Fizikai tervek (folytatás)

- Paraméterek, költségek
- Fizikai fájlszervezés,
  - szekvenciális (kupac), hasító indexek (statikus, dinamikus (kiterjeszthető, lineáris)
  - Rendezett állomány, elsődleges, másodlagos indexek, többszintű indexek, B^+-fák, B^*-fák
- Műveletek megvalósítása, kiszámítási költség, outputméret
- Optimális fizikai terv meghatározása

Indexelés

- **Rendezett állomány**
- Egy **rendező mező** alapján rendezett, azaz a blokkok láncolva vannak, és a következő blokkban nagyobb értékű rekordok szerepelnek, mint az előzőben.
- Ha a rendező mező és **kereső mező** nem esik egybe, akkor kupac szervezést jelent.
- Ha a rendező mező és kereső mező egybeesik, akkor **bináris (logaritmikus) keresés**t lehet alkalmazni:
  - **beolvassuk a középső blokkot,**
  - ha nincs benne az A=a értékű rekord, akkor eldöntjük, hogy a blokklánc második felében, vagy az első felében szerepelhet-e egyáltalán,
  - beolvassuk a felezett blokklánc középső blokkját,
  - addig folytatjuk, amíg megtaláljuk a rekordot, vagy a vizsgálandó maradék blokklánc már csak 1 blokkból áll.
- **Keresési idő**: **$\log_2(B)$**

Indexelés

- **Beszúrás:**
  – keresés + üres hely készítés miatt a rekordok eltolása az összes blokkban, az adott találati blokktól kezdve (B/2 blokkot be kell olvasni, majd az eltolások után visszaírni=B művelet)
- **Szokásos megoldások:**
- **Gyűjtő (túlcsordulási) blokk** használata:
  – az új rekordok számára nyitunk egy blokkot, ha betelik hozzáláncolunk egy újabb blokkokat,
  – keresést 2 helyen végezzük: **$\log_2(B-G)$** költséggel keresünk a rendezett részben, és ha nem találjuk, akkor a gyűjtőben is megnézzük (**G** blokkművelet, ahol **G a gyűjtő mérete**), azaz az összköltség: **$\log_2(B-G)+G$**
  – ha a G túl nagy a $\log_2(B)$ – hez képest, akkor újrarendezzük a teljes fájlt (a rendezés költsége **$B*\log_2(B)$**)

Indexelés

- **Üres helyeket** hagyunk a blokkokban:
  - például félig üresek a blokkok:
  - a keresés után 1 blokkművelettel visszaírjuk a blokkot, amibe beírtuk az új rekordot,
  - tárméret **2*B** lesz
  - keresési idő: $\log_2(2*B) =$ **$1+\log_2(B)$**
  - ha betelik egy blokk, vagy elér egy határt a telítettsége, akkor 2 blokkba osztjuk szét a rekordjait, a rendezettség fenntartásával.
- **Törlés:**
  - keresés + a törlés elvégzése, vagy a törlési bit beállítása után visszaírás (1 blokkírás)
  - túl sok törlés után újraszervezés
- **Frissítés: törlés + beszúrás**

Indexelés

- **Indexek használata:**
  - keresést gyorsító segédstruktúra
  - több mezőre is lehet indexet készíteni
  - az index tárolása növeli a tárméretet
  - **nem csak a főfájlt, hanem az indexet is karban kell tartani, ami plusz költséget jelent**
  - ha a keresési mező egyik indexmezővel sem esik egybe, akkor kupac szervezést jelent
- **Az indexrekordok szerkezete:**
  - **(a,p)**, ahol a egy érték az indexelt oszlopban, p egy **blokkmutató**, arra a blokkra mutat, amelyben az **A=a** értékű rekordot tároljuk.
  - az **index mindig rendezett** az indexértékek szerint
  - Oracle SQL-ben:
  - **egyszerű index:**
    - **CREATE INDEX supplier_idx**
      ON supplier (supplier_name);
  - **összetett index:**
    - **CREATE INDEX supplier_idx**
      ON supplier (supplier_name, city)
      COMPUTE STATISTICS;
    - -- az optimalizáláshoz szükséges statisztikák elkészítésével --

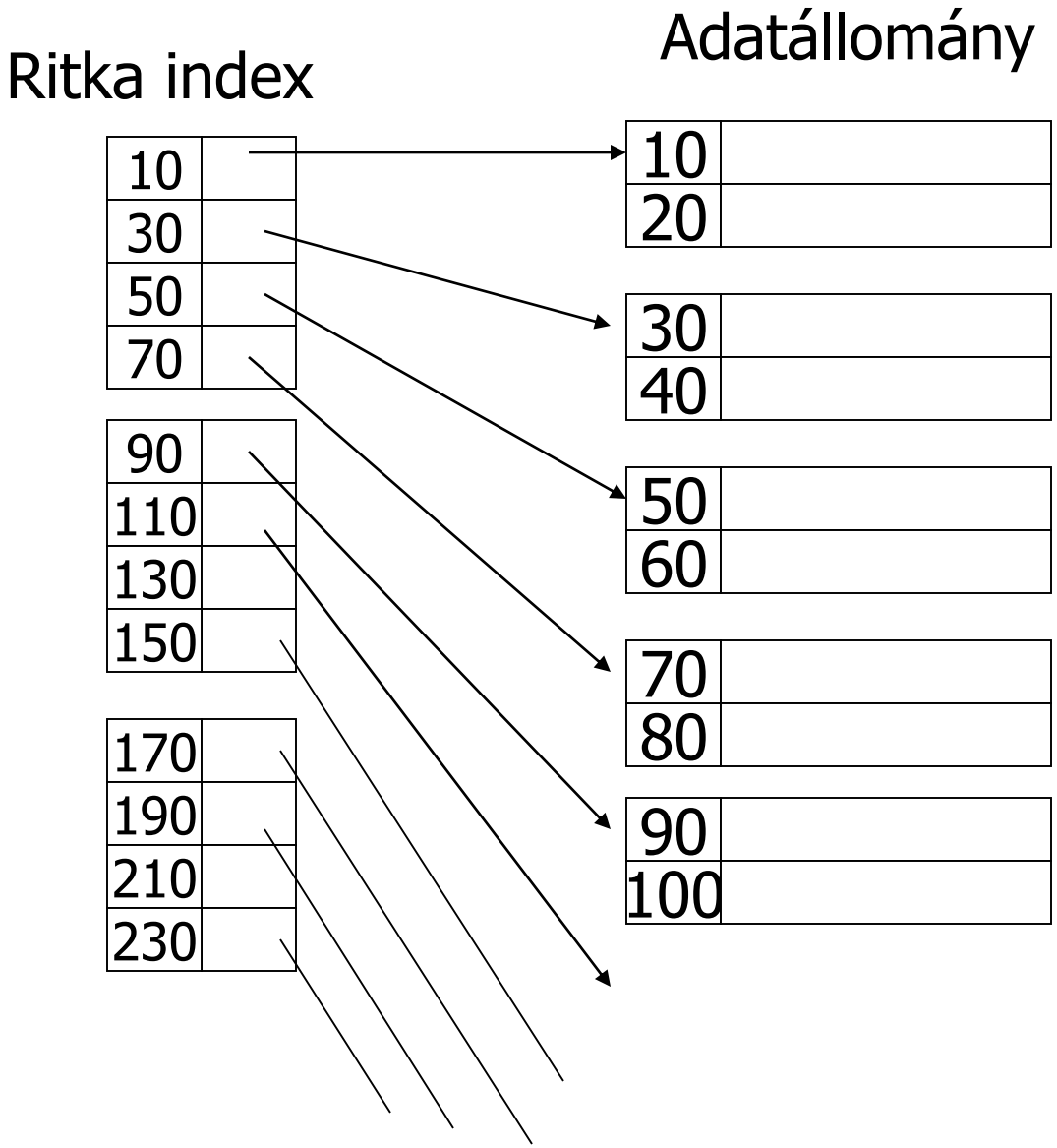
Indexelés

- **Elsődleges index:**
  - – **főfájl is rendezett**
  - – csak 1 elsődleges indexet lehet megadni (mert csak egyik mező szerint lehet rendezett a főfájl.
  - – elég a főfájl minden blokkjának legkisebb rekordjához készíteni indexrekordot
  - – indexrekordok száma: T(I)=B **(ritka index)**
  - – indexrekordból sokkal több fér egy blokkba, mint a főfájl rekordjaiból: **bf(I)>>bf**, azaz az indexfájl sokkal kisebb rendezett fájl, mint a főfájl:
  - **B(I)** = B / bf(I) **<<** **B**=T / bf
- **Keresési idő:**
  - – az indexfájlban nem szerepel minden érték, ezért csak **fedő értéket kereshetünk**, **a legnagyobb olyan indexértéket, amely a keresett értéknél kisebb vagy egyenlő**
  - – fedő érték keresése az index rendezettsége miatt bináris kereséssel történik: **$\log_2(B(I))$**
  - – a fedő indexrekordban szereplő blokkmutatónak megfelelő blokkot még be kell olvasni
  - – **$1+\log_2(B(I)) << \log_2(B)$ (rendezett eset)**
- **Módosítás:**
  - – rendezett fájlba kell beszúrni
  - – ha az első rekord változik a blokkban, akkor az indexfájlba is be kell szúrni, ami szintén rendezett
  - – megoldás: **üres helyeket hagyunk a főfájl, és az indexfájl blokkjaiban is**. Ezzel a tárméret duplázódhat, de a beszúrás legfeljebb egy főrekord és egy indexrekord visszaírását jelenti.

Indexelés

Elsődleges index

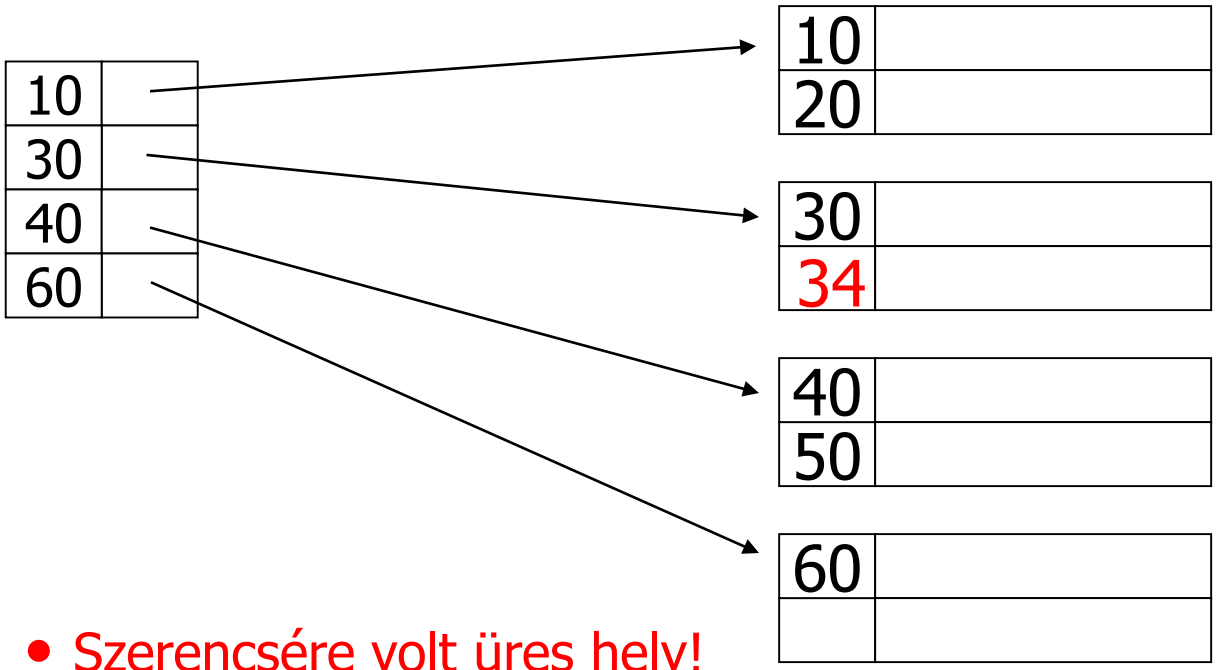
Az adatfájl rendezett, ezért elég a blokkok első rekordjaihoz indexrekordokat tárolni.



Indexelés

Beszúrás ritka index esetén:

Vigyük be a 34-es rekordot!

