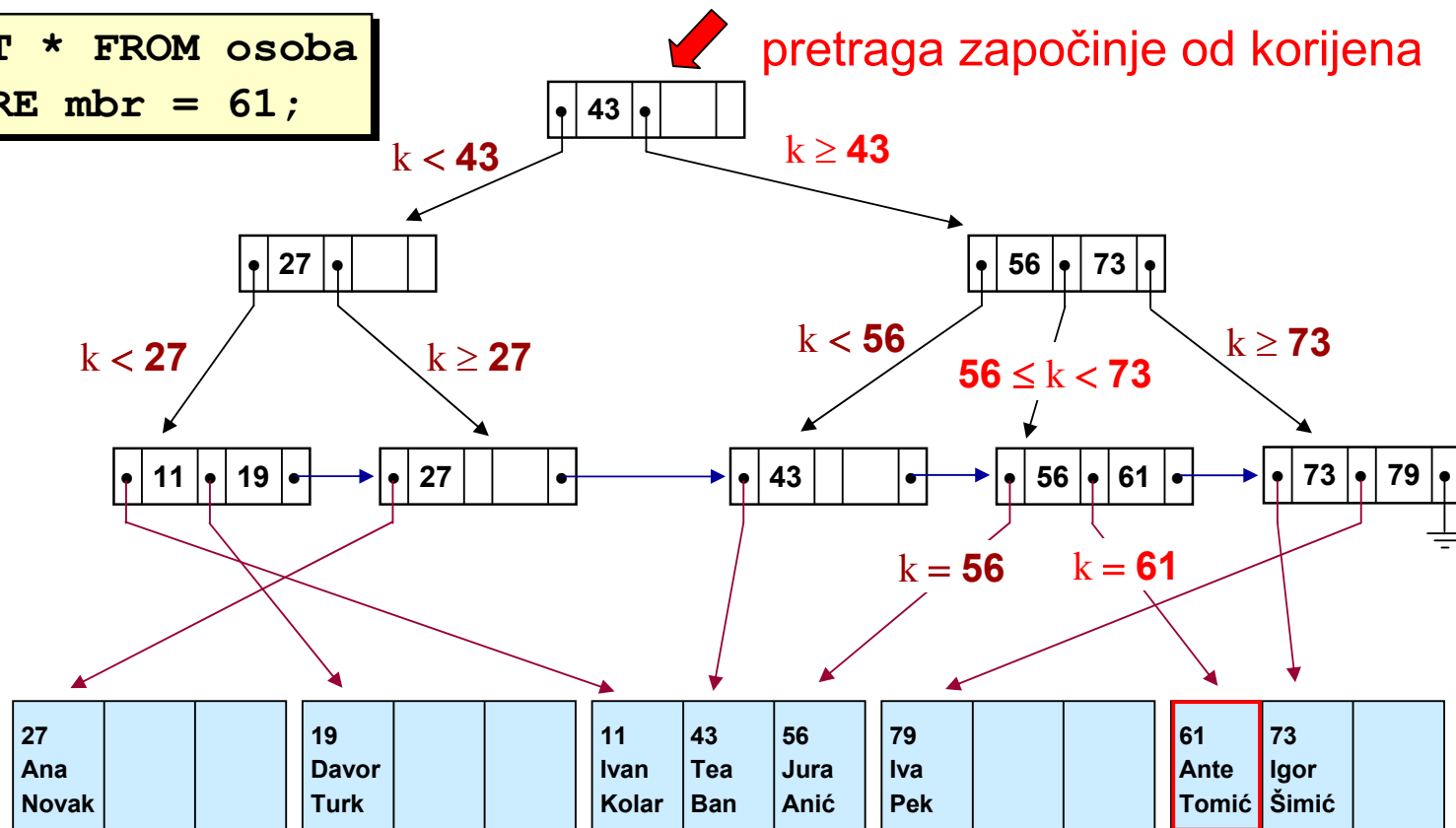
Struktura lista B⁺-stabla

- U B⁺-stablu reda **n**, **list** sadrži:
 - najviše **n-1** vrijednosti K_i i pripadnih kazaljki na zapise
 - najmanje $\lceil (n-1)/2 \rceil$ vrijednosti K_i i pripadnih kazaljki na zapise
 - svi listovi sadrže kazaljku na sljedeći list
 - omogućava upite tipa od-do (prema zadanim granicama intervala)



Algoritam za pronalaženje zapisa putem B⁺-stabla

```
SELECT * FROM osoba  
WHERE mbr = 61;
```