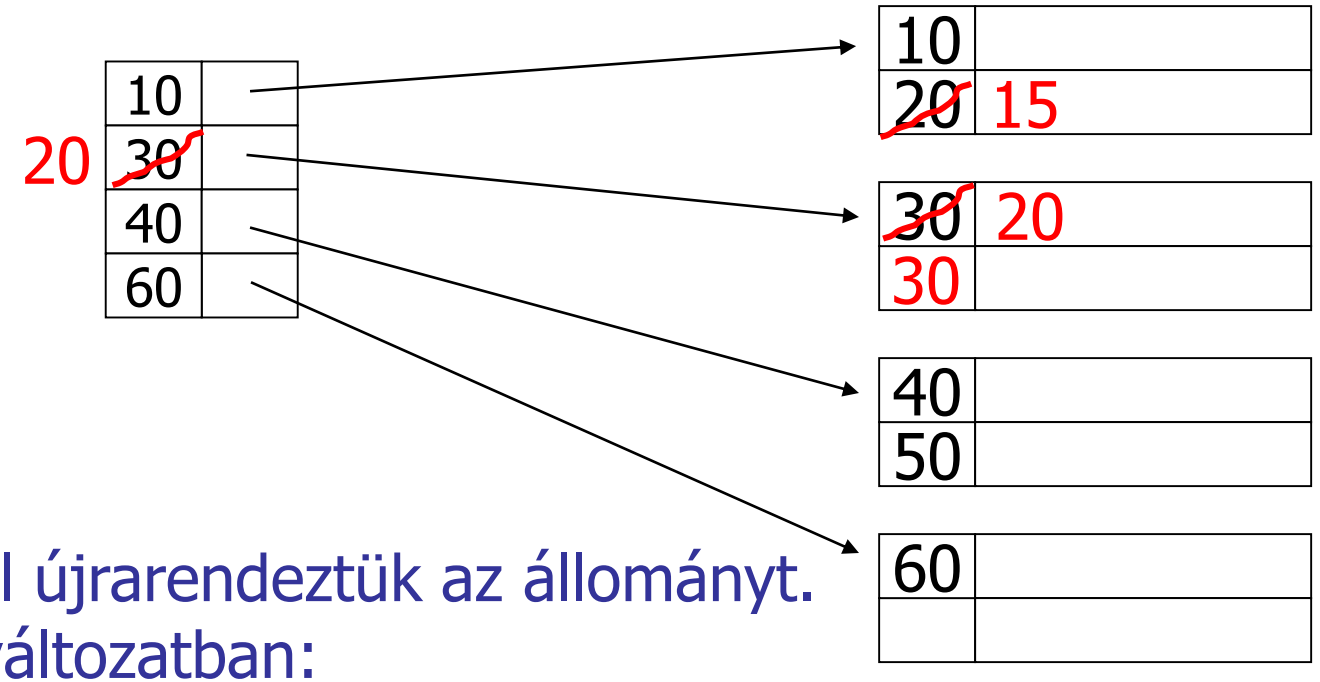


- Szerencsére volt üres hely!

Indexelés

Beszúrás ritka index esetén:

Vigyük be a 15-ös rekordot!

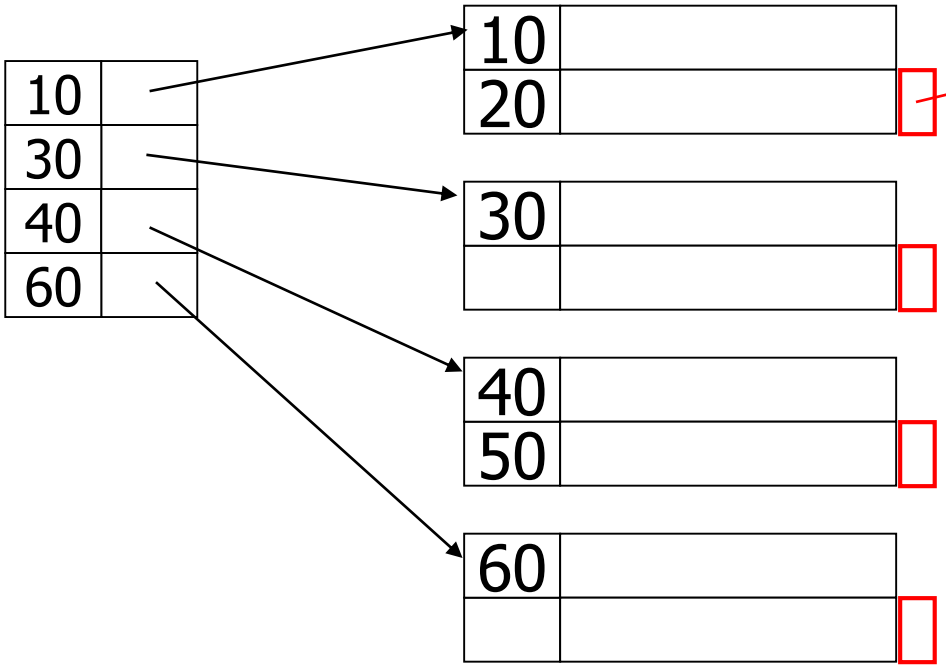


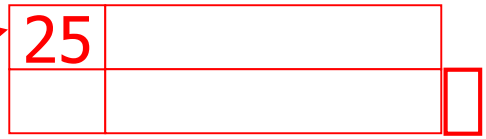
- Azonnal újrarendeztük az állományt.
- Másik változatban:
  - túlcsordulási blokkot láncolunk a blokkhoz

Indexelés

Beszúrás ritka index esetén:

Vigyük be a 25-ös rekordot!



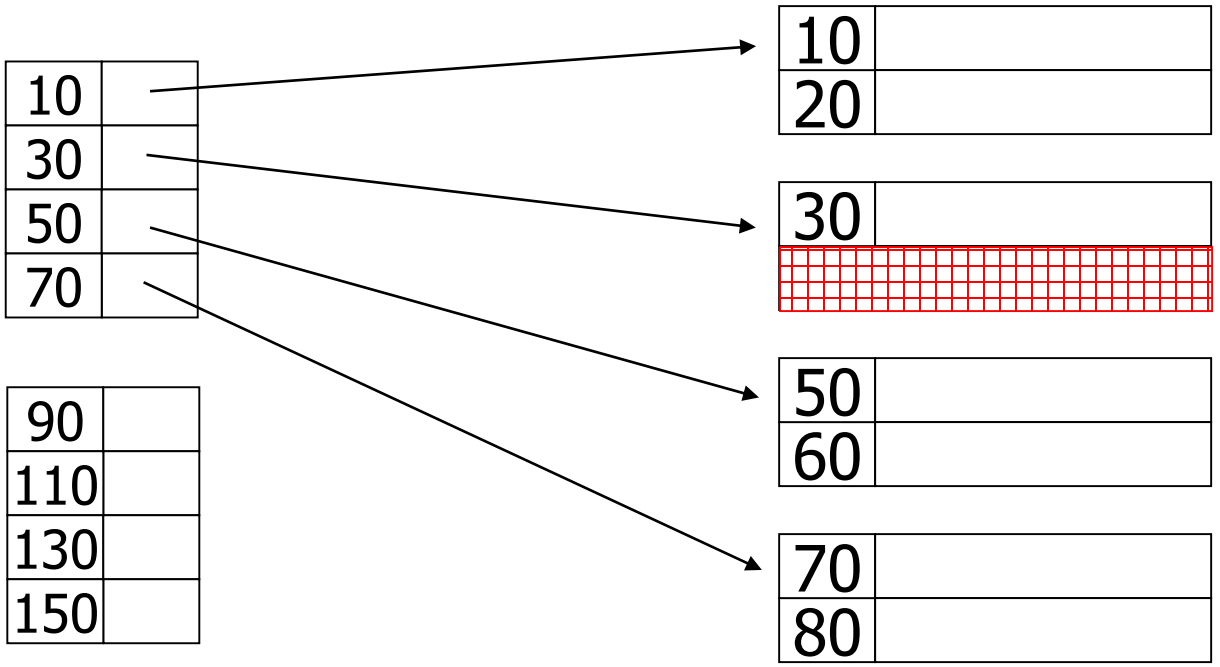


Túlcsordulási blokkot nyitunk, és későbbre halasztjuk az újrarendezést.

Indexelés

Törlés ritka indexből:

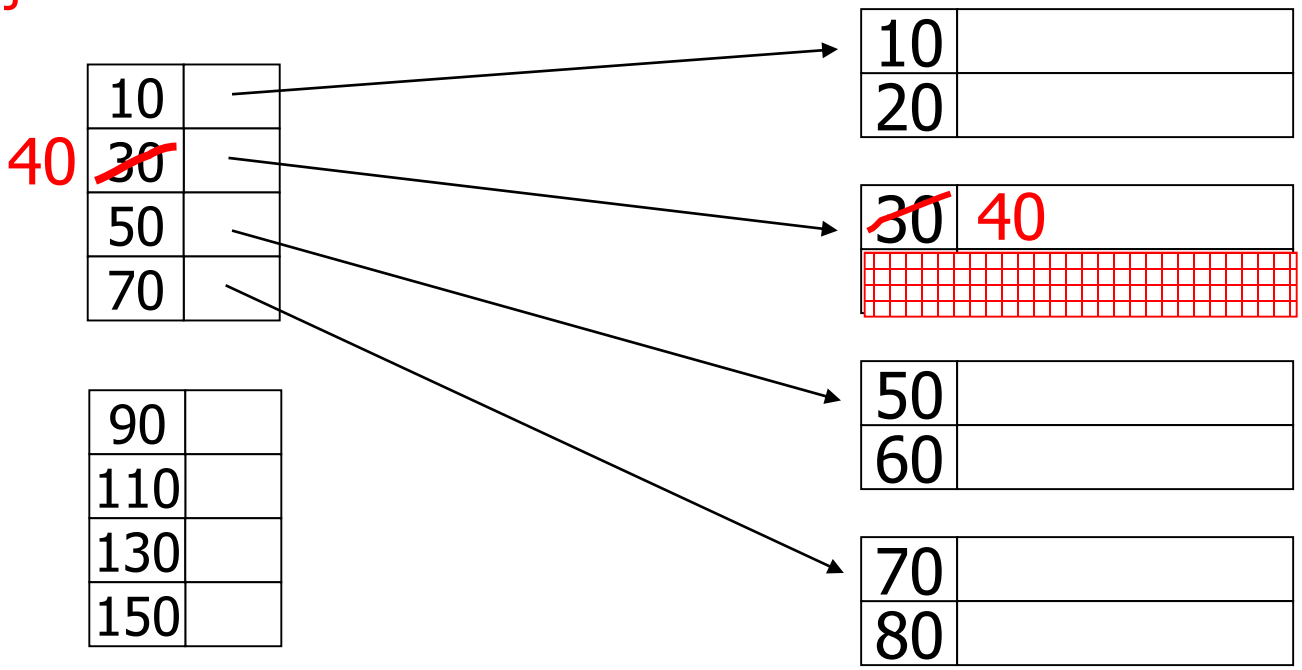
Töröljük a 40-es rekordot!



Indexelés

Törlés ritka indexből:

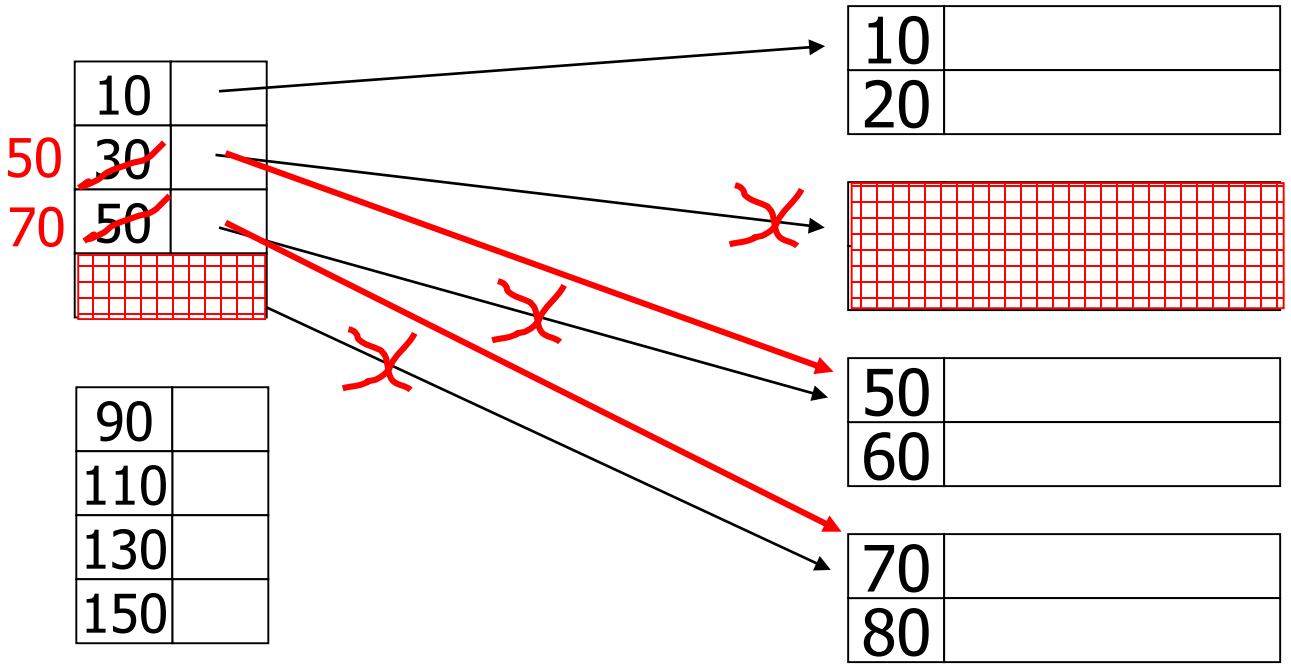
Töröljük a 30-as rekordot!



Indexelés

Törlés ritka indexből:

Töröljük a 30-as és 40-es rekordot!