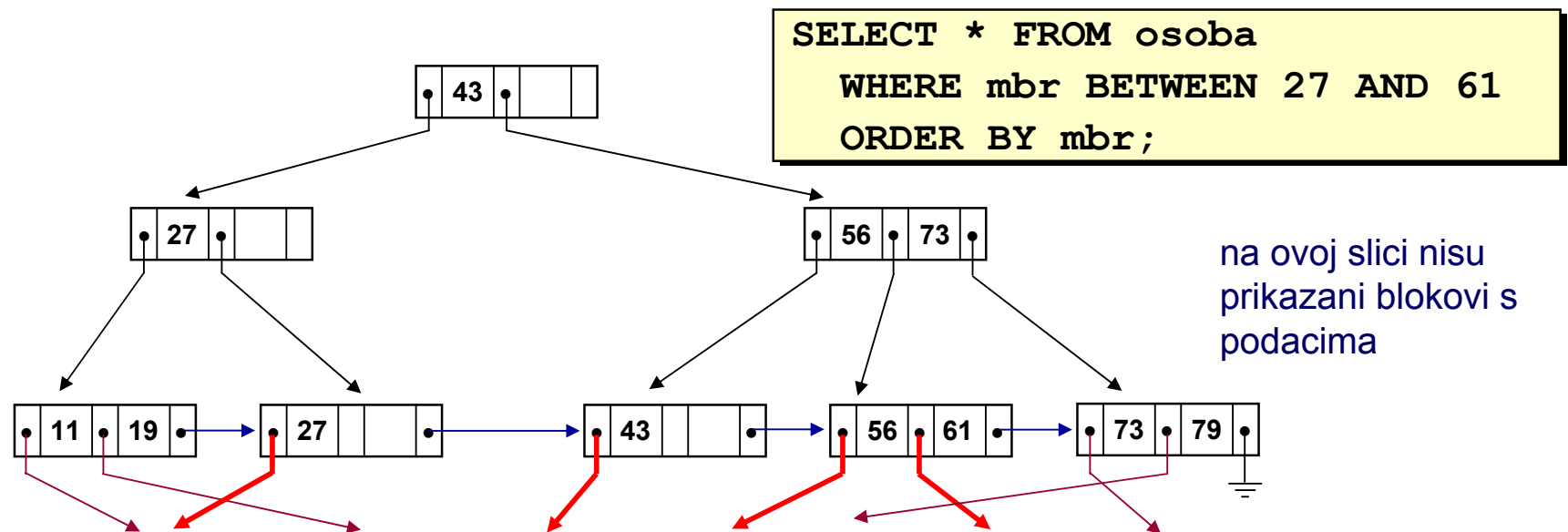


- slijediti odgovarajuću kazaljku do sljedeće razine
- postupak se ponavlja dok se ne dođe do lista u kojem će se naći kazaljka na zapis u bloku s podacima

Algoritam za pronalaženje zapisa putem B⁺-stabla

- **algoritam za traženje zapisa s ključem vrijednosti k je rekurzivan**
 - cilj je u svakom koraku rekurzije (pretraga i -te razine) pronaći čvor na nižoj, $(i+1)$ -voj razini, koji će voditi prema listu u kojem se nalazi ključ čija je vrijednost k
- **traženje zapisa započinje od korijena (0-te razine)**
- u čvoru i -te razine potrebno je pronaći najveću vrijednost ključa koja je manja ili jednaka traženoj vrijednosti k
 - za prvu kazaljku internog čvora nije navedena vrijednost ključa, pa ona "pokriva" sve vrijednosti ključeva manje od prve vrijednosti ključa (K_1) navedene u čvoru
- nakon pronalaženja odgovarajuće vrijednosti ključa, slijedi se pripadna kazaljka i time se obavlja pozicioniranje na $(i+1)$ -vu razinu
- postupak se ponavlja rekurzivno sve dok se ne dođe do lista. U njemu se mora nalaziti, ukoliko postoji, ključ čija je vrijednost k , te pripadna kazaljka prema traženom zapisu (n -torki)

Dohvat podataka iz intervala, sortiranje



- u listu pronaći kazaljku na zapis s ključem **27**
- redom dohvaćati kazaljke i pripadne zapise dok se ne dođe do kazaljke na zapis s ključem **61**
 - taj postupak omogućavaju kazaljke među listovima
- dobiveni su svi traženi zapisi, pri tome su poredani prema mbr
- Ako se obavlja **SELECT * ... ORDER BY mbr DESC**
 - pronađene zapise jednostavno ispisati obrnutim redoslijedom

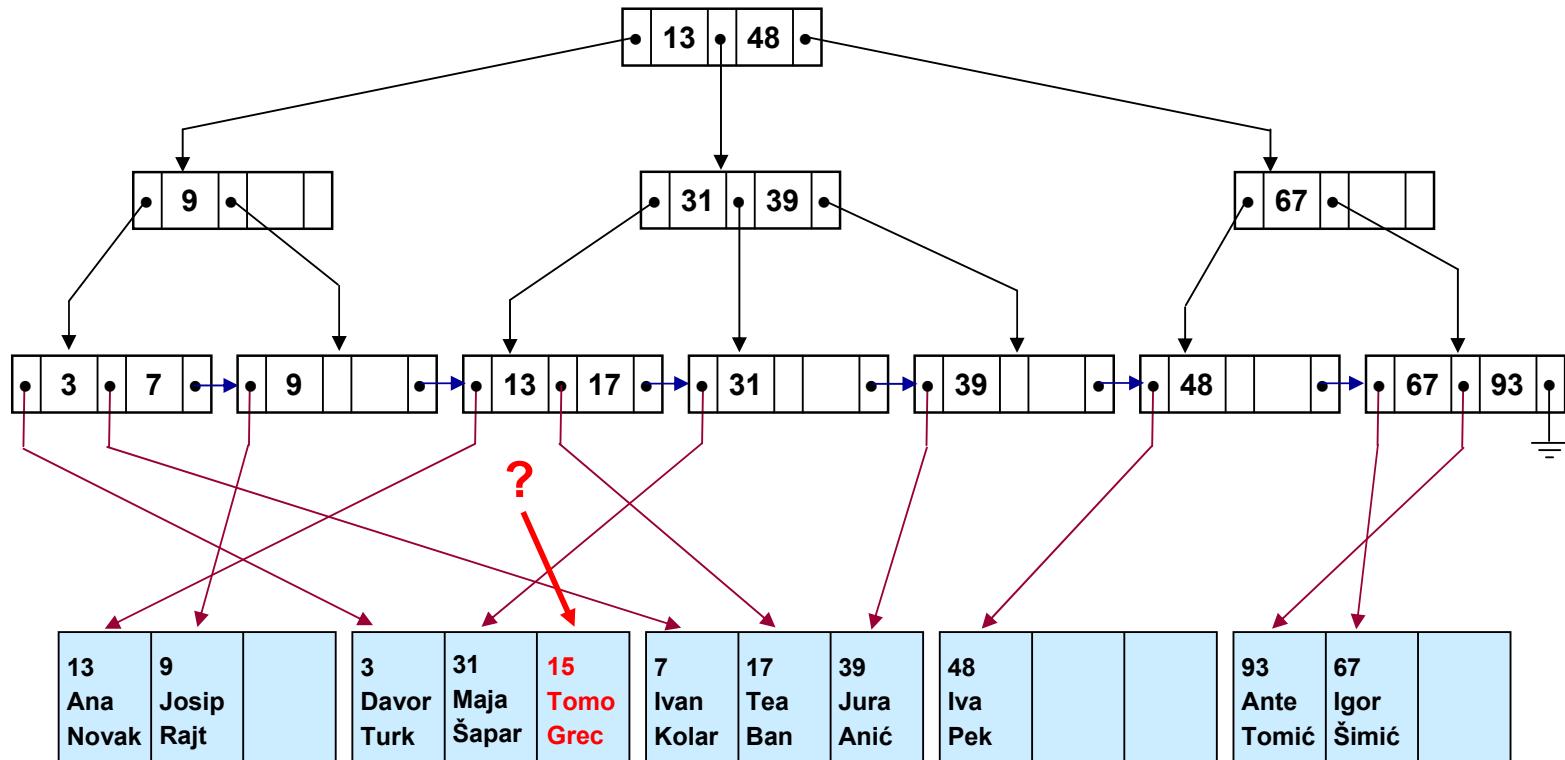
Dodavanje i brisanje zapisa

- nakon dodavanja ili brisanja zapisa u bloku s podacima, mijenja se i sadržaj B^+ - stabla
 - koriste se algoritmi za dodavanje i brisanje zapisa u B^+ - stablu
 - algoritmi osiguravaju ispravnu popunjenost internih čvorova i listova B^+ - stabla
 - pri tome se može dogoditi da stablo promijeni dubinu
- operacija izmjene
 - ukoliko se ne mijenjaju vrijednosti atributa za koje je izgrađeno B^+ -stablo, u B^+ -stablu nisu potrebne izmjene
 - ukoliko se mijenjaju vrijednosti atributa za koje je izgrađeno B^+ -stablo, u B^+ -stablu se obavlja algoritam za brisanje zapisa i algoritam za dodavanje zapisa
- **Važno dobro svojstvo algoritama B-stabla: dubina stabla se automatski prilagođava broju zapisa - čvorovi stabla (osim korijena) su uvijek barem 50% popunjeni**

Primjer dodavanja zapisa

- dodavanje zapisa u B^+ -stablo:

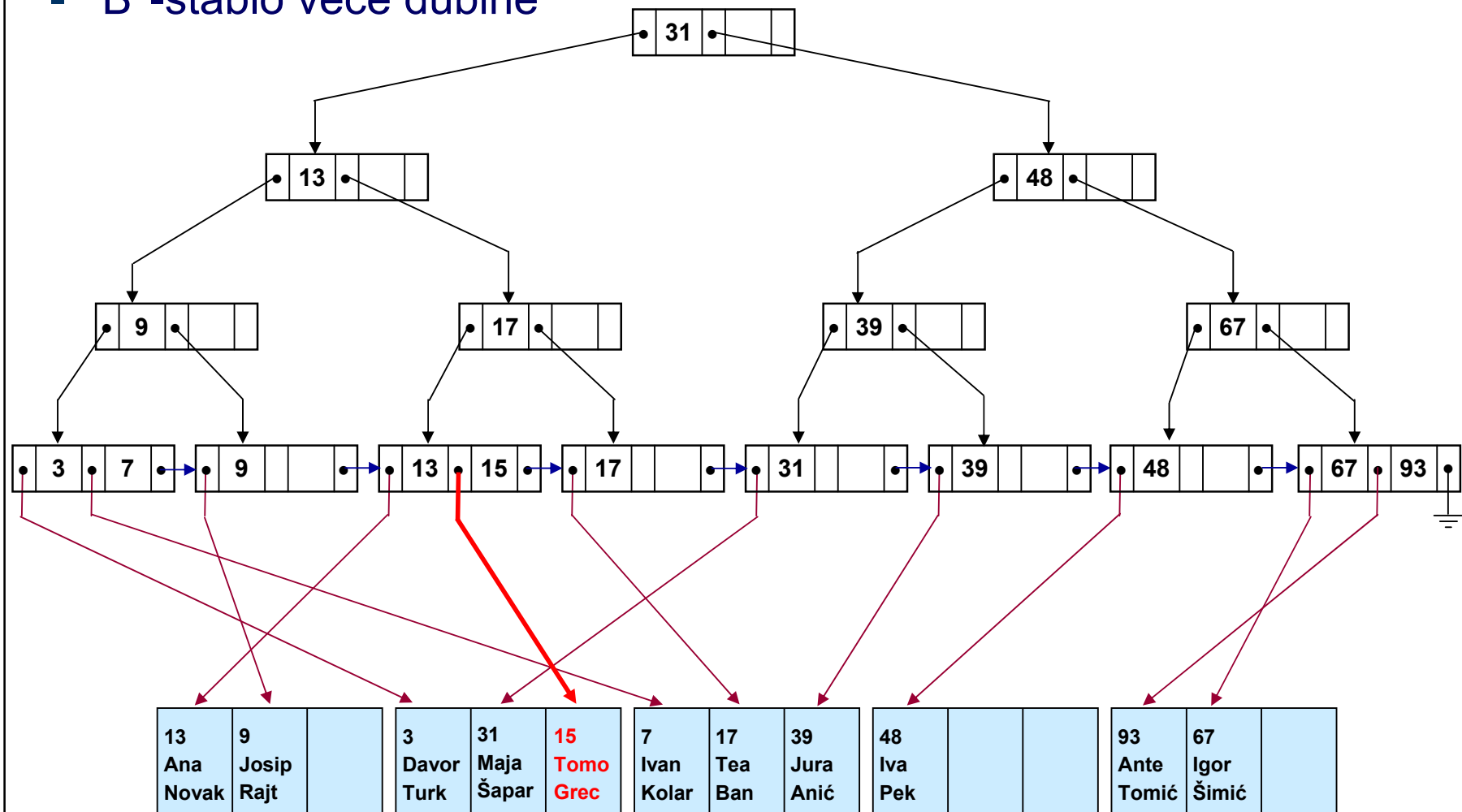
```
INSERT INTO osoba  
VALUES (15, 'Tomo', 'Grec');
```