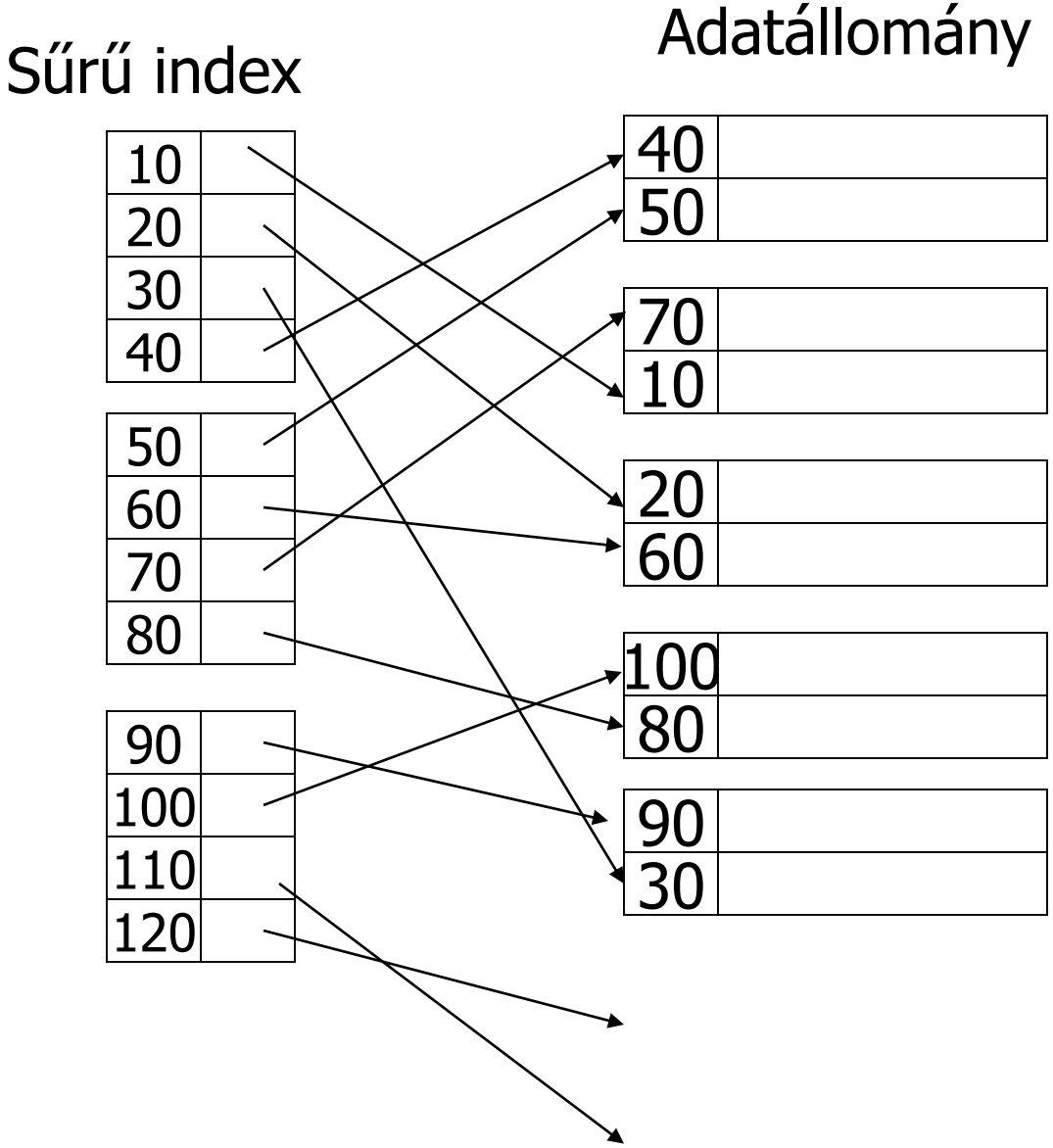
Indexelés

- **Másodlagos index:**
  - **főfájl rendezetlen (az indexfájl mindig rendezett)**
  - több másodlagos indexet is meg lehet adni
  - **a főfájl minden rekordjához kell készíteni indexrekordot**
  - indexrekordok száma: T(I)=T (sűrű index)
  - indexrekordból sokkal több fér egy blokkba, mint a főfájl rekordjaiból: **bf(I)>>bf**, azaz az indexfájl sokkal kisebb rendezett fájl, mint a főfájl:
  - **B(I) = T/bf(I) << B=T/bf**
- **Keresési idő:**
  - az indexben keresés az index rendezettsége miatt bináris kereséssel történik: **$\log_2(B(I))$**
  - a talált indexrekordban szereplő blokkmutatónak megfelelő blokkot még be kell olvasni
  - **$1+\log_2(B(I))$** << $\log_2(B)$ (rendezett eset)
  - az elsődleges indexnél rosszabb a keresési idő, mert több az indexrekord
- **Módosítás:**
  - a főfájl kupac szervezésű
  - rendezett fájlba kell beszúrni
  - ha az első rekord változik a blokkban, akkor az indexfájlba is be kell szúrni, ami szintén rendezett
  - megoldás: **üres helyeket hagyunk a főfájl, és az indexfájl blokkjaiban is**. Ezzel a tárméret duplázódhat, de a beszúrás legfeljebb egy főrekord és egy indexrekord visszaírását jelenti.

Indexelés

Másodlagos index

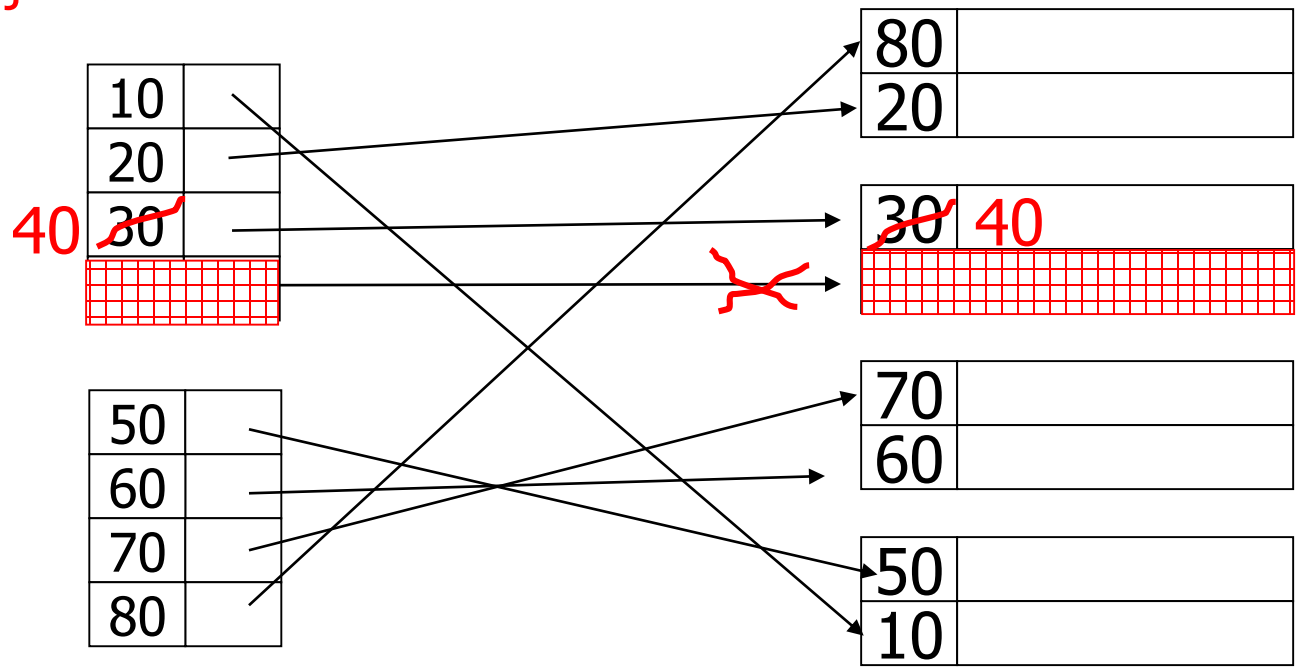
Minden rekordhoz tartozik indexrekord.



Indexelés

Törlés sűrű indexből:

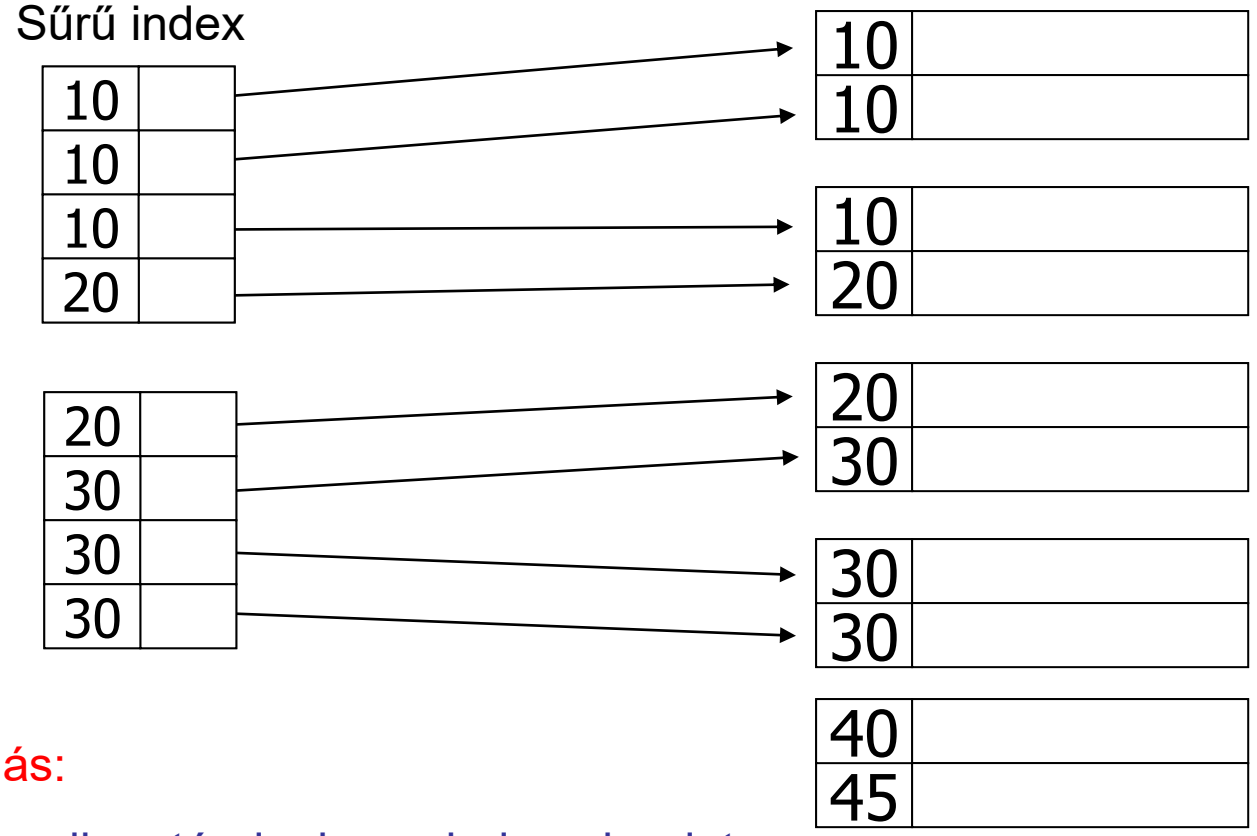
Töröljük a 30-as rekordot!



Indexelés

Mi történik, ha egy érték többször is előfordulhat?

Több megoldás is lehetséges. Először tegyük fel, hogy rendezett az állomány.

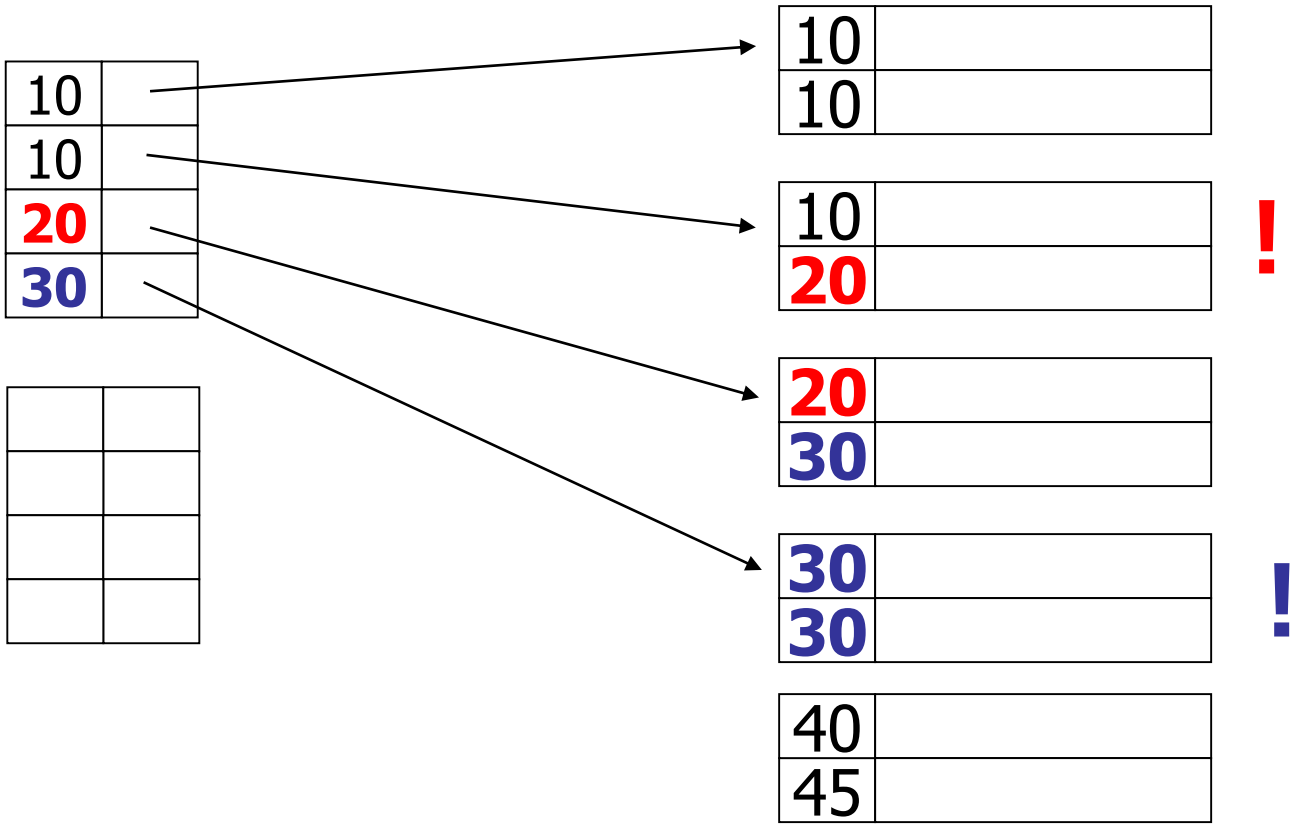


1. megoldás:

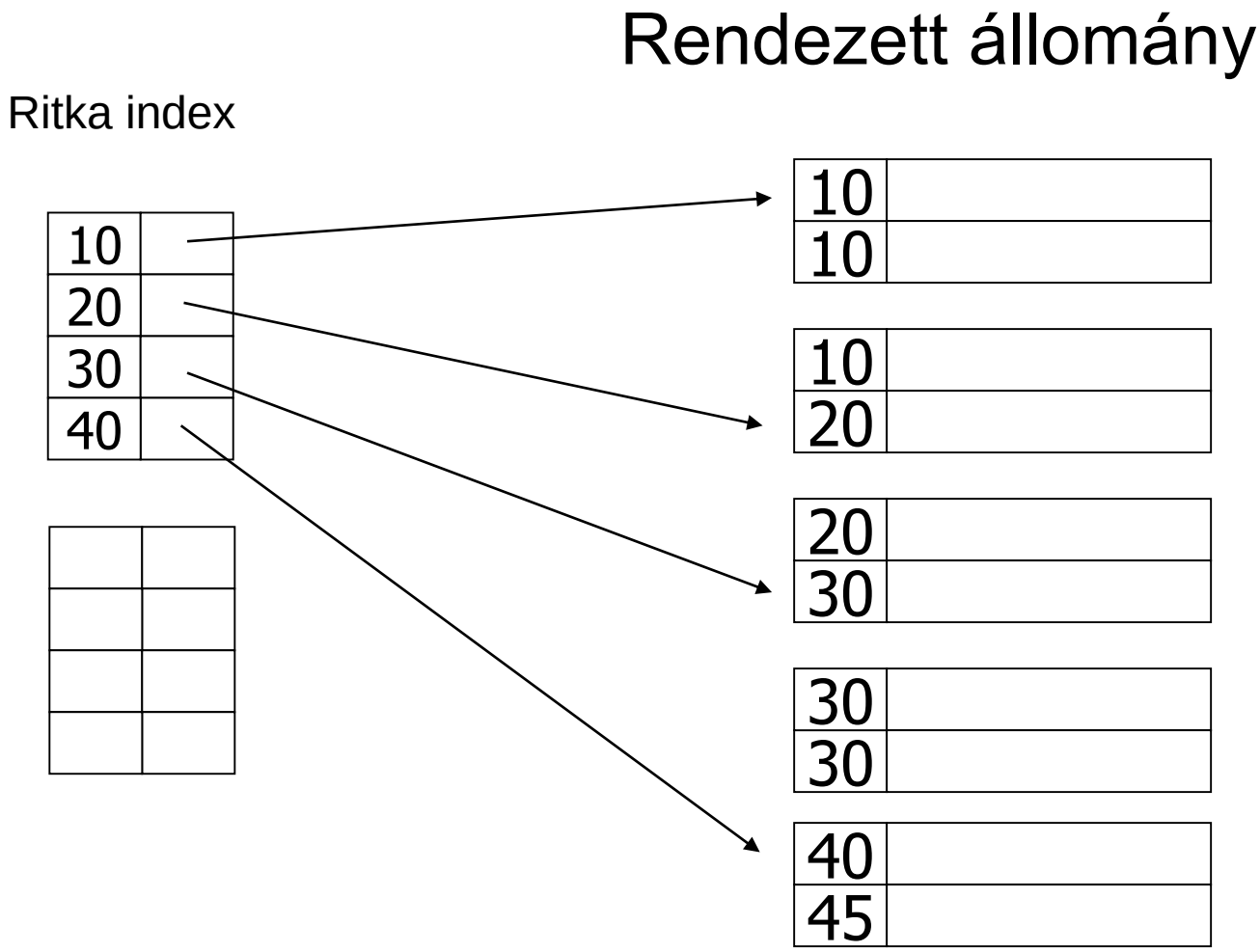
Minden rekordhoz tárolunk egy indexrekordot.

Indexelés

Vigyázat! Ritka index nem jó. A fedőértéknek megfelelő blokk előtti és utáni blokkokban is lehetnek találatok. Például, ha a 20-ast vagy a 30-ast keressük.



Indexelés

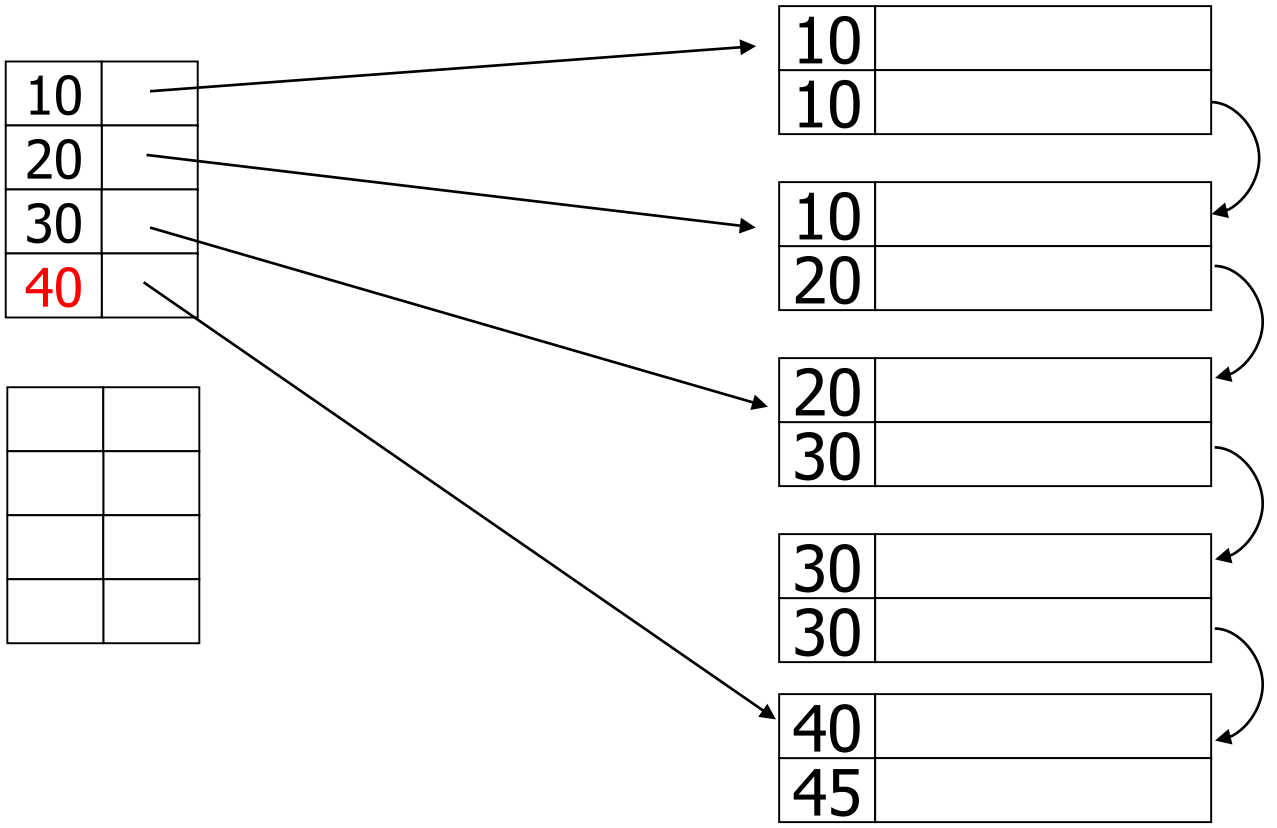


2. megoldás:

Rendezett állomány esetén csak az első előforduláshoz tárolunk egy indexrekordot.

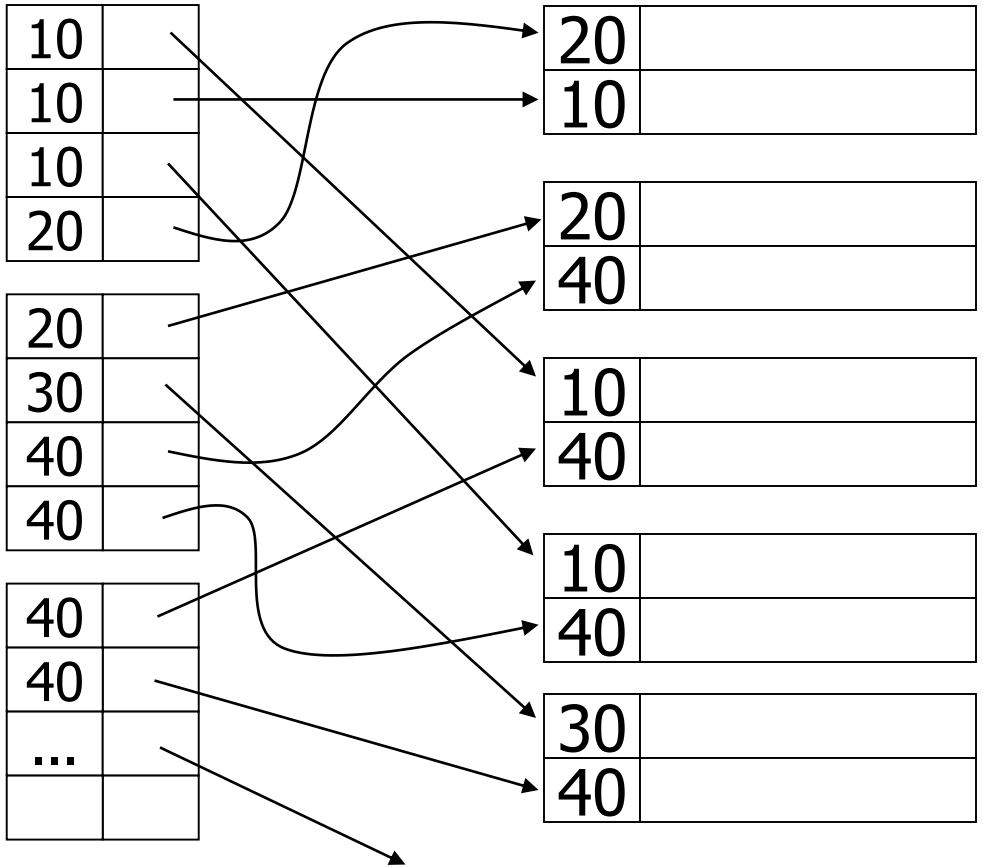
Indexelés

Rendezett állomány esetén nem az első rekordhoz, hanem az értékhez tartozó első előforduláshoz készítünk indexrekordot. Az adatállomány blokkjait láncoljuk.