- rezultat dodavanja zapisa u B^+ -stablo je na sljedećoj slici:

Rezultat dodavanja zapisa

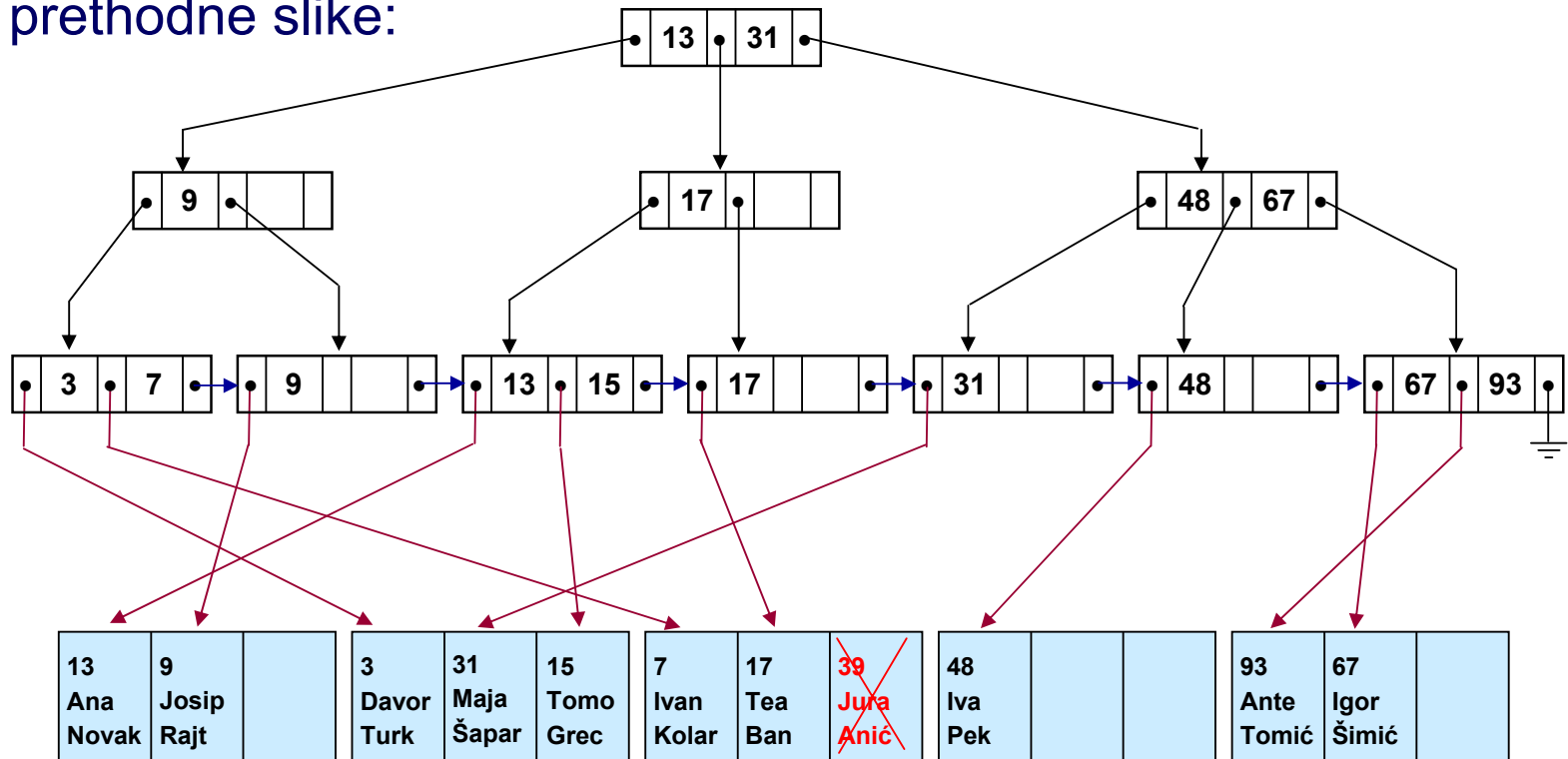
- B⁺-stablo veće dubine



Primjer brisanja zapisa

- obavljanjem operacije nad B⁺-stablom s prethodne slike:

DELETE FROM osoba WHERE mbr = 39;



- B⁺-stablo manje dubine

Učinkovitost operacije pretrage u B⁺-stablu

- pretpostavka: stablo reda **n** sadrži kazaljke na **m** zapisa podataka
- **n** se odabire tako da se sadržaj čvora može smjestiti u jedan fizički blok
 - ⇒ za dohvat **jednog** čvora potrebna je **jedna** UI operacija
- broj UI operacija u stablu pri traženju zapisa ovisi o broju razina u stablu jer se pri dohvat zapisa mora obaviti po jedna UI operacije za svaki čvor B-stabla na putu od čvora do lista
- B-stablo ima najveći broj razina onda kada su čvorovi najmanje popunjeni
 - ⇒ moguće je odrediti koliko će UI operacija biti potrebno obaviti u najlošijem slučaju

Učinkovitost operacije pretrage u B⁺-stablu

- **Primjer:** za broj n-torki $m = 1\ 000\ 000$, za **red** stabla $n = 70$, ukupni broj razina (uključujući i razinu korijena) u najlošijem slučaju je 4:

1. 

točno 1 čvor, najmanje 2 kazaljke

2. 

najmanje 2 čvora, najmanje 70 kazaljki

3. 

najmanje 70 čvorova, najmanje 2 450 kazaljki

4. 

najmanje 2 450 čvorova, najmanje 85 750 kazaljki

5.  najmanje 85 750 čvorova, najmanje 3 001 250 kazaljki

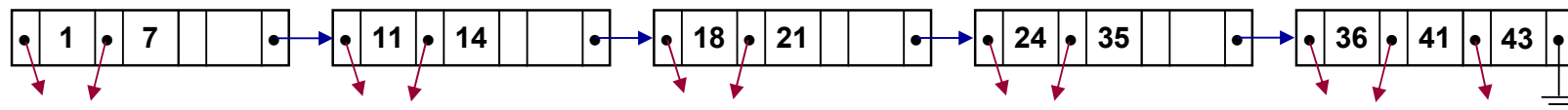
- B⁺-stablo koje bi imalo ukupno 5 razina, moralo bi imati **najmanje** 3 001 250 kazaljki na zapise. To znači da B-stablo reda 70 čije kazaljke u listovima pokazuju na 1 000 000 n-torki može imati najviše 4 razine.

⇒ Za dohvat zapisa prema vrijednosti ključa potrebno je najviše 5 UI operacija (4 UI operacije za dohvat lista u kojem se nalazi kazaljka na zapis + 1 UI operacija za dohvat bloka s podacima)

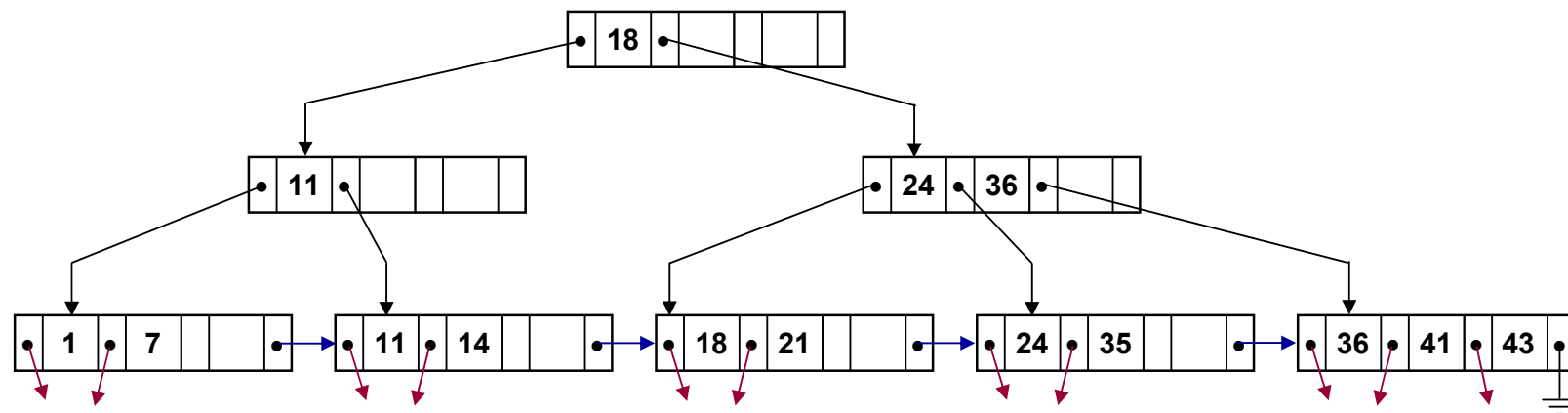
Zadatak 1.

- Relacija *stud* (*mbr*, *prez*, *ime*) sadrži n-torke sa sljedećim vrijednostima atributa *mbr* : 1, 7, 11, 14, 18, 21, 24, 35, 36, 41, 43. Nacrtati B⁺-stablo reda 4 za atribut *mbr* tako da popunjenost stabla bude minimalna.