Indexelés

Ha nem rendezett az állomány, akkor nagy lehet a tárolási és keresési költség is:

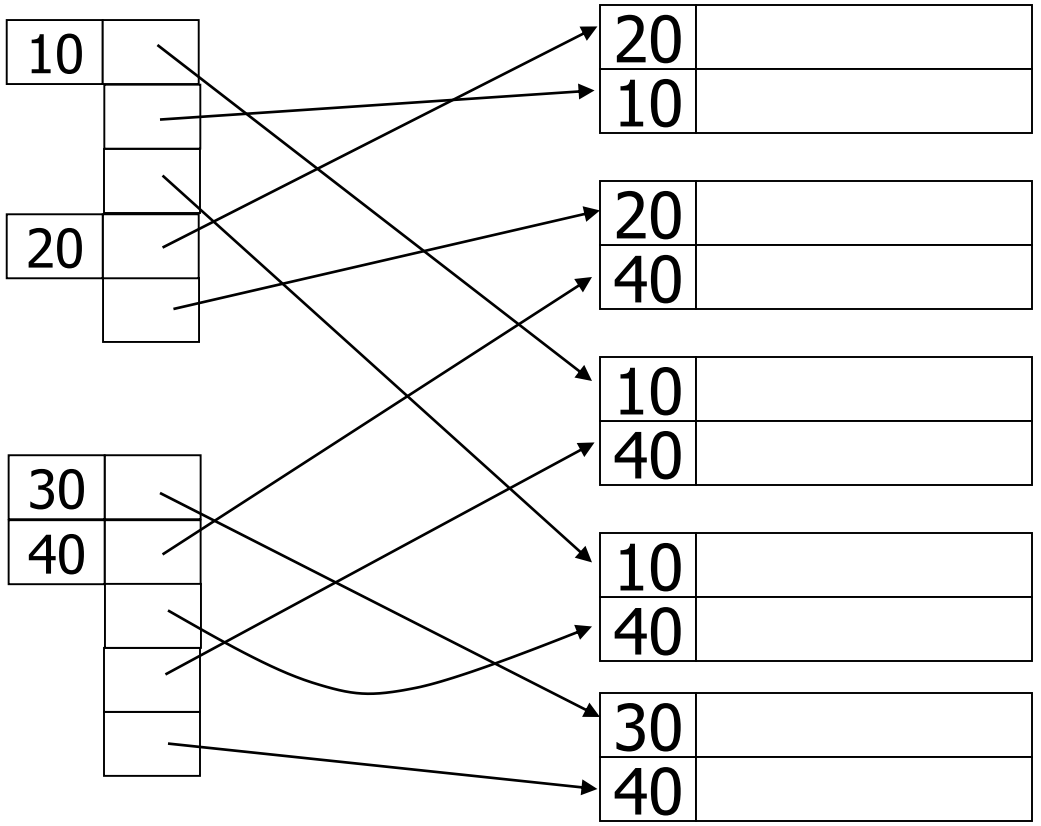


Indexelés

Egy lehetséges megoldás, hogy az indexrekordok szerkezetét módosítjuk:

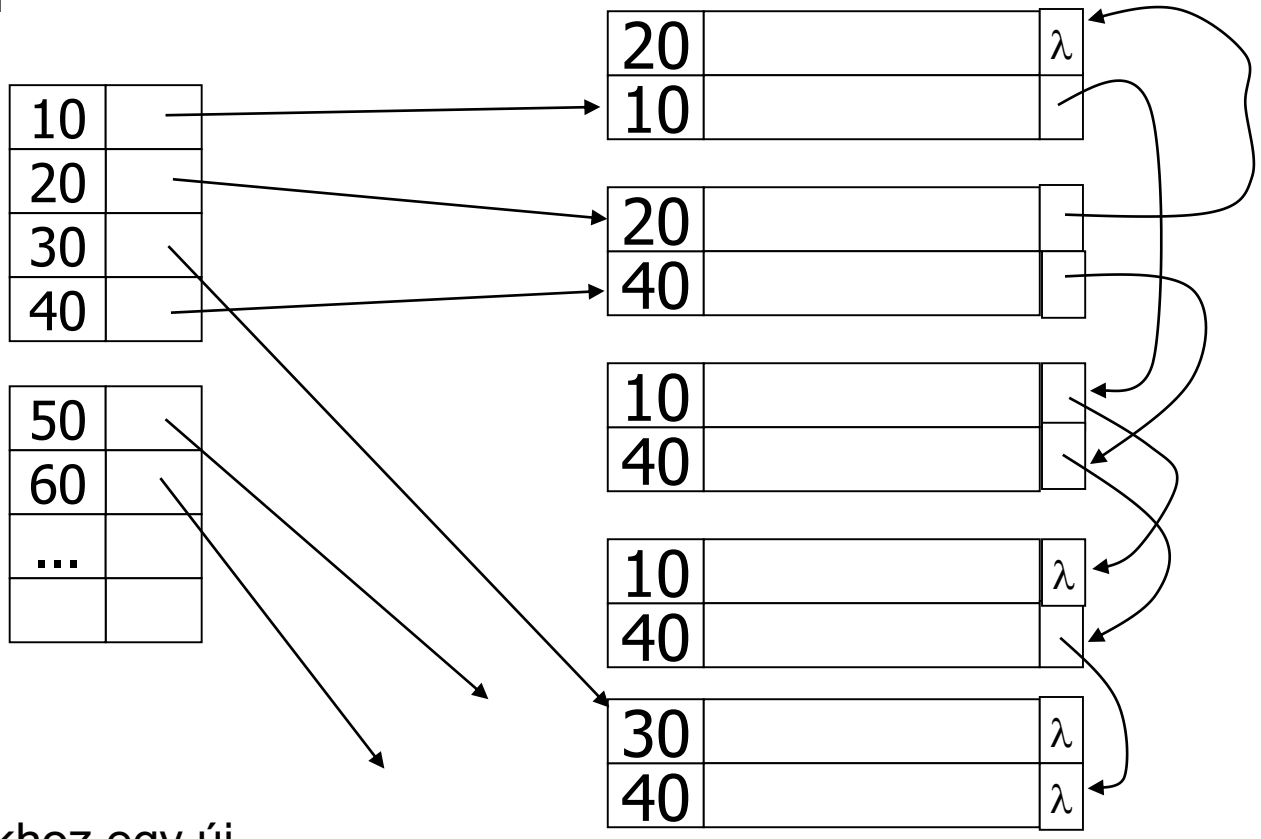
(indexérték, mutatóhalmaz)

Probléma: változó hosszú indexrekordok keletkeznek



Indexelés

Összeláncolhatjuk az egyforma értékű rekordokat.

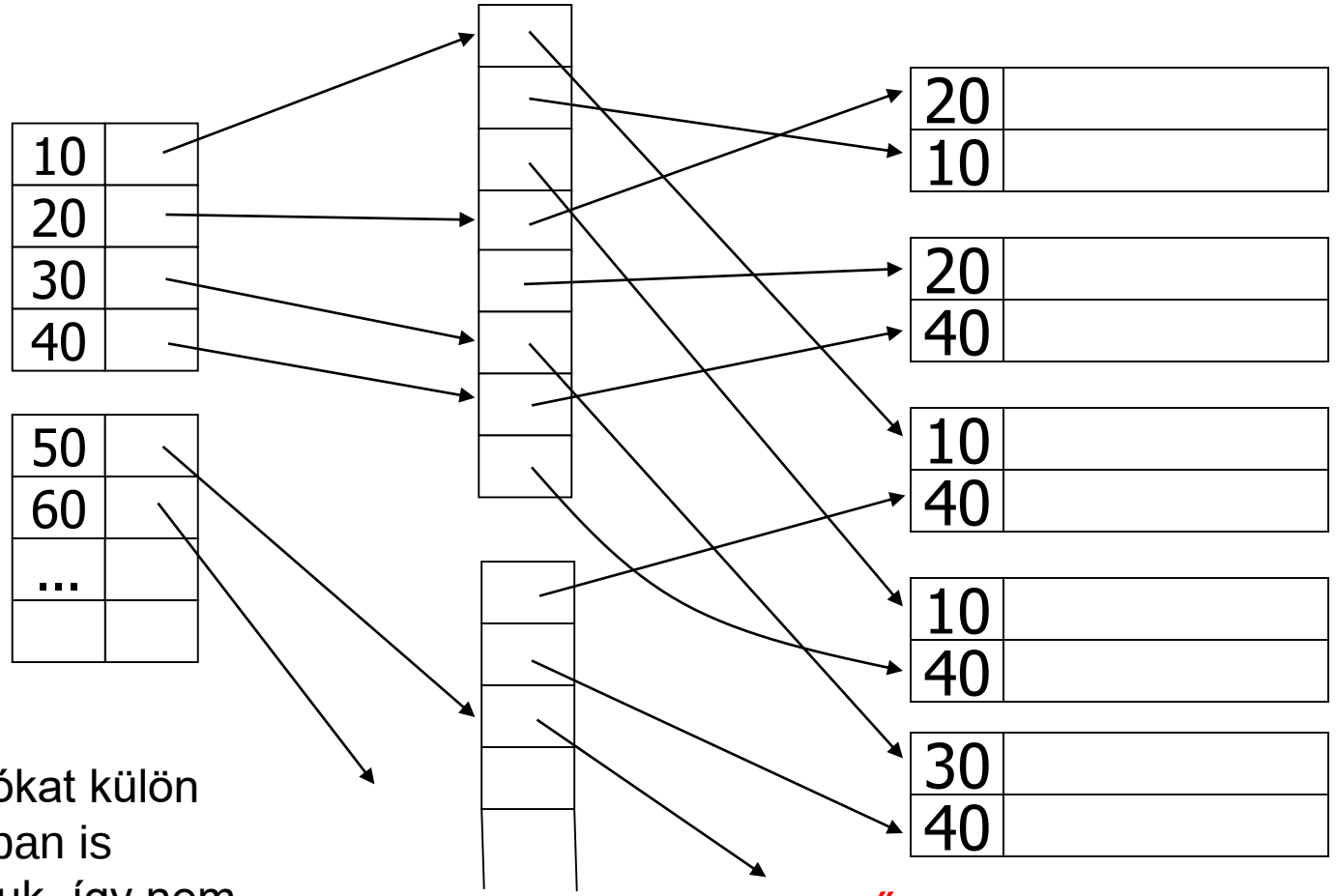


Probléma:

- a rekordokhoz egy új, mutató típusú mezőt kell adnunk

- követni kell a láncot

Indexelés



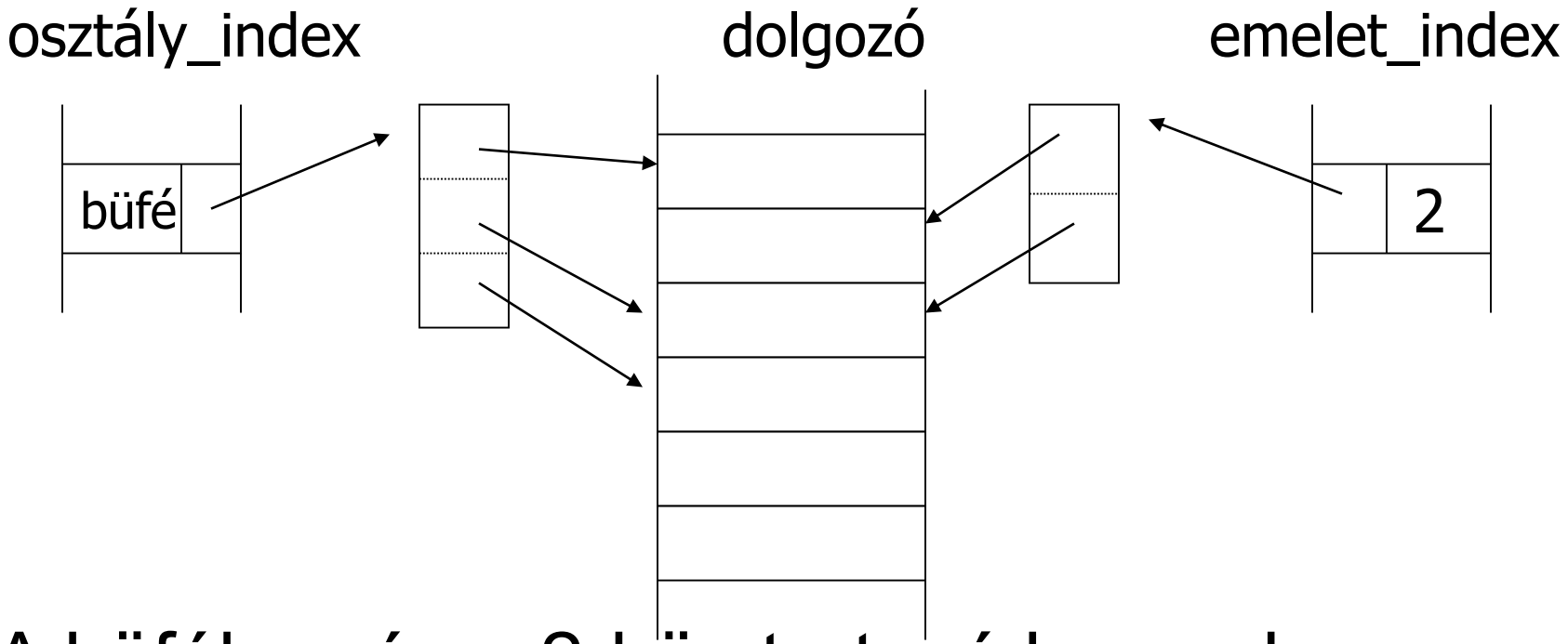
A mutatókat külön blokkokban is tárolhatjuk, így nem kell változó hosszú indexrekordokat kezelni.

Kosarak

ELŐNY: több index esetén a logikai feltételek halmazműveletekkel kiszámolhatók.