- min. broj kazaljki na zapise (n-torke) u jednom listu je $\lceil (4 - 1) / 2 \rceil = 2$

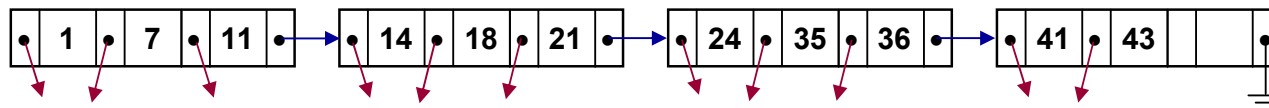


- min. broj kazaljki u jednom internom čvoru (osim korijena) je $\lceil 4 / 2 \rceil = 2$

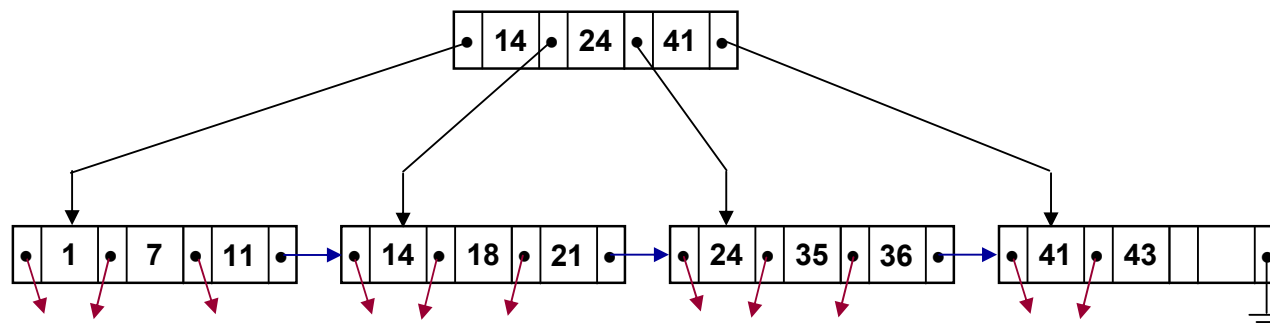


Zadatak 2.

- Relacija *stud* (*mbr*, *prez*, *ime*) sadrži n-torke sa sljedećim vrijednostima atributa *mbr* : 1, 7, 11, 14, 18, 21, 24, 35, 36, 41, 43. Nacrtati B⁺-stablo reda 4 za atribut *mbr* tako da popunjenost čvorova u stablu bude maksimalna.
- maksimalni broj kazaljki na zapise (n-torke) u jednom listu je $4 - 1 = 3$



- maksimalni broj kazaljki u jednom internom čvoru je 4



Zadatak 3.

- Koliko n-torki sadrži relacija ako je nad njom izgrađeno B⁺-stablo reda 101, s **ukupno** 5 razina, s minimalno dopuštenom popunjenošću **svih** čvorova
- min. broj kazaljki u jednom listu je $\lceil (101 - 1) / 2 \rceil = 50$
- min. broj kazaljki u jednom internom čvoru (osim korijena) je $\lceil 101 / 2 \rceil = 51$
- min. broj kazaljki u korijenu je 2
- relacija sadrži $2 \cdot 51 \cdot 51 \cdot 51 \cdot 50 \approx 1.33 \cdot 10^7$ n-torki
- **ZAKLJUČAK:** ako je B-stablo reda 101, do svake n-torke u relaciji koja sadrži $\approx 1.33 \cdot 10^7$ n-torki može se pristupiti, u najlošijem slučaju, korištenjem tek 6 UI operacija (5 UI za dohvat lista, 1 UI za dohvat fizičkog bloka u kojem se nalazi n-torka)

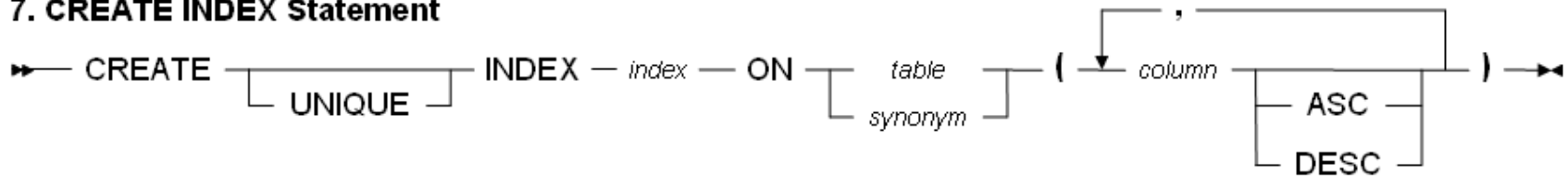
Zadatak 4.

- Koliko n-torki sadrži relacija ako je nad njom izgrađeno B⁺-stablo reda 101, s **ukupno** 5 razina, s maksimalno popunjenim **svim** čvorovima
- max. broj kazaljki u jednom listu je $101 - 1 = 100$
- max. broj kazaljki u internom čvoru je **101**
- relacija sadrži $101 \cdot 101 \cdot 101 \cdot 101 \cdot 100 \approx 1.04 \cdot 10^{10}$ n-torki

- **ZAKLJUČAK:** ako je B-stablo reda 101, u najboljem slučaju, korištenjem tek 6 UI operacija može se dohvatiti blok s n-torkom koja se nalazi u relaciji koja sadrži čak $\approx 1.04 \cdot 10^{10}$ n-torki

SQL: Indeksi

7. CREATE INDEX Statement



- Obavljanjem naredbe za kreiranje indeksa nad relacijom, nad blokovima s podacima relacije formira se struktura B-stabla

1

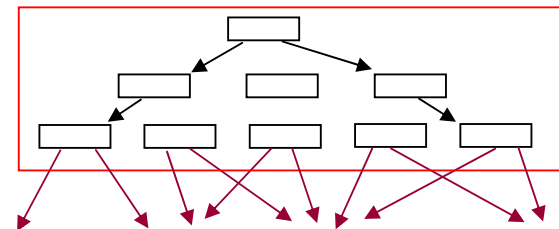
```
CREATE TABLE osoba (  
  mbr  INTEGER  
, ime  NCHAR(20)  
, prez NCHAR(20));  
  
INSERT INTO ...;
```

2

```
CREATE INDEX osoba_prez  
ON osoba (prez);
```



B-stablo za osoba.prez



7	17	39	48			93	67
Ivan	Tea	Jura	Iva			Ante	Igor
Kolar	Ban	Anić	Pek			Tomić	Šimić