Indexelés

select * from dolgozó where osztály='büfé' and emelet=2;



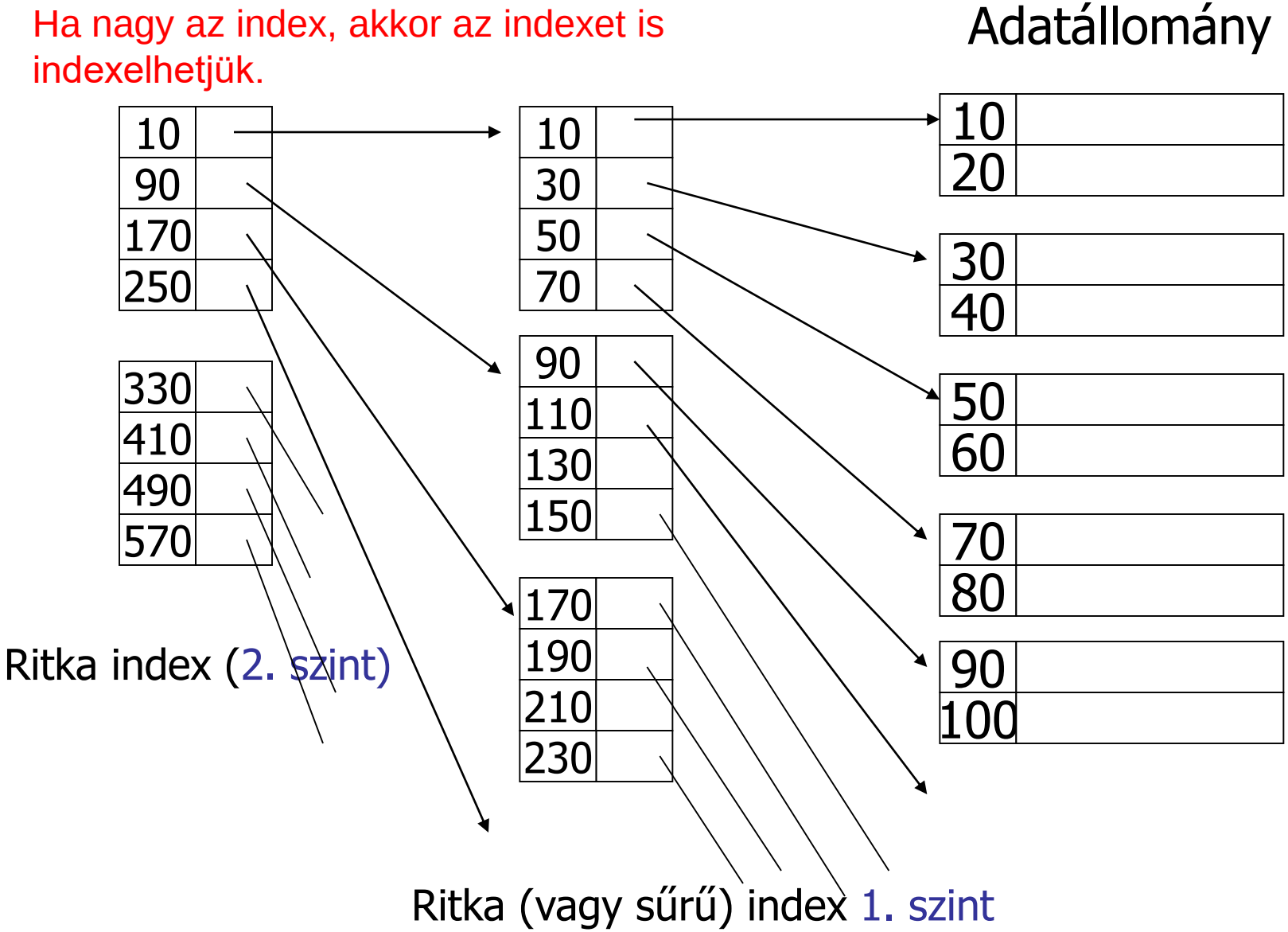
A büféhez és a 2-höz tartozó kosarak metszetét kell képezni, hogy megkapjuk a keresett mutatókat.

Indexelés

- **Klaszter (nyaláb, fürt)**
- Klaszterszervezés egy tábla esetén egy A oszlopra:
  - az azonos A-értékű sorok fizikailag egymás után blokkokban helyezkednek el.
  - CÉL: az első találat után az összes találatot megkapjuk soros beolvasással.
- **Klaszterindex:**
  - klaszterszervezésű fájl esetén index az A oszlopra
- Klaszterszervezés két tábla esetén az összes közös oszlopra:
  - a közös oszlopokon egyező sorok egy blokkban, vagy fizikailag egymás utáni blokkokban helyezkednek el.
  - CÉL: összekapcsolás esetén az összetartozó sorokat soros beolvasással megkaphatjuk.

Indexelés

Ha nagy az index, akkor az indexet is indexelhetjük.



Indexelés

- **Többszintű index:**
  - az indexfájl (1. indexszint) is fájl, ráadásul rendezett, így ezt is meg lehet indexelni, elsődleges indexszel.
  - a főfájl lehet rendezett vagy rendezetlen (az indexfájl mindig rendezett)
  - **t-szintű index**: az indexszinteket is indexeljük, összesen t szintig

Keresési idő:

- a t-ik szinten ($I^{(t)}$) bináris kereséssel **keressük meg a fedő indexrekordot**
- követjük a mutatót, minden szinten, és végül a főfájlban: **$\log_2(B(I^{(t)}))$+t blokkolvasás**
- ha a legfelső szint 1 blokkból áll, akkor **t+1** blokkolvasást jelent. **(t=?)**
- **minden szint blokkolási faktora megegyezik**, mert egyforma hosszúak az indexrekordok.

Indexelés

	FŐFÁJL	1. szint	2. szint	...	t. szint
blokkok száma	B	B/bf(I)	B/bf(I)^2	...	B/bf(I)^t
rekordok száma	T	B	B/bf(I)	...	B/bf(I)$^{(t-1)}$
blokkolási faktor	bf	bf(I)	bf(I)	...	bf(I)

- t-ik szinten 1 blokk: $1=B/bf(I)^t$

azaz $t=\log_{bf(I)}B < \log_2(B)$ azaz jobb a rendezett fájlszervezésnél.

- $\log_{bf(I)}B < \log_2(B(I))$ is teljesül általában, így az egyszintű indexeknél is gyorsabb

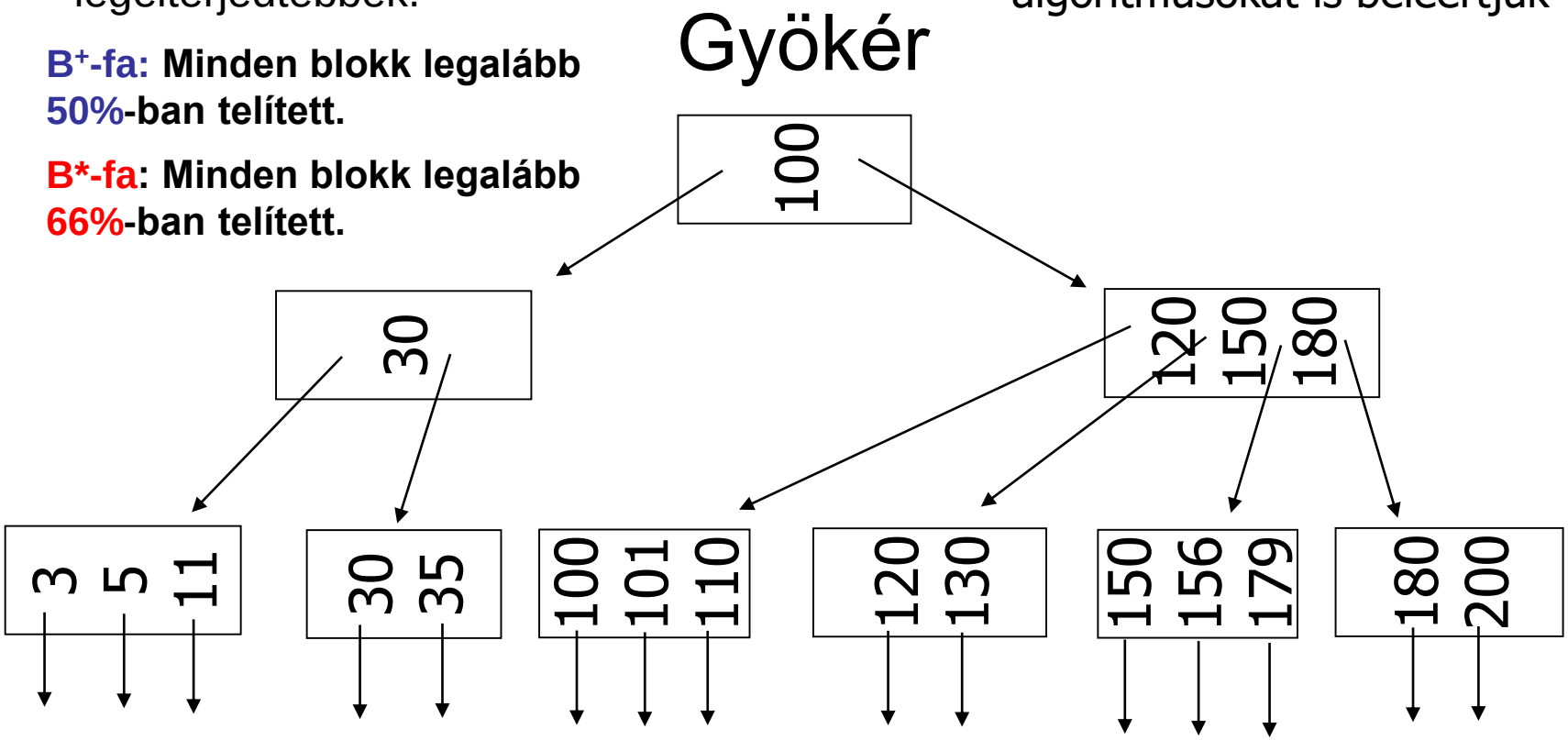
Indexelés

A többszintű indexek közül a **B⁺-fák, B*-fák** a legelterjedtebbek.

B⁺-fa: Minden blokk legalább 50%-ban telített.

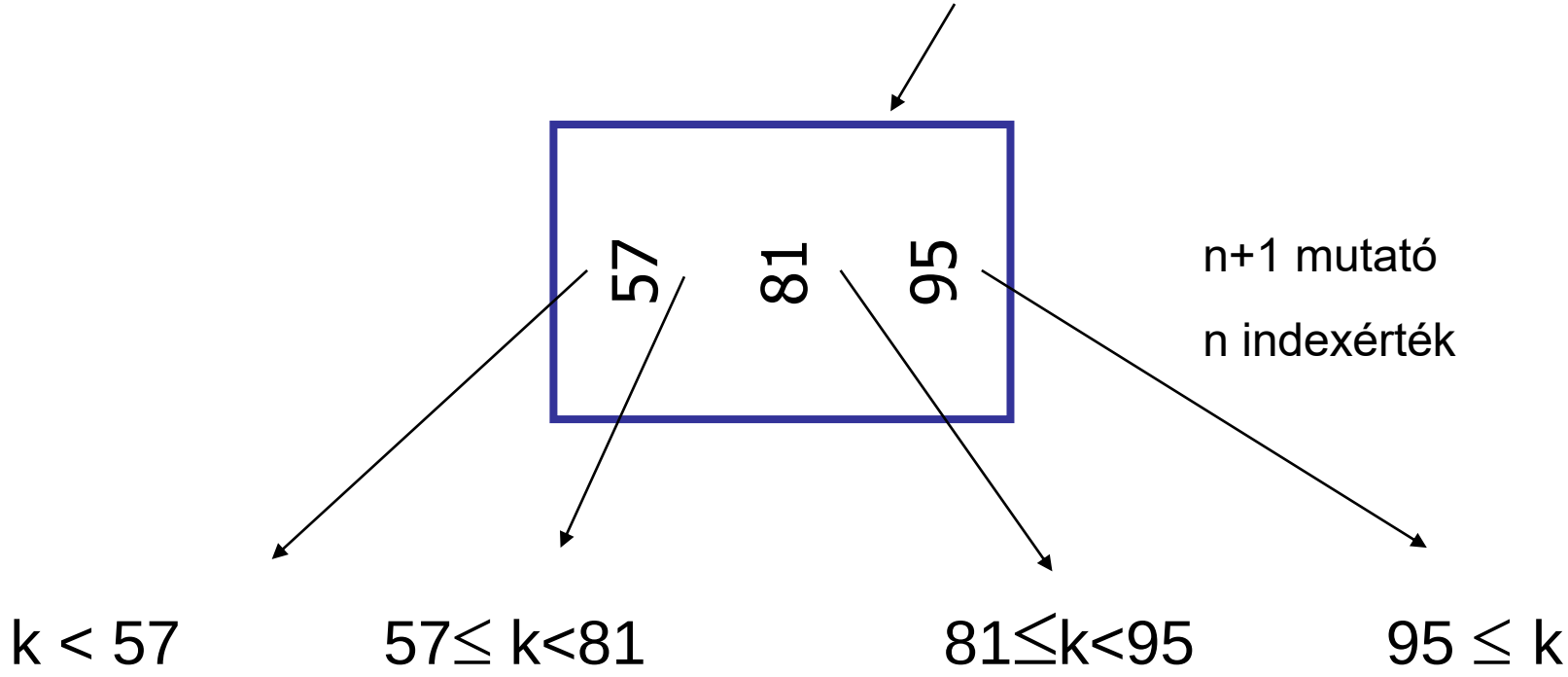
B*-fa: Minden blokk legalább 66%-ban telített.

B⁺-fa: a szerkezeten kívül a telítettséget biztosító karbantartó algoritmusokat is beleértjük



Indexelés

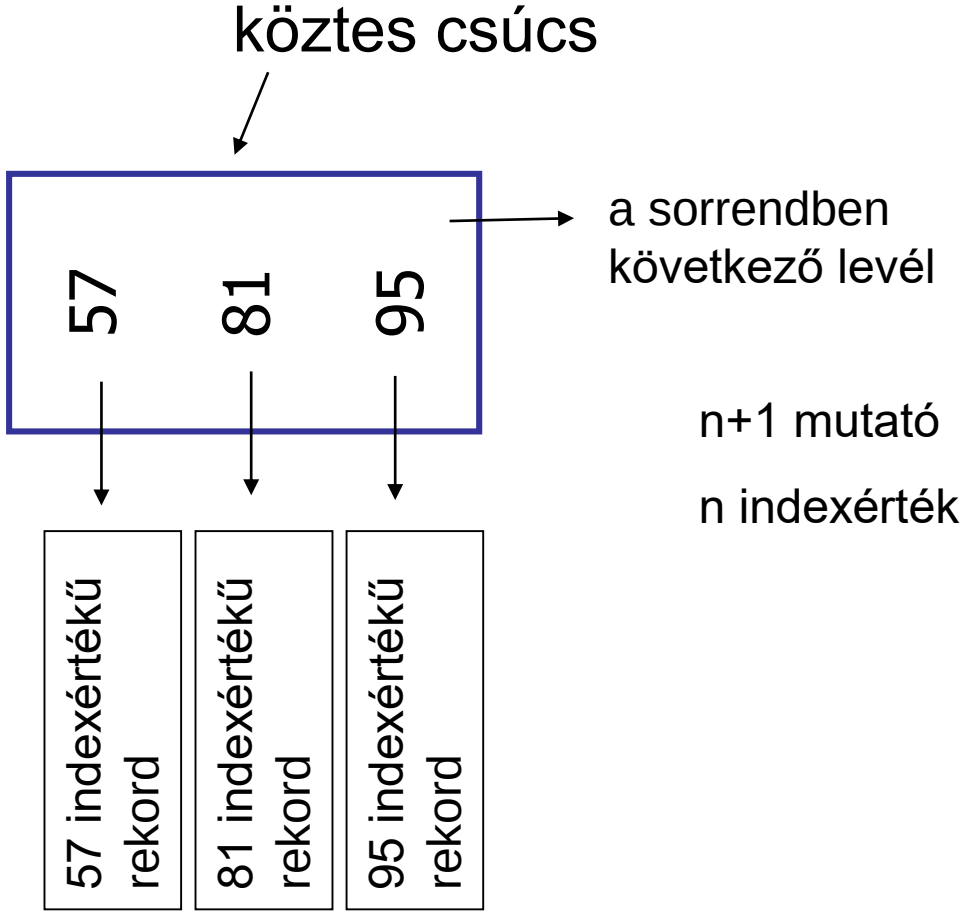
Köztes (nem-levél) csúcs szerkezete



Ahol k a mutató által meghatározott részben (részgráfban) szereplő tetszőleges indexérték

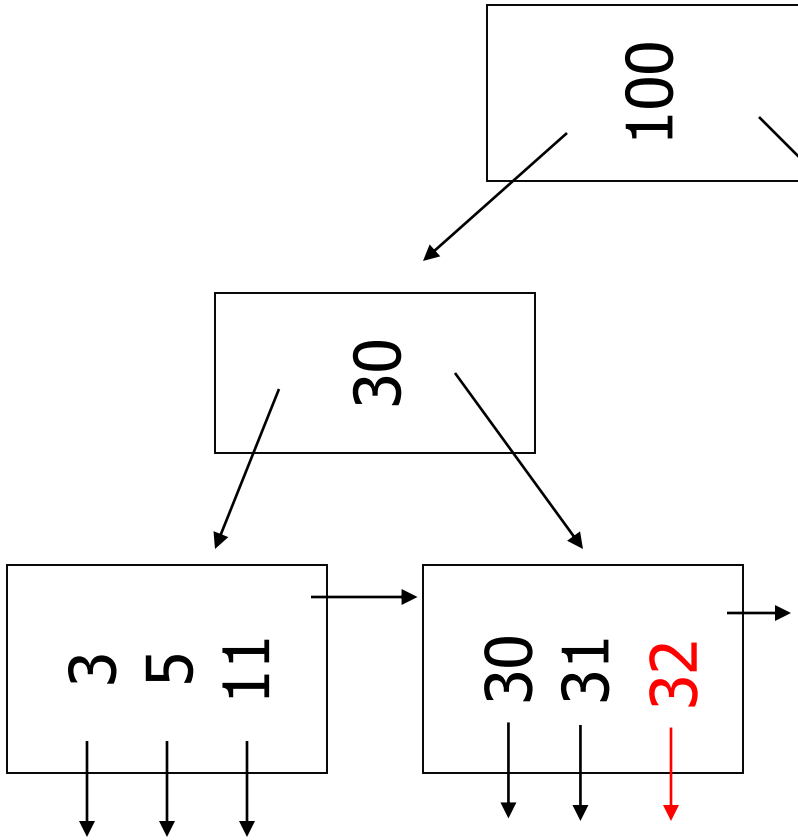
Indexelés

Levél csúcs szerkezete



Indexelés

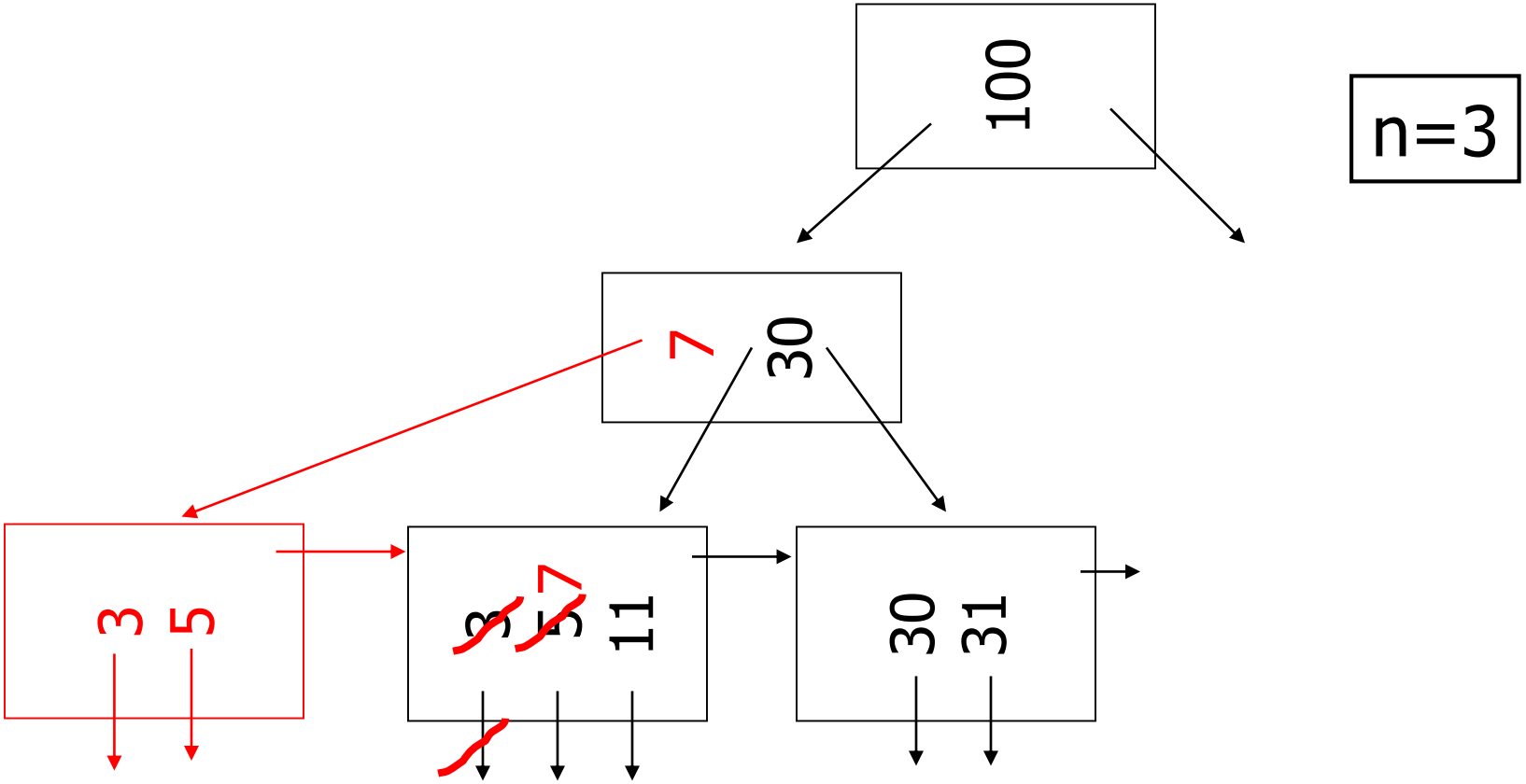
Szúrjuk be a 32-es indexértékű rekordot!



n=3

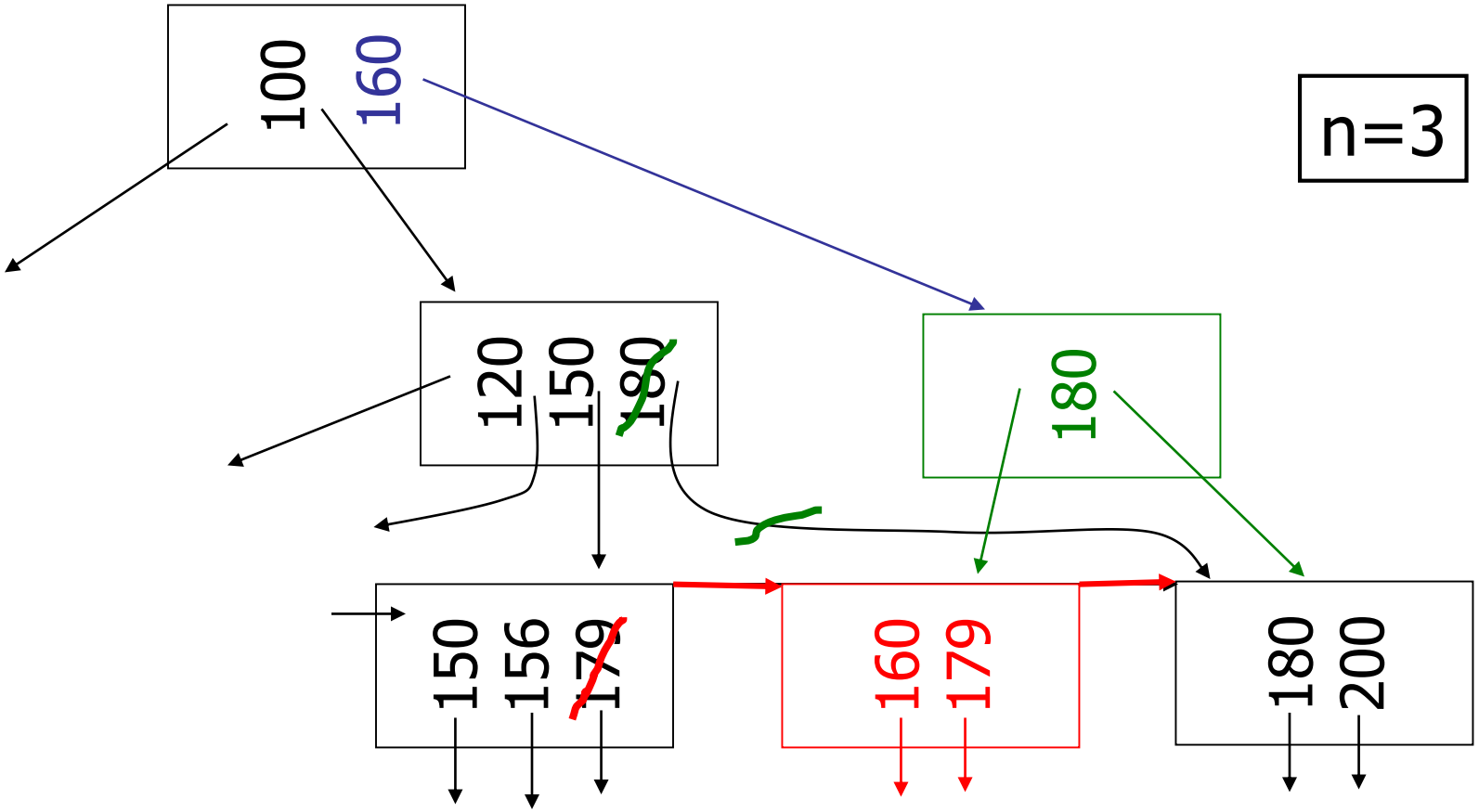
Indexelés

Szúrjuk be a 7-es indexértékű rekordot!