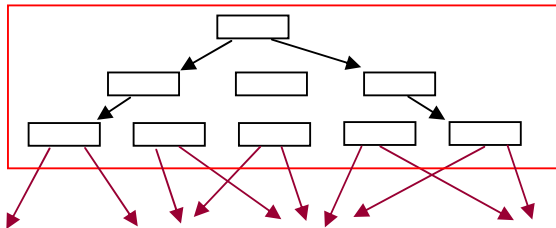
SQL: Indeksi

- Kreiranjem indeksa uz navođenje rezervirane riječi UNIQUE osigurava se jedinstvenost vrijednosti navedenog atributa

```
CREATE UNIQUE INDEX osoba_mbr  
ON osoba (mbr);
```



B-stablo za osoba.mbr



7	17	39	48		93	67
Ivan Kolar	Tea Ban	Jura Anić	Iva Pek		Ante Tomić	Igor Šimić

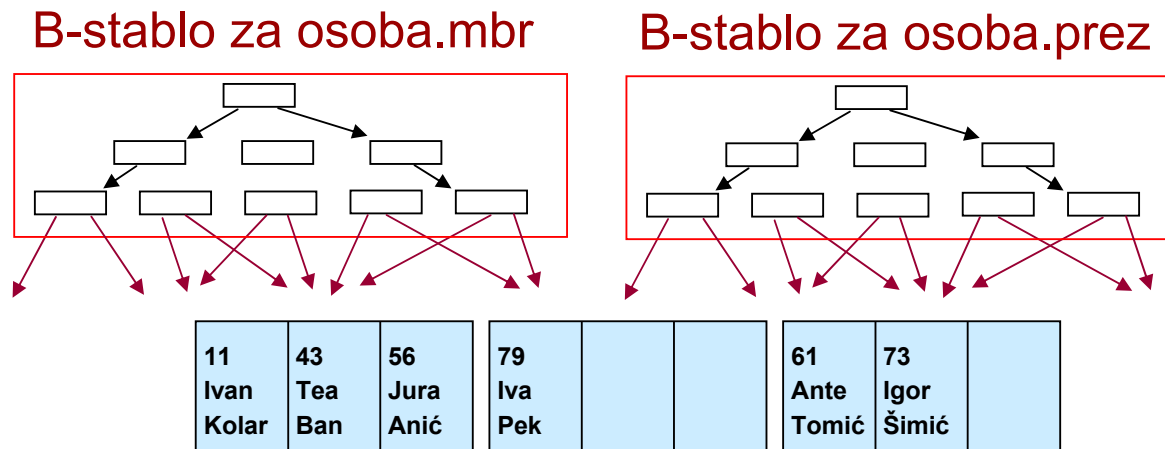
- ukoliko se indeks pokuša kreirati nad relacijom u kojoj već postoje duplikati vrijednosti atributa mbr, sustav će odbiti kreirati indeks i dojaviti pogrešku
- pokuša li se nakon kreiranja ovog indeksa unijeti n-torka s vrijednošću atributa mbr koja već postoji u nekoj n-torci, sustav će odbiti operaciju i dojaviti pogrešku

- Uništavanje indeksa - primjer:

```
DROP INDEX osoba_mbr;
```

Više indeksa nad istom relacijom

- nad istom relacijom može se izgraditi više indeksa



- B-stabla (indeksi) prikazani u primjeru omogućuju efikasno obavljanje upita s uvjetima (=, >, >=, <, <=, BETWEEN) i efikasno sortiranje (ASC, DESC)
 - za atribut mbr
 - za atribut prez
- prema uvjetima koji sadrže atribut ime, podacima se može pristupiti jedino linearnom pretragom svih blokova

Više indeksa nad istom relacijom

- Ako indeksi omogućuju efikasan pristup do n-torki, znači li to da bi indekse trebalo kreirati za svaki atribut u relaciji?

NE !!! Zašto?

- indeksi zauzimaju prostor
- operacija unosa ili brisanja n-torke
 - uvijek rezultira promjenama (manjim ili većim) B-stabla
 - npr. ako je nad relacijom izgrađeno 10 različitih indeksa, unosom jedne n-torke u blokove s podacima morat će se unijeti zapisi i u 10 različitih indeksa
- operacija izmjene n-torke
 - izmjena vrijednosti atributa A jedne n-torke rezultirat će brisanjem i dodavanjem zapisa u svim B-stablama za indekse u kojima se koristi atribut A

Za koje atribute treba kreirati indeks?

- za atribute koji se često koriste za postavljanje uvjeta selekcije (zašto?)
- za atribute prema kojima se obavlja spajanje relacija (zašto?)
 - primarni i alternativni ključevi relacije
 - strani ključevi
- za atribute prema kojima se često obavlja sortiranje ili grupiranje (zašto?)

Za koje attribute treba kreirati indeks?

- **Primjer:**

```
CREATE TABLE stud (  
  mbr  INTEGER  
, ime  NCHAR(20)  
, prez NCHAR(20));
```

- Često se postavljaju upiti oblika:

```
SELECT * FROM stud  
WHERE mbr = 12345;
```

```
SELECT * FROM stud  
WHERE prez > 'Kolar'  
ORDER BY prez;
```

⇒ Kreirati indeks za mbr i indeks za prez

- Što se nakon kreiranja navedenih indeksa dešava pri obavljanju:

```
UPDATE stud SET prez = UPPER(prez)  
WHERE ime = 'Ivan';
```

- n-torke se pronalaze linearnom pretragom (loše), a zbog izmjene vrijednosti atributa *prez* mora se izmijeniti sadržaj B-stabla za indeks nad atributom *prez* (loše)

```
UPDATE stud SET ime = UPPER(ime)  
WHERE prez = 'Horvat';
```

- n-torke se pronalaze pomoću B-stabla (dobro), nad atributom *ime* nije izgrađen indeks, stoga ne postoji B-stablo čiji se sadržaj mora izmijeniti (dobro)

Za koje atribute ne treba kreirati indeks?