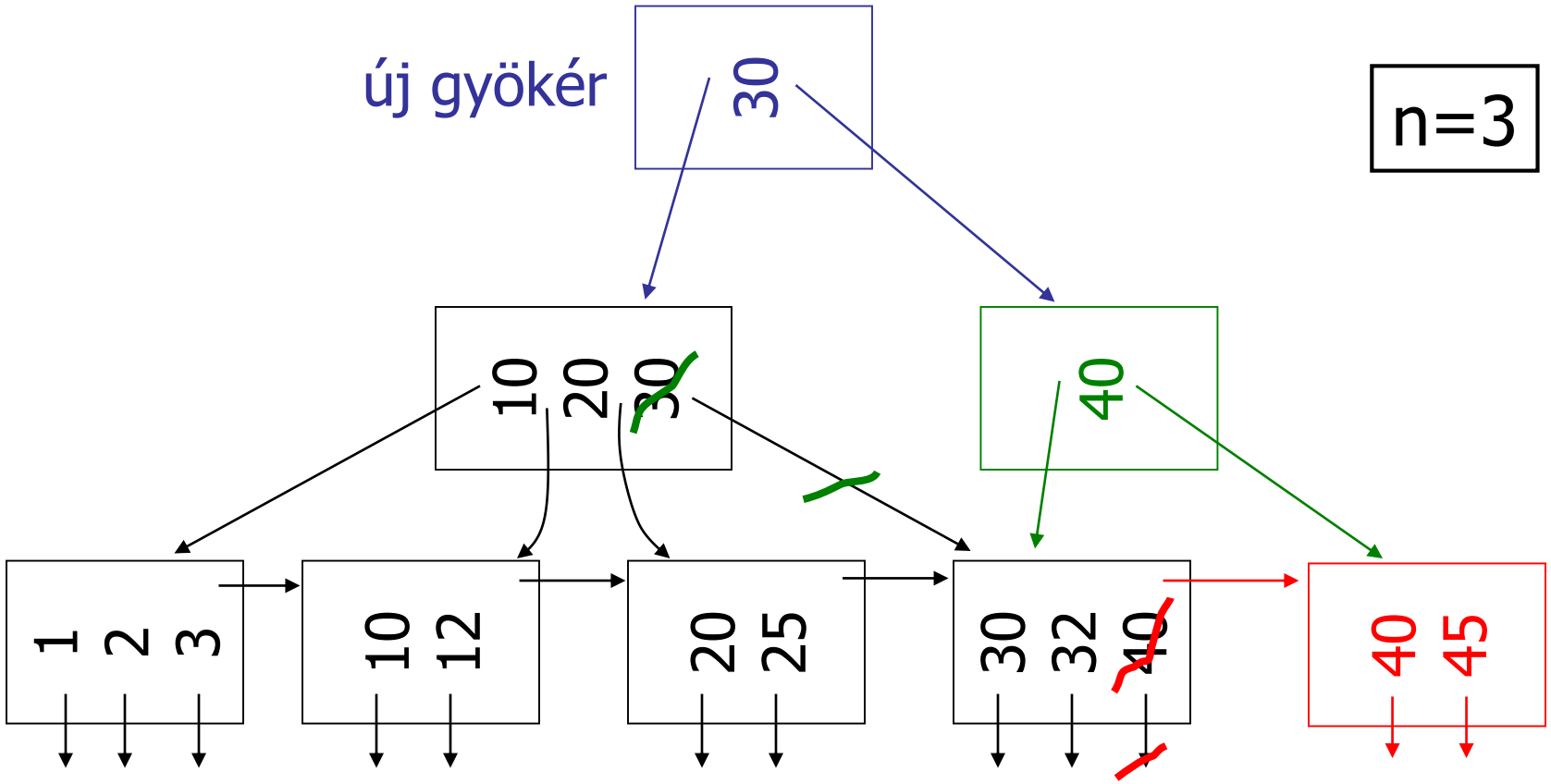
Indexelés

Szúrjuk be a 160-as indexértékű rekordot!



Indexelés

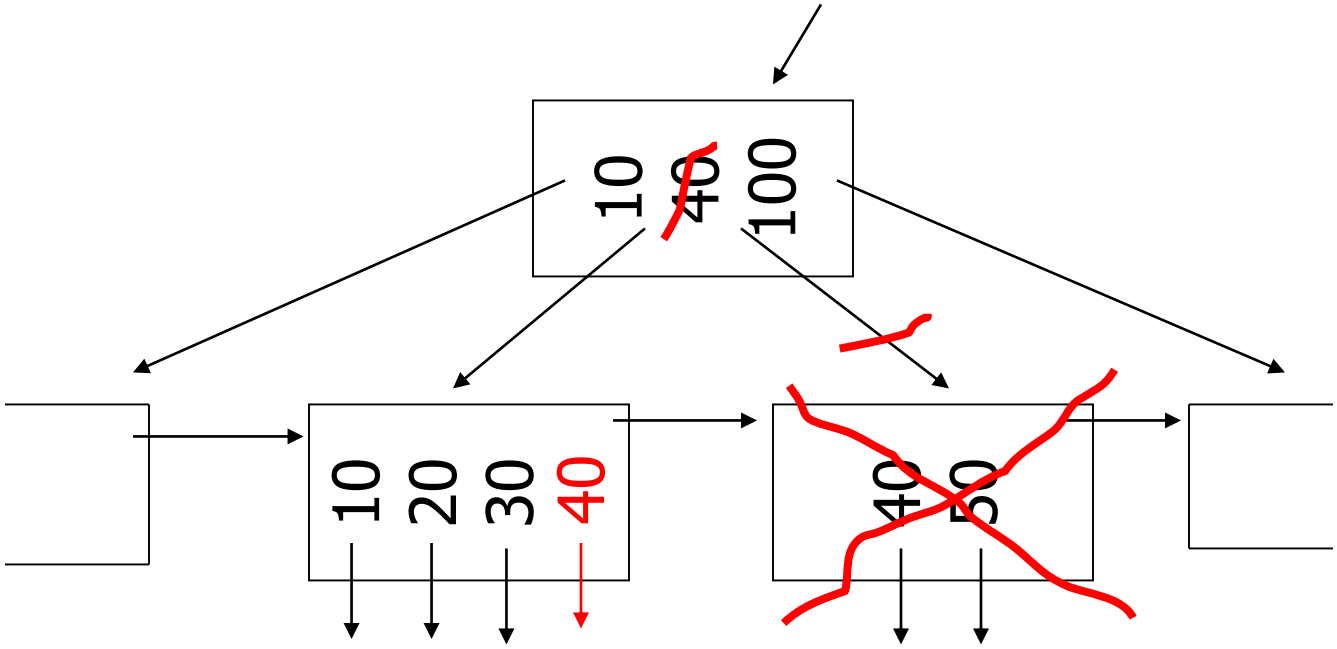
Szúrjuk be a 45-ös indexértékű rekordot!



Indexelés

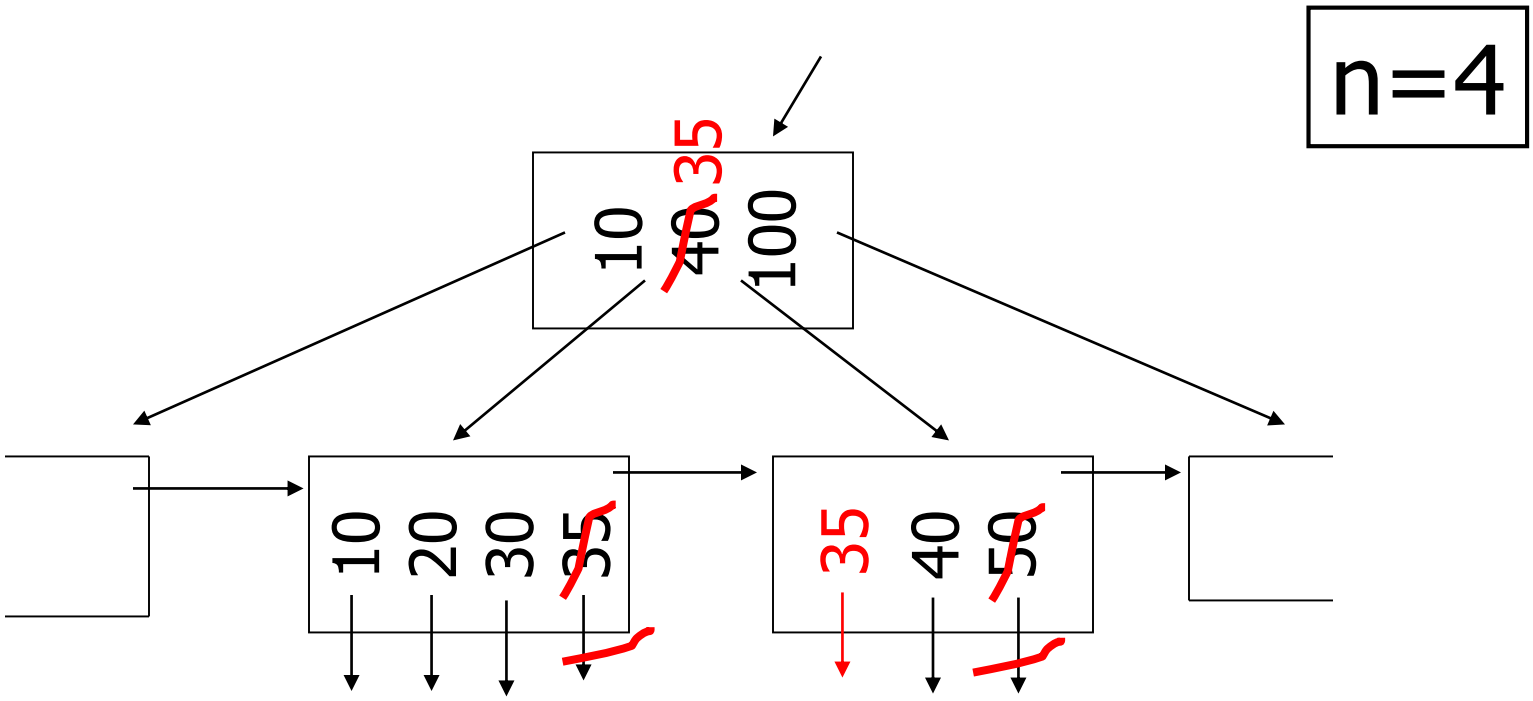
Töröljük az 50-es indexértékű rekordot!

n=4



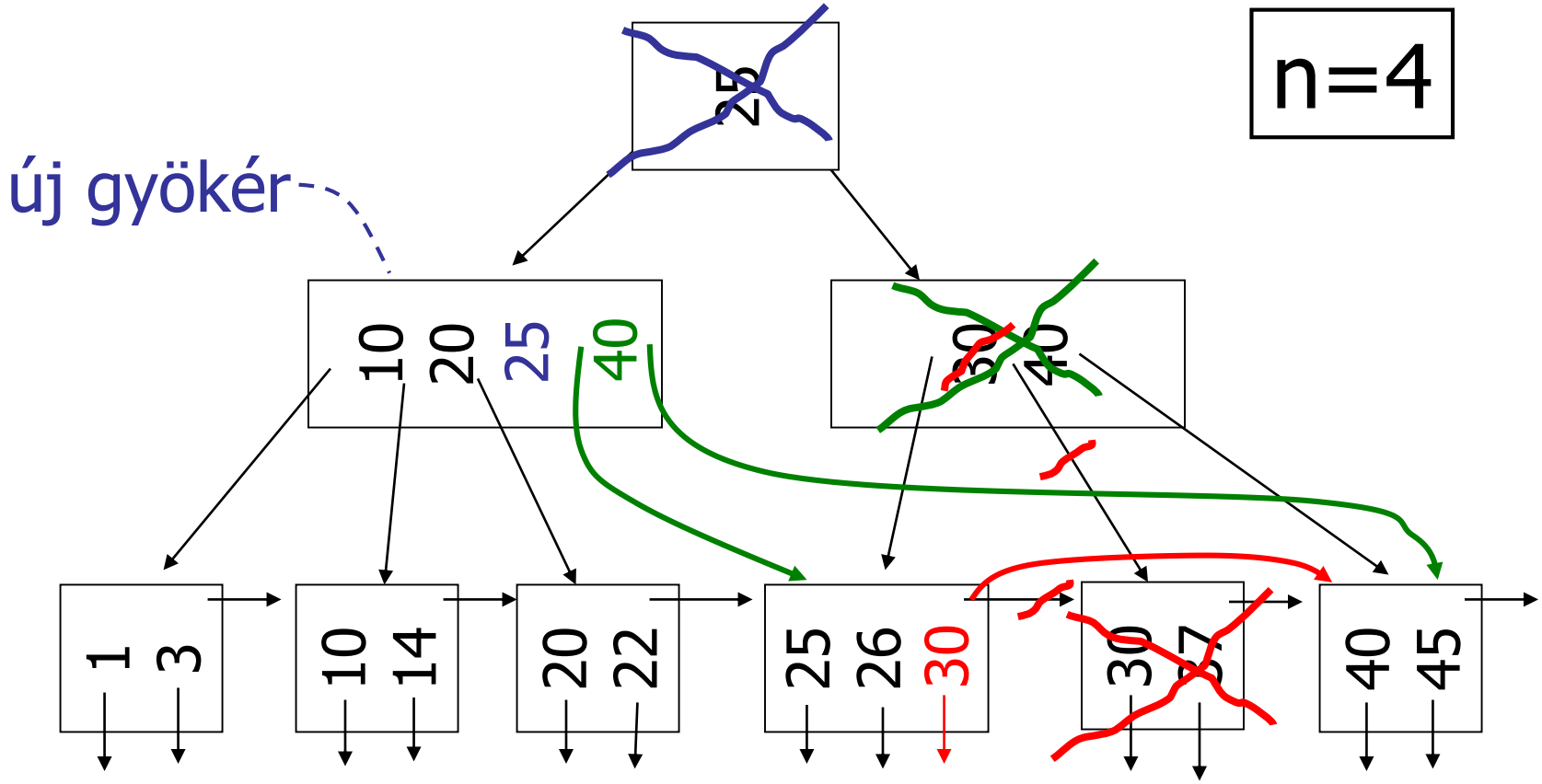
Indexelés

Töröljük az 50-es indexértékű rekordot!



Indexelés

Töröljük a 37-es indexértékű rekordot!



Összefoglalás:
Keresés rendezett állományban,
elsődleges és másodlagos indexekben,
többszintű indexekben, B^+-fákban, B*-fákban

Köszönöm a figyelmet!