- ako vrijednosti atributa imaju relativno mali broj različitih vrijednosti
 - npr. atribut spolOsobe s dozvoljenim vrijednostima M, Ž
- ako relaciji predstoji velik broj upisa, izmjena ili brisanja n-torki. Preporuča se u takvim slučajevima postojeće indekse izbrisati, te ih ponovo izgraditi tek nakon obavljenih promjena nad podacima
- ako relacija sadrži relativno mali broj n-torki (sve n-torke su pohranjene u nekoliko blokova). U takvim slučajevima B-stablo ne pridonosi efikasnosti pretrage
 - npr. relacija zupanija

Složeni indeksi

```
CREATE TABLE stud (  
    mbr    INTEGER  
    , ime  NCHAR(20)  
    , prez NCHAR(20));
```

```
CREATE INDEX osoba_ime ON osoba (ime);  
CREATE INDEX osoba_prez ON osoba (prez);
```

- efikasno se obavljaju upiti oblika:

```
SELECT * FROM stud  
WHERE prez = 'Kolar';
```

```
SELECT * FROM stud  
WHERE ime = 'Ivan';
```

- upit oblika:

```
SELECT * FROM stud  
WHERE prez = 'Kolar'  
AND ime = 'Ivan';
```

- pomoću indeksa nad atributom prez dohvatit će se n-torke studenata čije je prezime 'Horvat', ali će se u dobivenom skupu n-torki linearnom pretragom morati pronaći oni čije je ime 'Ivan' (ili indeksom po imenu, a onda linearno po prezimenu)

Složeni indeksi

- Prethodni upit se efikasnije obavlja ako se umjesto posebnih indeksa za attribute ime i prezime, kreira složeni indeks:

```
CREATE INDEX osoba_prez_ime ON osoba (prez, ime);
```

- problem: ovaj indeks se koristi za upite oblika:

```
SELECT * FROM stud  
  WHERE prez = 'Kolar'  
     AND ime = 'Ivan';
```

=

```
SELECT * FROM stud  
  WHERE ime = 'Ivan'  
     AND prez = 'Kolar';
```

- također i za upite oblika:

```
SELECT * FROM stud  
  WHERE prez = 'Kolar';
```

- ali se ne može koristiti za upite oblika:

```
SELECT * FROM stud  
  WHERE ime = 'Ivan';
```

Složeni indeksi

- Kako upotreba složenih indeksa utječe na sortiranje:

```
CREATE INDEX osoba_prez_ime1 ON osoba (prez, ime);
```

- indeks osoba_prez_ime1 se efikasno koristi za sortiranje oblika:

```
SELECT * FROM stud  
ORDER BY prez, ime;
```

```
SELECT * FROM stud  
ORDER BY prez DESC, ime DESC;
```

- ali ne i za:

```
SELECT * FROM stud  
ORDER BY prez DESC, ime;
```

- ako se kreira indeks:

```
CREATE INDEX osoba_prez_ime2 ON osoba (prez DESC, ime);
```

- indeks osoba_prez_ime2 se efikasno koristi za sortiranje oblika:

```
SELECT * FROM stud  
ORDER BY prez DESC, ime;
```

```
SELECT * FROM stud  
ORDER BY prez, ime DESC;
```

Složeni indeksi

- Ako je nad relacijom r (ABCD) kreiran složeni indeks r_abc1

```
CREATE INDEX r_abc1 ON r (A, B, C);
```

tada sljedeće indekse nije potrebno kreirati:

```
CREATE INDEX r_abc2 ON r (A DESC, B DESC, C DESC);  
CREATE INDEX r_a1 ON r (A);  
CREATE INDEX r_a2 ON r (A DESC);  
CREATE INDEX r_ab1 ON r (A, B);  
CREATE INDEX r_ab2 ON r (A DESC, B DESC);
```

- Indekse r_abc2, r_a1, r_a2, r_ab1 i r_ab2 ne treba kreirati jer SUBP može koristiti indeks r_abc1 u svim slučajevima u kojima bi se koristili indeksi r_abc2, r_a1, r_a2, r_ab1 i r_ab2.

Zadatak 5. zadana je relacija stud (mbr ime prez postBr)

- Za relaciju stud kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje (pomoću B⁺-stabla) svih navedenih upita:
 1. SELECT * FROM stud ORDER BY ime DESC, prez;
 2. SELECT * FROM stud ORDER BY ime DESC, prez DESC;
 3. SELECT * FROM stud ORDER BY ime, prez, pbrStan;
 4. SELECT * FROM stud WHERE prez = 'Novak' AND ime = 'Ivo';
 5. SELECT * FROM stud WHERE pbrStan > 51000 ORDER BY pbrStan DESC;

1. (ime DESC, prez)
2. (ime DESC, prez DESC)
3. (ime, prez, pbrStan) - ali sada više nije potreban indeks pod 2.
4. može se koristiti indeks pod 3.
5. (pbrStan)

Konačno rješenje - kreirati indekse za: (ime DESC, prez)
(ime, prez, pbrStan)
(pbrStan)

SQL naredbe za
kreiranje indeksa
napisati za vježbu!